Federated Unsupervised Machine Learning

Hartebrodt, Anne

DOI:
10.21996/z4yw-jm67

Publication date:
2022

Document version:
Final published version

Citation for pulished version (APA):
Hartebrodt, A. (2022). *Federated Unsupervised Machine Learning*. [Ph.D. thesis, SDU]. Syddansk Universitet. Det Naturvidenskabelige Fakultet. https://doi.org/10.21996/z4yw-jm67

Department of Mathematics
and Computer Science,
University of Southern Denmark

SDU

Federated Unsupervised Machine Learning

PhD Thesis

Anne Hartebrodt

Supervisor
Assoc. Prof. Dr. Richard Röttger

Feburary 2022

Abstract

Federated learning (FL) is an emerging privacy-aware machine learning paradigm motivated by an increasing request for confidential and private data mining. Federated learning operates under the assumption that raw data cannot leave the data owners computer. Instead, only aggregated parameters can be exchanged between the participants. These can range from simple summary statistics to entire gradients in deep learning. Since, its formal conception in 2016, much effort has been put into the research of federated learning, covering not only the development of efficient and accurate algorithms suitable for decentralized data, but also their privacy.

The biomedical community is in the process of adopting federated learning to provide better care while remaining privacy-aware. In addition to privacy considerations, biomedical data is challenging to work with due to its high dimensionality and the absence of labels. Hence, traditional unsupervised techniques, such as dimensionality reduction and clustering, are widely used in the biomedical domain. However, classical data mining is underrepresented in the federated learning literature.

In order to further promote the adoption of federated learning in biomedical research, this thesis studies popular algorithms for the analysis of biomedical data, more specifically dimensionality reduction and clustering. The present work provides a general introduction to federated learning and the relevant unsupervised algorithms. The remainder of the work consists of a collection of manuscripts, followed by a general discussion.

The first three manuscripts included in this thesis map out the conception of an efficient version of a federated singular value decomposition algorithm (SVD) suitable for high dimensional data, beginning with an assessment of suitable principal component analysis (PCA) schemes for horizontal cross-silo federated learning. Motivated by specific requirements in Genome-Wide Association Studies (GWAS), a federated PCA algorithm for vertically partitioned data is developed. This research is concluded by the conception of a new generic and efficient algorithm for horizontal and vertical data partitioning which, although motivated by the application to GWAS, is suitable for any application in bioinformatics and data science in general.

The third and forth manuscript contain research on potential privacy leaks in the studied algorithms, which includes iterative leakage and potential data reconstruction in two of the PCA algorithms.

The fifth and final manuscript provides an overview and evaluation of fed-

erated clustering strategies using the K-Means algorithm. It then proceeds to develop a strategy to infer the parameter k from the data, a step which is neglected in previous publications, but crucial for the application of federated K-Means in realistic analysis scenarios.

Overall, this work addresses a few challenges in a rapidly developing field. The relative recency of federated learning and the vast field of unsupervised machine learning leave many challenges for future research, including further improvements of the privacy and efficiency of the algorithms.

Dansk resumé

Federated learning (FL) er et voksende fortrolighedsbevidst (privacy-aware) machine learning paradigme, motiveret af en stigende efterspørgsel for fortrolig og privat data mining. Federated learning opererer under præmisen at rå data ikke må forlade ejerens computer. Istedet kan kun aggregerede parametre blive udvekslet mellem deltagere. Disse parametre kan være alt fra simple opsummerende statistikker til gradienter i deep learning. Siden den formelle udformning i 2016 er der blevet lagt en stor indsats indenfor forskning i federated learning, der ikke kun dækker idéen om effektive og nøjagtige algoritmer, der er egnede til decentraliseret data, men også deres fortrolighed og sikkerhed.

Det biomedicinske felt er igang med at vedtage federated learning for at sikre bedre behandling, der stadig forbliver fortrolig. Udover hensynet til fortrolighed er biomedicinsk data udfordrende at arbejde med på grund af dataens høje dimensionalitet samt manglen på overbevisende labels. Derfor er traditionelle unsupervised teknikker, såsom dimensionalitetsreduktion og clustering, meget udbredte indenfor det biomedicinske felt. Klassisk data mining er derimod underrepræsenteret indenfor federated learning literaturen.

For at fremme vedtagelsen af federated learning i biomedicinsk forskning undersøger denne afhandling populærer algoritmer til analyse af biomedicinsk data, mere specifikt dimensionalitetsreduktion og clustering. Afhandlingen giver en generel introduktion til federated learning og de relevante unsupervised algoritmer. Resten af afhandlingen består af samlinger af manuskripter, efterfulgt af en generel diskussion.

De første tre manuskripter, der er inkluderet i denne afhandling, kortlægger idéen om en effektiv version af federated singular value decomposition (SVD), der egner sig til data med høj dimensionalitet. Først er der en vurdering af principal component analysis (PCA) strategier, der egner sig til horisontal cross-silo federated learning. Motiveret af specifikke krav i Genome-Wide-Association Studies (GWAS) leverer vi derefter en federated PCA algoritme til vertikalt opdelt data. Dette afsluttes af en idé til en ny generisk og effektiv algoritme til horisontal og vertikal dataopdeling, som, selvom den er motiveret af applikationen til GWAS, er velegnet til enhver applikation inden for bioinformatik.

Det fjerde manuskript indeholder forskning om potentielle fortrolighedslækager i tre federated QR algoritmer, som inkluderer potentiel datarekon-

struktion i to af QR-algoritmerne. Kun QR dekomposition, som bruger Gram-Schmidt algoritmen, tilfredstiller fortroligheds krav.

Det femte og sidste manuskript giver et overblik og en evaluering af federated clustering ved hjælp af K-Means-algoritmen. Dette efterfølges af en ny strategi til at udlede k-parameteren udfra dataen, et skridt der er forsømt i tidligere publikationer, men er afgørende for anvendelsen af federated K-Means i realistiske analysescenarier.

Samlet set adresserer denne afhandling flere udfordringer indenfor dette hastigt voksende felt. Da federated learning stadig er relativt nyt, mens feltet indenfor unsupervised learning er så omfangsrigt, er der stadig mange udfordringer der kræver videre forskning, herunder yderligere forbedringer af fortroligheden og effektiviteten af algoritmerne.

Acknowledgments

I sincerely thank my supervisor Richard Röttger for his guidance and advice during the past three years. Thanks for leaving me the freedom to work on my projects, yet always being there to support me in my research endeavors.

A big thanks to the members of the Computational Biology Group, Dominika, Mathias, Philipp, Tobias, Tobias, Maria, Juan, Johannes, and everyone else I met during my time at SDU. Without you my time as a PhD student would not have been half as fun!

Thanks to Rudolf Mayer and the crew in Vienna for welcoming me during my lab exchange at SBA Research. Thanks for all the discussions, and the coffee ;)

Of course, thanks to everyone who proofread parts of this thesis. I hope you learned something!

Thanks to my collaborators in the FeatureCloud Consortium[1] and specifically the Platform development group for the great collaboration we established throughout the years. Special thanks to Reza Nasirigerdeh and David Blumenthal for our collaboration on federated Singular Value Decomposition.

A big thanks to all my friends near and far for supporting me all those years.

I am eternally grateful to my family for their love and support throughout the years. I would not be the person I am today without my parents and brothers.

[1]The FeatureCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078. This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

List of publications

Submitted manuscripts included as chapters in this thesis:

1. **Hartebrodt, A.** and Röttger, R. (2021), Federated Horizontally Partitioned Principal Component Analysis for Biomedical Applications *(Under review)*
2. **Hartebrodt, A.**, Nasirigerdeh, R., Blumenthal, D. B., and Röttger, R. (2021). Federated Principal Component Analysis for Genome-Wide Association Studies, (ICDM), 1090–1095. `https://doi.org/10.1109/ICDM51629.2021.00127`
3. **Hartebrodt, A.**, Röttger, R. and Blumenthal, D. B., (2021). Federated Singular Value Decomposition for High Dimensional Data *(Under review)*
4. **Hartebrodt, A.**, and Röttger (2021). Federated QR decomposition – algorithms, privacy, and applications *(Submitted)*

Notable publications not included in this thesis:

1. Matschinske, J., Späth, J., Nasirigerdeh, R., Torkzadehmahani, R., **Hartebrodt, A.**, Orbán, B., ... & Baumbach, J. (2021). The FeatureCloud AI Store for Federated Learning in Biomedicine and Beyond. arXiv preprint arXiv:2105.05734. *(Under review)*

Overview

This thesis consists of an introductory chapter; a total of five manuscripts, four of which are submitted at the time of writing; and a general discussion of the work. The first chapter contains a general introduction to federated learning (FL) and its challenges and a brief presentation of the relevant unsupervised machine learning algorithms. It is followed by three manuscripts discussing federated principal component analysis (PCA) in detail. The third manuscript represents a consolidation of this work, in the sense that it presents a generally applicable, efficient algorithm for federated singular value decomposition (SVD). The fourth manuscript contains work on federated QR decomposition, as an extension of an algorithm presented in the third manuscript, and includes further privacy considerations. The fifth manuscript presents preliminary work on federated K-Means. The final chapter is dedicated to a discussion of the work and a brief conclusion.

Federated Learning

The scope of the first chapter is a motivation and general introduction to federated machine learning. It briefly discusses technical and privacy challenges. It also introduces privacy enhancing techniques, and how they can be applied to make federated learning more resilient against threats. The notion of federated learning is context dependent, therefore this chapter introduces the specific assumptions on the federated learning setup assumed in this thesis. A few of these assumptions are motivated by the FeatureCloud Platform which is described briefly. The presented algorithms are Principal Component Analysis and Singular Values Decomposition, QR factorization, as well as the popular K-Means algorithm.

Manuscript 1: Federated Horizontally Partitioned Principal Component Analysis for Biomedical Applications

This paper represents initial work on Principal Component Analysis for federated horizontally partitioned data. The notion of data partitioning will be explained in the introduction, for now it is sufficient to understand that these algorithms can be used for classical dimensionality reduction, for instance as a preprocessing step in single cell data analysis. In this article, the effect of limited data availability and biased sample distribution on federated PCA is studied and algorithms are analyzed w. r. t. to their data disclosure.

Manuscript 2: Federated Principal Component Analysis for Genome-Wide Association Studies

This manuscript treats the application of PCA for sample stratification in the federated domain, motivated by, but not limited to, the use in federated Genome-Wide Association Studies (GWAS). Notably, the algorithm presented in this article mitigates a potential privacy breach in the consecutive application of federated PCA and the association test, by the application of federated Gram-Schmidt orthonormalization.

Manuscript 3: Federated Singular Value Decomposition for High-Dimensional Data

The third manuscript is an extension of the previous work and includes proofs for both federated SVD and Gram-Schmidt orthonormalization. It extends the works by applying ideas from approximate federated horizontally partitioned PCA and randomized PCA, making the algorithms more communication efficient in terms of data volume and communication rounds. The manuscript also investigates iterative leakage in power iteration.

Manuscript 4: Federated QR decompositition – algorithms, privacy, and applications

The focus of the article is on the suitability and privacy of three popular algorithms used for QR decomposition, Householder reflection, Givens rotation, and Gram-Schmidt decomposition, when deployed in a federated setting. This chapter describes the full Gram-Schmidt algorithm, extending the previous description of the method in Manuscripts 2 and 3 which focused solely on the orthonormalization, but omitted the return of the upper triangular matrix. A specific algorithm for PCA, which can be solved using federated QR decomposition is studied to understand, whether any privacy can be gained by choosing this algorithm over the ones previously discussed.

Manuscript 5: Federated K-means

This article contains an evaluation of existing federated initialization and clustering strategies for the popular K-Means algorithm proposed in the literature. Furthermore, a comprehensive evaluation strategy for federated clustering of non-iid data is developed. A gap in research is the federated

determination of a good choice of k, therefore in this chapter an approach for this problem is presented and evaluated, using the suggested evaluation scheme.

Discussion and Conclusion

The last chapters are dedicated to a joint discussion of the articles with respect to the challenges in federated learning introduced in the first chapter. The major challenges addressed in the present work are privacy, communication efficiency, and data heterogeneity, as well as the accuracy of the algorithms. Naturally, this discussion will reveal future research directions in the field of unsupervised federated learning.

Contents

Chapter 1

Federated Learning

1.1 Motivation

The growing availability of data and compute power has lead to a sharp increase in the use of machine learning in areas ranging from commercial applications to basic research. Machine learning relies on large data rather than expert curation to create models for the prediction and understanding of phenomena. The field can be broadly categorized into **supervised** and **unsupervised** machine learning (Hastie 2017). Currently, (deep) neural networks are the method of choice for many tasks in both supervised and unsupervised machine learning.

A popular instance of supervised learning is classification where the task is the prediction of the label for a data point which has previously not been "seen" by the algorithm. Supervised learning requires the presence of a labeled data set. During training the hyperparameters of the model are adjusted to minimize the difference of the predicted class and the true labels. A pitfall in supervised learning is "overfitting" which means the model achieves very good accuracy on the training data, but generalizes poorly, and cannot be used on new, unseen data. Overfitting is especially difficult to overcome with small or biased data sets (Hastie 2017).

Unsupervised learning is a group of techniques relying on the intrinsic properties of the data to create explanatory models, and can therefore be used on label free data. This includes for instance clustering, which has the goal of finding subgroups in the data, and dimensionality reduction. In computational biology, unsupervised machine learning plays an equally important

role as supervised learning, because biological data is not necessarily labeled, or the labels are biased and therefore not amenable to supervised learning (F. Li et al. 2021). The recent trend towards deep-learning in bioinformatics does not impede the use of the classical techniques. Karim et al. (2021), for instance, suggest to combine deep learning based feature embedding with traditional clustering. A practical instance of this approach in modern bioinformatics is for example the combination of deep generative modeling (Lopez et al. 2018) with neighborhood embedding (Mcinnes et al. 2018) and clustering (Traag et al. 2019) in single cell transcriptomics.

Both supervised and unsupervised machine learning rely on the availability of high quality data. In many areas of computational biology, the number of available data sets has risen in the recent past (Drysdale et al. 2020), but not all have seen an equal trend in their growth. Areas that still suffer from data scarcity are rare diseases, which affect fewer than *2000* patients, according to the definition of the European Union (Kerr et al. 2020). It has been shown that many trials in rare diseases fail due to the lack of a large enough cohort (Rees et al. 2019). A second area in bioinformatics that would benefit from larger data sets are Genome-Wide Association Studies (GWAS). GWAS try to find associations between genotype and phenotype using statistical tests and large cohorts. While with 500 000 participants, the UK Biobank is sizable (ukbiobank.ac.uk 2022), this does not imply a sufficient diversity (Sirugo et al. 2019). Many GWAS studies are performed on cohorts of predominantly European ancestry which only represent a small fraction of the world's population (Mills and Rahal 2019). More diverse studies would significantly strengthen the results of the analyses, but the access to genetic data is understandably strictly regulated. Overall, the data required to learn trustworthy models is not always as readily available as it seems at the first glance.

Biomedical data contains many sensitive attributes and allows to infer a high amount of information on the individual, such as genetic risk factors and disease status. They must remain private to protect the individual from harm. Furthermore, the data is at least partially immutable. The genome is not only stable over an individuals lifetime, it is also partially transmitted to the next generation (Bonomi et al. 2020). Bonomi et al. (2020) provide an overview of attacks using public genetic data exposing the individual and assess mitigation strategies. As a result, they suggest to further research privacy-aware genomics. Infectious diseases such as HIV can be managed well today, but may still expose the affected to serious stigmatization, if this information is disclosed to a third party (Feyissa et

al. 2019). Wearable devices such at fitness watches monitor daily activities and physical conditions. While useful to assess the overall and individual level of health of a population, the data should not be accessible to third parties. Commercial solutions have been shown to be vulnerable to attacks (Fereidooni et al. 2017). The public research must therefore promote privacy-aware technologies and data mining.

The access to clinical data has always been tightly regulated (Veen 2018). The introduction of the General Data Protection Regulation (GDPR) in Europe further discourages arbitrary data sharing to protect the individual. It partially motivates the interest of the research community in data privacy. Two main avenues are under active investigation, (1) the generation of synthetic data, and (2) federated learning. Synthetic data is generated by machine learning models which have been trained to reflect the properties of the original data, for example the marginal distribution of covariates (Gootjes-Dreesbach et al. 2020). Federated learning is a more generic approach making no assumptions on the nature of the question asked (Rieke et al. 2020). Federated learning allows decentralized data to be analyzed as if they were located on a single server, but the raw data is never moved outside of the data owners computer. Local models are trained on the local data and *only* the models are sent to an external party which aggregates the model into a global model. The models are assumed to reflect only general properties of the data not allowing to infer information on the individuals in the study.

According to the GDPR, the data controllers are the instance that decides, if and how the data is processed (Veen 2018). FL would allow data controllers to join machine learning based studies, while complying with legal regulations within their jurisdiction, because they retain the control over the data. In medical research, the potential of federated learning has already been demonstrated to outperform centralized models, as more data could be used for the training. FL has been used for tasks such as physical and mental disorder prediction and is applicable to the prevention, monitoring and management of diseases (Yoo et al. 2021). While compelling, legal and ethical hurdles are not the only argument for federated learning. Gaye et al. (2014) raise the problem that even scientists are concerned about sharing "their" raw data. They prefer to share analysis results because they invest significant effort in the curation of the data sets and are concerned about intellectual property. Federated learning allows partners who, in other circumstances would not be incentivized to collaborate, to perform a joint study. This can for instance be private companies or banks (Liu, Fan, et al.

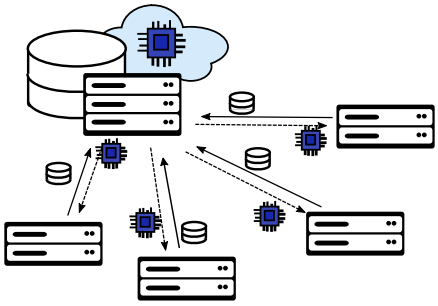
2021) that do not want to share their data for economical reasons. Mobile device users, who are reluctant to share their private information with a global server, can participate in federated learning and profit from better models trained on data that remains private (Hard et al. 2018).

As the potential benefits have been recognized by the biomedical community, federated learning is an active area of research. Not all questions have been answered to as satisfying consensus yet. A major question is whether the privacy promises made by federated learning hold true in practice. Furthermore, technical challenges such as communication efficiency and model accuracy are not solved yet. The focus of many authors (Kairouz et al. 2021; Q. Li et al. 2019) is on federated deep learning, and some admonish the lack of "non-deep-learning" research. Therefore, this thesis is dedicated to the study of **unsupervised federated machine learning** in the biomedical domain. The thesis contains a general introduction to the relevant and most prominent challenges in FL, followed by five manuscripts presenting individual research projects. Chapters 2 to 4 study the specific problem of federated Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). The fifth chapter presents insights into the privacy of QR decomposition. Chapter 6 contains unpublished research on the K-Means algorithm. The thesis concludes with a discussion of the presented work.

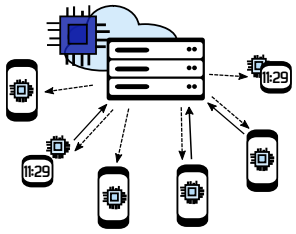
In the remainder of the current chapter, the concept and challenges in federated learning will be presented. In sections 1.2.1 to 1.2.3 the required basic notions of federated machine learning will be introduced. The following section 1.3.4 and section 1.4.1 will discuss threats to federated learning and data privacy as well as mitigation strategies. An additional section is dedicated to the brief presentation of the FeatureCloud platform. The final section 1.6 of this chapter will introduce the relevant centralized unsupervised algorithms and the aims of this thesis.

1.2 Federated learning

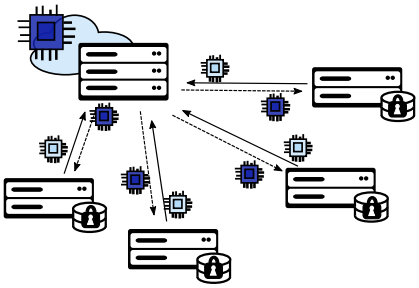
Motivated by the difficulties in accessing medical data, federated learning has been introduced into the medical domain. The term federated learning (FL) is relatively recent and is attributed to a publication by McMahan et al. (2017) although the concept of decentralized computation has existed previously. Federated learning is a branch of machine learning that deals with data that is physically distributed among several machines or devices. These machines are also referred to as clients or participants. The data is private and cannot be send to a central server to be analyzed. Instead, the



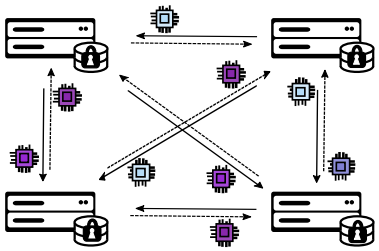
(a) Centralized learning – data and model in the cloud.



(b) Cross-device FL – data on device, model in the cloud



(c) Cross-silo FL (Centralized) – data in silos, model at the aggregator



(d) Cross-silo FL (Decentralized) – data in silos, model on the edge

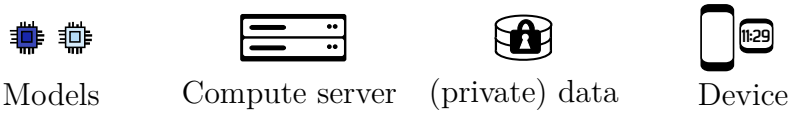


Figure 1.1 – Different types of federated learning currently discussed in the literature.

machines are connected via a network and exchange aggregated statistics and parameters that can be combined in a global model. This learning process can involve multiple rounds of network communication and arbitrarily complex parameters, such as gradient updates in deep learning (Kairouz et al. 2021).

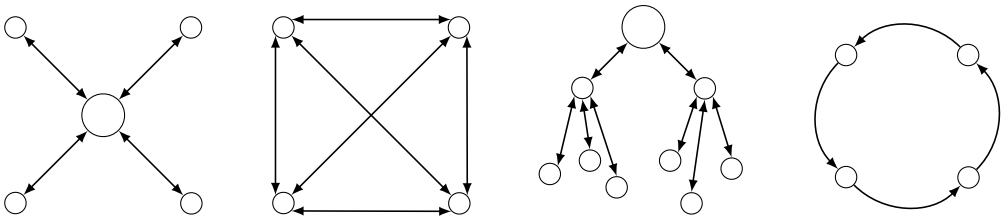
1.2.1 Cross-silo vs. cross-device federated learning

The data is distributed among several data holding machines. Two main branches of FL have emerged, **cross-silo** and **cross-device** federated learning. The distinction is important because it has implications on the requirements of the federated learning system.

Cross-device FL is characterized by a very high number of loosely connected devices (up to 10^{10}) such as mobile phones or sensors which join the learning process. These devices have access to limited data, such as the data of one mobile phone user, and their compute power is limited. The number of data points may vary from device to device. The system is assumed to be dynamic, meaning the devices may join and drop out of the learning process at any time, for instance through device failure and time zone differences (Bonawitz et al. 2019). The devices are connected via slow connections such as mobile or wireless networks which represents the major bottleneck in cross-device federated learning. The high number of clients popularizes stochastic learning processes where only a randomly selected subset of clients sends their parameter updates in each round of federated learning (Kairouz et al. 2021).

In contrast, cross-silo federated learning deals with a static network of a few "data silos" connected by a high-speed internet connection. These data silos have access to larger data bases including observations for multiple individuals. Such data silos may be hospitals or public institutions dealing with the conflicting interests of data protection and societal advancement. Typical numbers of clients range between 2 to 100 sites connected in the collaborative learning environment. The clients are assumed to be reliable, meaning they remain online during the entire learning process. Due to their low number and high reliability, the clients usually participate in every round of computation. In cross-silo FL, network communication may not be the major bottleneck, if time consuming local computations are required (Kairouz et al. 2021).

Although the focus of this thesis is on cross-silo federated learning, both cross-silo and cross-device federated learning promise benefits (Xu et al. 2021; Nguyen et al. 2021; Rieke et al. 2020) and are used for biomedical FL (Yoo et al. 2021).



(a) Star-like **(b)** P2P **(c)** Hierarchical **(d)** CIFL

Figure 1.2 – Architectures in federated learning. (a) Star-like architecture with a central server: the clients communicate only with the aggregator. (b) Decentralized architecture: the clients use P2P communication to exchange parameters. (c) Hierarchical architecture: the clients communicate with a local hub, the hubs communicate with the server. (d) Cyclic institutional FL: the clients receive parameters from a single client and communicate the updated model to the next participant.

1.2.2 Federated system architectures

The design of a federated system is not trivial and subject to many considerations. The base assumption is the availability of a set of machines that are connected in a network. Depending on the number and nature of clients, the computers can assume different roles in the system. Star-like architectures rely on a single powerful aggregation server which receives updates from the clients and orchestrates the learning process. This architecture is used for instance in the FeatureCloud platform (Matschinske et al. 2021) and the assumed federated learning architecture in this work. The central server has high internet connection speed, and the centralized models have favorable convergence properties (Q. Li et al. 2019). The disadvantage of this architecture is that the central server is a "single-point-of-attack", both for the physical system and the learned models which can be problematic for privacy and fairness (Yoo et al. 2021). Most proposed solutions assume a *trusted* central sever which is difficult to achieve in realistic FL scenarios (Q. Li et al. 2019). Furthermore, the system does not scale well to FL with more participants, if the central aggregator works as the only aggregator (Nguyen et al. 2021).

In contrast, in a fully decentralized network all clients have the same privilege. The model exists only on the "edge", meaning it is created simultaneously at all clients, and not at a central aggregator (Warnat-Herresthal et al. 2021; Roy et al. 2021). This has the advantage of not requiring a trusted central party, but increases the communication overhead (Q. Li et al. 2019).

The learning process in fully decentralized learning can involve randomized client selection and "gossiping", the communication with close neighbors in the network Sluciak et al. 2012. Another coordinator free architecture is a cyclic network where the model updates are made in a sequential fashion and used for instance in the "Personal Health Train" (Beyan et al. 2020). This architecture achieved worse performance compared to aggregator based FL due to "forgetting", where more recent updates contribute to the model overproportionally (Sheller, Edwards, et al. 2020). Therefore it is less privacy aware, as the previous client in the chain can be attacked more easily (Pustozerova and Mayer 2021). There are also hierarchical FL systems, featuring local hubs which are interconnected (Rieke et al. 2020; Liu, Ma, et al. 2020). Targeted more towards cross-device FL, Bonawitz et al. (2019) suggest multiple roles, including a controller, which organizes the learning, aggregators, which perform the model aggregation, and selectors choosing the clients whose data is used for the updates. This system has a larger emphasis on a variable number of devices which is not as crucial in cross-silo FL. While the network determines the update strategies to some extent, any system that is able to address all its clients (an assumption of cross-silo FL (Kairouz et al. 2021)) can run the same algorithms and it is not unlikely that other hybrid federated learning architectures will emerge. Q. Li et al. (2019) briefly present a selection of open source federated learning systems.

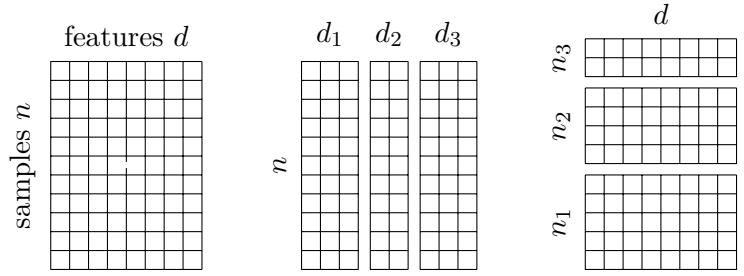
1.2.3 Data partitioning in federated learning

In federated learning, data is distributed over several distant sites and cannot be shared in its raw form between the different participants. Data in this context consists of data points, which may have an arbitrary number of dimensions, and may also be referred to as samples, individuals, or patients depending on the context. Dimensions are also referred to as features or variables and represent measurements. In federated learning, data can be partitioned in different ways.

Horizontal partitioning describes the case where the individuals are distributed over the sites, but all variables are measured for these individuals (S. X. Wu et al. 2018; Kairouz et al. 2021; Yoo et al. 2021; Q. Li et al. 2019; Gaye et al. 2014). This would for instance be the case, if electronic health records with the same features are available at all S sites for different populations, resulting in tabular data with n_s patients and d dimensions at each client $s \in [S]$.

Vertical partitioning refers to the complementary case, where at sites s for all n patients a different subset of features d_s would be measured (S. X. Wu et al. 2018). This is for example the case, if one participant has access to the electronic health records of the population while another site possesses other information such as tumor histology or laboratory results. It is still challenging to obtain high quality test data for this scenario (Kairouz et al. 2021).

Arbitrary partitioning refers to the case where neither site possess the full range of all samples or variables in the study. In this scenario federated transfer learning can be used (Kairouz et al. 2021). The notion of data partitioning is a factor influencing the design of the algorithms, as the parameter integration differs depending on how the data is distributed. According to Q. Li et al. (2019) the majority of the algorithms to date are developed for horizontal partitioning, partially because it is the more frequent case in commercial mobile applications.



(a) Full data set **(b)** Vertical partitioning **(c)** Horizontal partitioning

Figure 1.3 – Data partitioning in FL. (a) The data consists of n samples with d features. (b) Vertical partitioning: the clients have access to different features for the same set of samples. (c) Horizontal partitioning: the clients have different subsets of samples but the full feature space.

1.3 Challenges and opportunities in federated learning

Federated learning is an emerging research field which presents ample opportunity to contribute to the solution of various open problems. The relevance of the challenges depends on the subfield of federated learning, yet some generally relevant challenges have emerged. This section will describe

the more prominent open problems in the literature and briefly summarize other miscellaneous issues raised by various authors in different contexts. The major challenges are the design of resource and communication efficient algorithms, their performance on non-iid data, as well as the privacy and security of the learning process and the models.

1.3.1 Network communication

A major challenge for federated learning is the network communication that the learning process requires, more specifically the size and the number of updates. In classical data centers, the internal communication is fast and usually not considered as much in performance optimization. In FL, every network communication introduces a delay. In cross-device FL, device-to-device communication is more expensive compared to the communication with a server (Marfoq et al. 2020) and the communication is generally slow. In cross-silo federated learning, with its relatively low number of clients and higher network bandwidth, this problem is potentially less severe compared to cross-device federated learning. Assuming that the system uses data centers that are connected by high speed internet connections, the delay may even be neglected (Q. Li et al. 2019; Marfoq et al. 2020). However, Marfoq et al. (2020) also show that by optimizing the topology of the network connection, the time spent on the learning can be decreased in cross-silo FL, making the system less prone to congestion at the aggregator. Regardless of the type of FL, the number of communication steps is a novel factor to be optimized and considered when studying the convergence of the algorithms. Standard secure computation techniques (cf. section 1.3.4) increase the number of required communication steps quadratically with the number of clients, making an overall low number of steps beneficial.

A number of strategies have been introduced to decrease the number of local parameter updates sent to the aggregation severs. An example is the execution of multiple local rounds of parameter updates, before sending the model to the aggregator (Xu et al. 2021; Sheller, Reina, et al. 2019). This is not yet theoretically founded, and it is unclear whether the suggested strategies are generically applicable, as they could negatively affect convergence (Rieke et al. 2020; Kairouz et al. 2021). The size of the updates is the other important factor to be optimized to reduce transmission costs and avoid delays for networks with low upload speed, such as mobile networks. Therefore, the compression of local and global parameters are under active research (Q. Li et al. 2019; Kairouz et al. 2021; Xu et al. 2021). The reduction in size of the model has additional benefits such as increased scalability

and easier model deployment (Kairouz et al. 2021). Overall, the communication overhead of federated learning is a novel factor to be considered in the choice and development of the algorithms.

1.3.2 Heterogeneity of the data and compute resources

One challenge extensively researched is the resilience of federated learning against non independently and identically distributed data ("non-iid-data"). As the data comes from many different servers or devices, the data sourcing process is not identical. Kairouz et al. (2021) and Yoo et al. (2021) review a number of challenges for federated learning, including covariate shift, where the marginal distributions differ for groups of clients; prior probability shift, where the prior probability of events differs for groups of clients; concept drift, where similar concepts are expressed using different features; concept shift, where the same concepts have different labels; and quantity skew, where clients have different sample sizes. In cross-device federated learning, the geographical location is a strong factor, because the it influences not only the device availability but also the observable data and labels (Kairouz et al. 2021; Yoo et al. 2021). For cross-silo FL, the time-dependent availability of the compute severs is less problematic, due to their higher compute power and dedicated purpose, however data-related challenges remain (different ancestry, nutrition, exposure, ...).

The non-iid-ness of the data influences for instance the convergence behavior of gradient decent in the federated setting, which has been studied theoretically and practically (Sheller, Edwards, et al. 2020). Different strategies have been introduced to adapt to the heterogeneity of the data using specialized versions of gradient descent or other techniques such as clustering (Q. Li et al. 2019; Yoo et al. 2021; Huang et al. 2019; Sattler et al. 2019). The decentralized nature of the data also introduces difficulties in verifying the source of deleterious behavior in FL, intentional or unintentional, and holding the responsible clients accountable (Yoo et al. 2021). A potential medical federated learning system consists of participants with different data quality and quantity as well as different access to resources. The system should not encourage participants to influence the learning process negatively, or non-contributors to profit inadequately. Game theoretic approaches to optimize the value of participation have been suggested (Yoo et al. 2021).

While non-iid-ness it is a challenge in federated learning, it also presents a number of opportunities. For instance, instead of creating a single, global

model which performs adequately for everyone, it is possible to create personalized models for subgroups of participants which perform better for the respective cohort (Xu et al. 2021; Tan et al. 2021). Overall, the heterogeneity inherent to FL systems can be deleterious to the model if not properly accounted for, but, on the other hand, represents an opportunity for increasing the fairness of machine learning.

1.3.3 Other challenges in federated learning

While the aforementioned challenges are prominent in the literature, a number of additional challenges have been mentioned by various authors. Yoo et al. (2021), Q. Li et al. (2019), and Rieke et al. (2020) raise the issue of proper incentive to participate in federated learning. Medical data sets are expensive to curate, therefore data providers might want to prevent non-contributing parties to profit from their work. Consequently, proper revenue models will have to be developed (Rieke et al. 2020). Furthermore, the data acquisition process (Sheller, Edwards, et al. 2020) and hospital infrastructure are heterogeneous (Rieke et al. 2020), making the data integration difficult even with willing participants. For medical applications, the FL learning system also needs to be reproducible, traceable (Rieke et al. 2020) and auditable (Passerat-Palmbach et al. 2019). Additionally, the incorporation of expert knowledge, while beneficial, is challenging (Xu et al. 2021). Naturally, the model accuracy and fairness is primordial for federated learning with potential diagnostic application (Xu et al. 2021; Yoo et al. 2021; Q. Li et al. 2019).

1.3.4 Privacy and Security

Although used interchangeably, privacy and security have different scopes (Andriole 2014). In the medical domain, patient privacy refers to *"the right of patients to determine when, how, and to what extent their health information is shared with others"* (Andriole 2014). Security refers to the measures taken to prevent harm from coming to the patient. Harm can manifest physically, for instance if proper treatment cannot be guaranteed due to nonavailability of the medical records. Therefore, secure systems should allow only authenticated and authorized used to access the data, keep the data confidential, and maintain data availability, integrity, and auditability (Andriole 2014).

Initially, the promise of FL was that the privacy of the participants was secured by not sharing their data with third parties. However, this view

was very quickly abandoned as attacks on federated models in particular, became known (Sun et al. 2019; Bagdasaryan et al. 2018; Nasr et al. 2019). Due to their importance, threats to federated learning and mitigation strategies will be discussed in more detail in the following section.

1.4 Private Federated Learning

In an ideal world, honest participants would join a learning process with their data and no party would learn anything apart from the final global model. Alas, we do not live in an ideal world. As explained in section 1.2, FL makes more data available to train better models, while mitigating a major concern of data controllers by preventing uncontrolled access. However, other threats to the data confidentiality and privacy remain, and novel challenges emerge. For medical research for instance, we assume that the learned models are published for public benefit. Publication does not necessarily imply the full disclosure of the model, including all the weights; it can also be deployed as a queryable model, such that only input and output are observable. Regarding the final model FL is conceptually subjected to the same threats as centralized machine learning, for example model stealing (Paleyes et al. 2020).

FL introduces new challenges through the data federation. Since the data remains in the silos, it is difficult to verify the integrity of the data, and the local updates (Augenstein et al. 2019). Furthermore, the local updates can disclose information about the local data that would not be inferrable from a global model. Research on the the privacy of machine learning and federated learning is a dynamic field with many emerging ideas. This section gives an overview over the most popular strategies for data privacy protection. Section 1.4.1 introduces the terminology and privacy challenges specific to federated learning, the following sections 1.4.2 to 1.4.6 discuss mitigation strategies, such as homomorphic encryption (HE), secure multiparty computation (SMPC) and differential privacy (DP), as well as their combination in hybrid federated learning.

1.4.1 Setting and terminology

In the context of federated learning, there are multiple actors which contribute to the learning process. An actor in the system who tries to learn more information than they should, according to the setup of the study, is called an attacker or adversary. All participants in the process can technically be adversaries: the clients, the aggregators, the analyst, and outsiders

who gain information for instance on the final model (Kairouz et al. 2021). It is assumed that all technical measures to secure the system, such as encryption of the transferred parameters and user authentication have been taken, so that outsiders have a very limited view of the system. Hence, the focus is on the information inferrable from the shared parameters and models available to active participants.

"Insider attackers" have a functional role in the training and can gain insight in the data, intermediate parameters, the architecture of the model, and the final results or subsets of those. Their attacks are stronger, and the main focus of the literature (Enthoven and Al-Ars 2021). There are broadly two types of attackers. **An honest-but-curious** or passive adversary follows the protocol, but tries to infer more information about the other participants than they should (Snyder 2014). These attacks are difficult to detect because they leave no trace in the federated system (Enthoven and Al-Ars 2021). **A malicious participant** can deviate arbitrarily from the training process and protocol, such as disrupting the training or injecting falsified data (Snyder 2014). Two or more attackers can collaborate, to target another participant in the training. This is called collusion. In healthcare consortia, Rieke et al. (2020) claim, that high trustworthiness and contractual collaboration agreements can allow more lenient security assumptions.

Q. Li et al. (2019) identify three major avenues of attacks: attacks on the data, the learning process and the final model. Different information and attack avenues are available at the client and server side. The client can see the global model, and manipulate its own data and model updates, including their importance. The server can see and manipulate the aggregated model, and potentially see the local updates, allowing differentiation (Enthoven and Al-Ars 2021). The aggregator as a privileged entity in the process is a prime target for attack, however according to Lyu et al. (2020), it is unclear whether peer-to-peer architectures mitigate the problem or open new avenues for attacks. Usynin et al. (2021) differentiate between attacks on utility (model poisoning, backdoor insertion and evasion) and attacks on privacy (membership inference, attribute inference, model inversion, model stealing).

In poisoning attacks, either the data or the model are modified such that the federated model systematically misclassifies a set of inputs (Yoo et al. 2021; Lyu et al. 2020). The goal can for instance be the creation of backdoors (secondary tasks), decreasing model accuracy, or run time misclassification to evade for example anomaly detection (Enthoven and Al-Ars 2021). Data

can be poisoned by flipping labels or introducing trigger patterns into the data. Models can also be poisoned by modifying gradients or other parameters (Sun et al. 2019; Bagdasaryan et al. 2018).

Inference attacks attempt to reconstruct class representatives, infer membership to the training data (i.e. whether a sample was part of the training or not) (Pustozerova and Mayer 2021; Nasr et al. 2019), or properties of the training data, such as training inputs and labels (Lyu et al. 2020; Enthoven and Al-Ars 2021). Data reconstruction attempts to infer precise data points. Depending on whether the attacker has knowledge of the model architecture or not, the attacks are classified as white-box or black-box attacks.

Attacks can be mitigated by the use of generic cryptographic tools, such as homomorphic encryption (HE), differential privacy (DP), and secure multiparty computation (SMPC), presented in the following. Other strategies suggested in the context of deep learning attempt for example to limit the information in the parameter updates, by compressing, regularizing, sub sampling or using robust aggregation of the gradients (Usynin et al. 2021). The term hybrid federated learning refers to a mixture of strategies where these techniques are combined to achieve a practical level of privacy. Generally, mitigation strategies assume a threat model which defines against which kind of adversary and which kind of attack the system gains resilience to by using the respective strategies. Enthoven and Al-Ars (2021) and Usynin et al. (2021) come to the conclusion that no individual strategy can mitigate all potential threats. Wainakh et al. (2021) also raise the issue that current attacks might follow unrealistic assumptions and might not be generically applicable yet.

1.4.2 Homomorphic Encryption

A homomorphic encryption scheme allows mathematical operations to be performed in the encrypted space, as if they were performed on the non-encrypted data. It has been suggested as a privacy preserving method for cloud computing where users can store their encrypted sensitive data, for example medical records, in the cloud. Computations and predictions are performed on the encrypted data (Lauter et al. 2011). In federated learning, HE could for instance be used for secure aggregation, however Kairouz et al. (2021) see open problems regarding the ownership of the keys.

The term fully homomorphic encryption (FHE) refers to a general purpose scheme, allowing all possible kinds of computation. Partially, or "somewhat" homomorphic encryption schemes allow a limited number of oper-

ations, including additions and multiplications. For many purposes this might be sufficient (Lauter et al. 2011). HE introduces a small amount of noise for each computation in the encrypted domain. Additions can be computed efficiently and require only a small amount of noise. Multiplications are expensive, and require more noise. Therefore, only a limited number of multiplications can be performed, before the noise becomes to large, in which case the decryption of the result becomes impossible. The number of possible multiplications is referred to as the multiplicative depth. Relinearization allows arbitrary depth circuits, however this procedure is expensive as well (Dobraunig et al. 2021).

In federated learning, homomorphic encryption could be used to mask the local updates from the server, if the aggregation step is performed under encryption and the aggregator does not have access to the private key. This is conceptually similar to the way one would perform secure aggregation using SMPC (see next section), with the advantage that the number of communication rounds would not increase. Currently, the high computational burden with high storage and time requirements even for small examples is an obstacle in the practical application of homomorphic encryption. The run times are still deemed unrealistic for the suggested applications, such as secure diagnostics and payroll computations (Pallas and Grambow 2018). Even state-of-the-art (Dobraunig et al. 2021) benchmarks using popular libraries are done on small examples and induce a high computational burden on client or server sites. With the potential size of the models in machine learning and the iterative, and interactive way of training, HE currently creates too large of a computational overhead making SMPC the more attractive choice for secure aggregation.

1.4.3 Secure Multiparty Computation

Secure Multiparty Computation (SMPC) is a group of techniques with the basic premise to evaluate a function without any of the participants gaining knowledge of the other parties inputs. Many of the techniques rely on secret sharing, where the values are split into independent shards which disclose nothing about the original value. A popular secret addition scheme relies on modular arithmetic. The N participants agree on a fixed prime p. They then mask their secret $s_n \in \mathbb{Z}_{\text{l}}$ by generating uniformly random numbers $r_{n,i}$ for $i \in [N-1]$ and computing the last random share as $r_{n,N} = s_n - \sum_{i=1}^{N-1} r_{n,i}$ mod p. The participants send each of th $N-1$ shares $r_{n,i}$ to the $N-1$ players and keep the Nth share themselves. The player sums up all the shares they receive, including the one they kept for themselves, as $r_i = \sum_{n=1}^{N} r_{n,i}$

and share r_i with all other players, which can now compute the output $o = \sum_i^N r_i = \sum_{n=1}^N s_n$, the sum over all s_n without gaining knowledge of the private inputs. This protocol is correct and secure, but not fault tolerant in the event that one of the parties drops out of the computation (Cramer et al. 2015). To gain fault tolerance, other schemes such as Shamir's secret sharing algorithm have been developed. Instead of creating random shares of the value, the participants now choose a secret polynomial f of degree k which evaluates to the secret value at $f(0)$. The secret shares are points on this polynomial. Using Lagrangian multipliers, the secret can be reconstructed using $k+1$ points of the polynomial. The advantage of this scheme is that if $k+1+k'$ shares are distributed, k' participants can fail and the secret can still be recovered. The schemes are designed for integer valued numbers. Float values can be processed using fixed-point arithmetic (Ryffel et al. 2018).

SMPC is not limited to addition. Much research has been done to allow the evaluation of arbitrarily complex circuits, for instance Yao's protocol (Snyder 2014). This allows the computation of entire pipelines, broadcasting only the final result of the computation, see for example Cho et al. (2018). Unfortunately, complex circuit evaluation implies a high computational burden with a high number of communication steps, and a considerable transmitted data volume. As a trade off, during the training of a model, simple aggregation such as addition, which can be computed efficiently using secure multiparty computation, can be used to protect the individual updates in each step, while allowing the broadcast of the global aggregated parameter updates in clear text.

The choice of a secure multiparty computation scheme depends on the assumptions on the adversary, the threat model. The aforementioned strategies work in a honest-but-curious setting, with all participants proceeding according to protocol. More advanced SMPC schemes are designed for systems assuming a dishonest majority, active attackers or detection of fraudulent behavior to protect honest participants Cramer et al. 2015.

Parameter obfuscation methods, such as HE and SMPC have the drawback that the clients' parameters cannot be inspected, so fraudulent updates, or even deleterious updates by honest users, are harder to detect. As the input data remains private, falsified or mislabeled data can be injected, a problem that can not be solved using secure function evaluation.

1.4.4 Differential Privacy

Differential privacy (DP) is a way of achieving privacy by perturbing the input, intermediate results or outputs of a computation. Informally speaking, DP is achieved if the results of a randomized algorithm applied on two very similar databases are almost indistinguishable. This means it is very hard to tell whether an individual participated in an analysis (was part of the database) or not. This has the nice property, that it does not only imply privacy but also statistical stability. Here, only the basic concepts of DP will be discussed. A more in-depth explanation of DP can be found in the monograph by Cynthia Dwork (Dwork and Roth 2014), a pioneer in the field of differential privacy.

In Dwork's work a view of a database is assumed where it is seen as a histogram of records where each bin represents a record of type R. This already implies, that the database contains many records of type R, meaning the data base should be large. In fact, *"Differential Privacy was designed with internet-scale data sets in mind."* (p.464, Dwork and Roth 2014).

Definition 1.4.1 (Differential Privacy)**.** *A mechanism* $\mathbf{M}$ *with output space* $\mathbf{O} \subset Range(\mathbf{M})$ *is* ϵ*-differentially private if for every set of adjacent databases* $\mathbf{D}, \mathbf{D'} : Pr[\mathbf{M}(\mathbf{D}) \in \mathbf{O}] = e^{\epsilon} * Pr[\mathbf{M}(\mathbf{D'}) \in \mathbf{O}]$

A way to achieve differential privacy is to add noise to the input data, during the computation, or to the result. The noise can be generated from different distributions, for instance the Laplace, the Exponential, or the Gaussian distribution. The scale of the noise is determined by the sensitivity of the function and the parameter ϵ which determines the privacy level. The sensitivity is a measure for how much the output of a function can change, when it is applied to similar data bases.

For example, in order to achieve ϵ-differential privacy using the Laplace mechanism, one adds noise drawn from the Laplace distribution with scale $b = \frac{\Delta f}{\epsilon}$, where Δf denotes the sensitivity.

$$Lap(x|b) = \frac{1}{2 * b} * exp(-\frac{|x|}{b}) \tag{1.1}$$

Hence, the smaller the sensitivity and the larger ϵ, the lower is the variance of the noise. Consequently, smaller ϵ imply better privacy.

There a more interesting properties of DP which make it a versatile cryptographic tool. First, a notion called "Closure under postprocessing". This

simply means, that any computations performed of the result of a differentially private analysis will remain differentially private. Second, the composition theorem, which states that two algorithms which are ϵ_1 and ϵ_2-DP can be composed to obtain an $(\epsilon_1 + \epsilon_2)$-DP algorithm.

Differential Privacy is a statistical tool to achieve provable guarantees for privacy. Closure under post-processing and composability allow us to propagate the privacy guarantees beyond the initial application of the algorithm. DP does not rely on encryption to achieve privacy, meaning the guarantees will persist even if an encryption scheme is eventually broken.

Although in theory elegant, DP has drawbacks. There is a limitation to the number of DP queries possible with an ϵ-DP mechanism, as with sufficient queries, the probability distributions of the outcomes of the randomized algorithm on two adjacent databases will become distinguishable (Bambauer et al. 2014). Apart from these theoretical considerations, the implementation of DP in practice requires computing the sensitivity of a function as well the choice of a suitable ϵ which may be hard for laymen to do. As the sensitivity of a function often depends on the dimensionality of the data, DP may be inapplicable to high dimensional data sets or the parameters choices might be to lenient in practice (Domingo-Ferrer et al. 2021).

1.4.5 Private data release

A partially orthogonal approach to federated learning is the the private publication of data. Early research on private data release lead to the conception of "k-anonymity" (Sweeney 1997) and "l-diversity" (Machanavajjhala et al. 2006). These techniques allow data release only, if a sufficient number of individuals have the same combination of a quasi-identifier and the sensitive attribute, and in the case of l-diversity a sufficient diversity in the sensitive attribute. However, these techniques are not sufficient to prevent privacy breaches (Rieke et al. 2020). A more recent technique is the generation of synthetic data. This type of data is generated based on real data and reflects its properties, but does not contain real samples. This data can for instance be generated using autoencoders (Abay et al. 2019) or generative adversarial networks (Beaulieu-Jones et al. 2019; Gootjes-Dreesbach et al. 2020). The concept is compatible with differential privacy, although strict privacy parameters lead to a decrease in performance and an increase of the cross-validation error (Hittmeir et al. 2019). Another similar technique is the generation of differentially private core-sets (Feldman et al. 2017) with the same goal of preserving statistical properties while allowing pri-

vate queries. The concept of synthetic data generation has been combined with federated learning explicitly in an attempt to overcome the problem that the decentralized data cannot be inspected and therefore bugs in the pipelines are hard to detect. By generating differentially private representatives of the input data which can be published safely to a server. Augenstein et al. (2019) directly combine the two domains. This approach could also be used for quality control and batch effect correction in bioinformatics pipelines.

1.4.6 Hybrid federated learning

A hybrid scheme combines FL with one or more privacy enhancing techniques to prevent privacy breaches (Torkzadehmahani et al. 2022). A hybrid scheme keeps the data in the silos, and additionally masks the parameters sent to the other participants or the aggregator. Additional trust in the aggregation server can be obtain through the use of a "Trusted Execution Environment" (TEE). This trusted hardware allows the verification that a computation is executed correctly, and has been suggested, although not implemented, for secure GWAS (X. Wu et al. 2021). Hybrid federated learning is the strategy of choice adopted in this thesis, which aims to reconcile data confidentiality and reasonable privacy guarantees with computational feasibility. For instance, local parameters are protected using secure multiparty computation, but intermediate aggregated parameters might become available in clear text.

1.5 The FeatureCloud Platform

FeatureCloud (Matschinske et al. 2021) is a platform for cross-silo federated learning in the biomedical domain. The main features of the system will be briefly discussed here, as they motivate the assumed federated learning setup in this thesis. The system is designed with hospitals and research institutions in mind. At the core of the platform is a workflow system, which allows the execution of one or several federated algorithms to perform a medical study. Algorithms are deployed as 'apps' which can be obtained from the FeatureCloud App store. Developers can contribute their own app, if no suitable app is available, making this system versatile for any kind of data mining or learning task.

The main components of the system are the "Controller", a software installed on all participants' IT infrastructure; a relay server, which can be

reached by all Controllers; and a global backend, required for the management of user data, authentication and the like. Users interact with the system via a browser frontend which communicates with the Controller and the global backend. The algorithms are deployed in virtualized containers which allows tight control over their access to data and resources, and in particular restrict direct internet access. Instead, the Controller polls the containers for updates and communicates with the relay server. Apart from managing the communication with the other participants, the Controller provides all workflow related functionalities such as the setup of the workflows, loading of the data, the creation of the app containers and shutdown after the termination of the workflow. The FeatureCloud platform is technically a star-like system, as all the Controllers communicate with the Relay server which receives and forwards the parameter updates to the intended recipients. However, through public-key cryptography, the peer-to-peer communication required for SMPC can be simulated. In FeatureCloud, the relay server does not perform the aggregation, instead one of the clients acts as the aggregator. The FeatureCloud platform comes with a Development Environment, the "Testbed", which allows the local simulation of multiple clients and speeds up the development process. To avoid malicious apps to be pushed into the app store, the FeatureCloud project envisions a certification process where the correctness, and functionality of the code, and the availability of the advertised privacy measures are certified. Most importantly the apps cannot send raw data and the source code must be available. The certified app is built and pushed to the app store by certificate issuer, not the developer.

1.6 Unsupervised Machine Learning

Unsupervised machine learning can be used on label free data and is a frequent part of bioinformatics pipelines. In this thesis, principal component analysis (PCA), the related singular value decomposition (SVD), QR decomposition, and the K-Means algorithm are investigated. These are classical data mining algorithms which have been studied for decades. However, the federated learning literature claims that there is a lack of research in unsupervised federated learning and classical data mining (Kairouz et al. 2021). This thesis aims to make a contribution towards closing this gap. The current section briefly introduced the centralized counterparts to the federated methods studied in later chapters, to which their detailed descriptions are deferred.

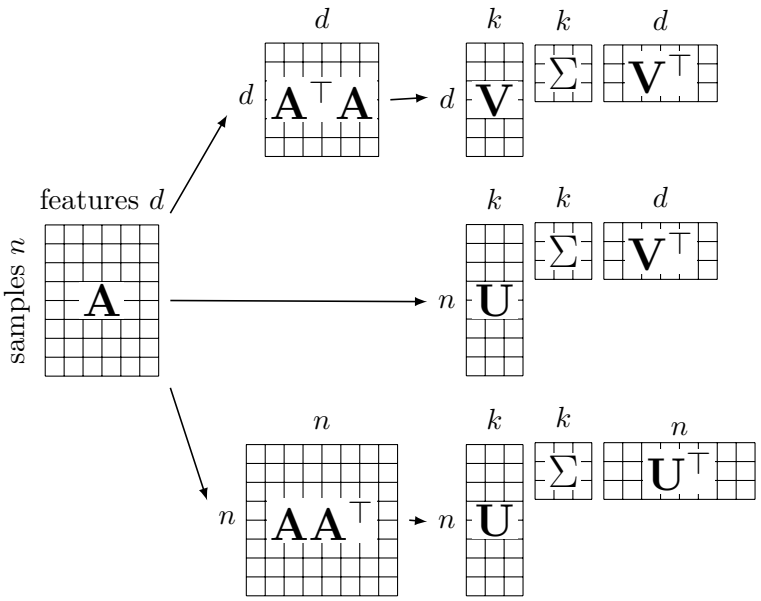


Figure 1.4 – Relationship between covariance matrices, PCA, and SVD.

1.6.1 Principal Component Analysis

Principal Component Analysis (PCA) (Jolliffe 2002) is a popular dimensionality reduction technique. It can summarize a high-dimensional data set to a few meta-variables explaining the main variability of the data along these axes. Conceptually, PCA is the Eigendecomposition of the covariance matrix of the centered and scaled data. Let $\mathbf{D} \in \mathbb{R}^{n \times d}$ be the original data matrix, where n is the number of samples and d is the number of features. The data is assumed to be drawn from a multivariate Gaussian distribution. Let $\bar{\mu}$ denote the column wise mean of $\mathbf{D}$ and σ denote the column-wise variance of $\mathbf{D}$. The data is then centered by subtracting the mean from each variable and dividing by the standard deviation.

$$\mathbf{A} = \frac{\mathbf{D} - \bar{\mu}}{\sqrt{\sigma}}. \tag{1.2}$$

Σ is a diagonal matrix containing the eigenvalues in non-increasing order, $\mathbf{V}$ denotes the matrix of corresponding eigenvectors. The PCA can then be computed as

$$\frac{1}{n-1}\mathbf{A}^\top \mathbf{A} = \mathbf{V}\Sigma\mathbf{V}^\top. \tag{1.3}$$

Singular value decomposition (SVD) is closely related to PCA and allows

the decomposition of the scaled data matrix $\mathbf{A}$ as follows:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top. \tag{1.4}$$

where $\mathbf{U}$ denotes the left singular vector, $\mathbf{V}$ denotes the right singular vector and Σ the diagonal matrix of singular values. The vectors $\mathbf{V}$ computed using singular value decomposition of $\mathbf{A}$ and computed via Eigendecomposition of $\mathbf{A}^\top\mathbf{A}$ are equivalent. Both PCA and SVD can be efficiently computed using for instance power iteration (Halko et al. 2011). The relationship between SVD and PCA is depicted in Figure 1.4.

In bioinformatics, PCA is for instance used to reduce the dimensionality of high-throughput sequencing data, followed by the embedding into a neighborhood graph and subsequent visualization via UMAP. In this case, the PCA is computed on the $d \times d$ feature-by-feature covariance matrix, and the data is projected onto the first k eigenvectors to form the principal components (Theis 2019). An orthogonal use of PCA is population stratification for instance used in Genome-Wide Association Studies. For GWAS, the PCA is computed on the sample-by-sample covariance matrix and the eigenvectors are included as covariates into the association test (Galinsky et al. 2016). A general challenge in both applications is the high dimensionality of the data, which requires efficient algorithms for routine application.

1.6.2 QR decomposition

QR decomposition, is the process of factorizing a matrix $\mathbf{A}$ into a square matrix $\mathbf{Q}$ with mutually orthonormal columns and an upper triangular matrix $\mathbf{R}$. In practice, the reduced QR decomposition which returns a rectangular $\mathbf{Q}$ is more memory efficient (Ford 2014). The three popular algorithms for QR decomposition use Householder reflections, Givens rotations or the Gram-Schmidt algorithm, although the latter is not recommended due to numerical instabilities (Ford 2014).

$$\mathbf{A} = \mathbf{QR} \tag{1.5}$$

In bioinformatics, QR decomposition is frequently used for the solution of systems of linear equations for example when performing linear regression. Linear regression in the programming language R for example, uses QR decomposition as the default solver (R Core Team 2021). In the context of power iteration, QR factorization can be used for the reorthogonalization of

the eigenvectors in power iteration. The frequent application of the method in centralized computation warrants an investigation of its federated counterpart.

1.6.3 K-Means Clustering

Clustering is the process of finding a partition of the data which is optimal with respect to some criterion. A popular and fast algorithm to perform this task is the K-Means algorithm which clusters the data into k clusters based for instance on the euclidean distance. This is done by iteratively minimizing the sum of the distances of the points to their respective cluster center arriving at a locally optimal clustering (MacQueen 1967; Lloyd 1982). The algorithm proceeds as follows: First, a set of k initial centroids $\mathcal{C}$ is chosen. All points are assigned to their nearest centoids. Then the centroids are updated by calculating the average over their assigned data points. Then, the convergence criterion is evaluated. If it is not fulfilled, the assignment and recomputation of new centroids is repeated. Once the termination criterion is fulfilled, the algorithm terminates. A suitable convergence criterion can for example be a fixed number of rounds, or convergence of the cluster centers. This algorithm is relatively cheap to compute, and conceptually simple. However, the performance of the algorithm crucially depends on the selection of good initial centroids (Fränti and Sieranoja 2019). Furthermore, K-Means requires the choice of a suitable k prior to the execution of the algorithm. This is inconvenient with novel data where the number of clusters is not known a-priori. In the centralized execution of the algorithm, these problems can be solved by repeated execution of K-Means. However, this may not be practical in federated clustering.

Federated clustering is the process of creating partitions (clusters) of a data set without having access to the entire data. It should not be confused with "clustered federated learning" (CFL) which has been suggested in the context of supervised FL. In CFL, the gradients are clustered, for instance to facilitate multitask learning (Sattler et al. 2019).

1.6.4 Unsupervised federated learning

Certain authors (Kairouz et al. 2021) claim that there is a lack of research in other areas than deep learning. While a comprehensive survey on unsupervised federated learning is outside of the scope of this work, it should be noted that the community has recognized the need for unsupervised learning algorithms in FL. In fact, research interest in decentralized data

analysis existed before the adoption of the term of "federated learning". Perhaps the use of different terminology and keywords popular prior to the emergence of the term "federated learning" make this literature less accessible. Table A.1 in Appendix A is non-exhaustive, yet shows an impressive collection of publications regarding federated principal component analysis over the years. Gaye et al. (2014) provide traditional tools such as histograms and generalized linear models in their platform. Zolotareva et al. (2020) provide federated differential testing. Several authors research federated GWAS (Cho et al. 2018; Nasirigerdeh et al. 2020; X. Wu et al. 2021). Our contributions are based on the excellent groundwork of previous researchers.

1.7 Summary & Aims of this thesis

This chapter introduced the relevant techniques in unsupervised learning and the important concepts and terminology, including cross-silo and cross-device FL, horizontal and vertical data partitioning, and possible architectural choices for FL systems. Then, the privacy of federated learning, as well as threats and mitigation strategies were discussed. The remainder of this work assumes a cross-silo federated learning setup. The default architecture is a star-like architecture, which allows peer-to-peer communication if required, motivated by the FeatureCloud platform.

The frequent application of PCA or SVD, QR decomposition and clustering in bioinformatics pipelines requires efficient versions of these methods for federated learning. In general, challenges that exist in the centralized computation are expected to remain relevant for the federated application of these methods. For instance, the computation of the covariance matrix in PCA should be avoided, as it requires a high availability of memory (see chapter 3 and 4). Section 1.3 and section 1.4 introduced novel challenges in federated learning, including communication efficiency, privacy and accuracy. Consequently, a federated algorithm should (a) require low resources at the clients and the aggregator, (b) not disclose private information about the input data, (c) be accurate, and (d) be communication efficient. The goal of this thesis is the identification, improvement and development of suitable federated algorithms for the aforementioned problems which fulfill these requirements.

References

Abay, N. C. et al. (2019). "Privacy Preserving Synthetic Data Release Using Deep Learning". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by M. Berlingerio et al. Cham: Springer International Publishing, pp. 510–526. ISBN: 978-3-030-10925-7.

Andriole, K. P. (2014). "Security of Electronic Medical Information and Patient Privacy : What You Need to Know". In: *Journal of the American College of Radiology* 11.12, pp. 1212–1216. ISSN: 1546-1440. DOI: 10.1016/j.jacr.2014.09.011. URL: http://dx.doi.org/10.1016/j.jacr.2014.09.011.

Augenstein, S. et al. (2019). "Generative Models for Effective ML on Private, Decentralized Datasets". In: *CoRR* abs/1911.06679. arXiv: 1911.06679. URL: http://arxiv.org/abs/1911.06679.

Bagdasaryan, E. et al. (2018). "How To Backdoor Federated Learning". In: 1. ISSN: 0007-1250. DOI: 10.1561/2200000016. arXiv: 1807.00459. URL: http://arxiv.org/abs/1807.00459.

Bambauer, J., Muralidhar, K., and Sarathy, R. (2014). "Foolś Gold : An Illustrated Critique of Differential Privacy". In:

Beaulieu-Jones, B. K. et al. (2019). "Privacy-preserving generative deep neural networks support clinical data sharing". In: *Circulation: Cardiovascular Quality and Outcomes* 12.7, pp. 1–10. ISSN: 19417705. DOI: 10.1161/CIRCOUTCOMES.118.005122.

Beyan, O. et al. (2020). "Distributed Analytics on Sensitive Medical Data: The Personal Health Train". In: *Data Intelligence* 2.1-2, pp. 96–107. DOI: 10.1162/dint_a_00032. URL: https://www.mitpressjournals.org/doi/abs/10.1162/dint_a_00032.

Bonawitz, K. et al. (2019). "Towards Federated Learning at Scale: System Design". In: ISSN: 2331-8422. arXiv: 1902.01046. URL: http://arxiv.org/abs/1902.01046.

Bonomi, L., Huang, Y., and Ohno-Machado, L. (2020). "Privacy challenges and research opportunities for genomic data sharing". In: *Nature Genetics* 52.7, pp. 646–654. ISSN: 1061-4036. DOI: 10.1038/s41588-020-0651-0. URL: http://www.nature.com/articles/s41588-020-0651-0.

Cho, H., Wu, D. J., and Berger, B. (2018). "Secure genome-wide association analysis using multiparty computation". In: *Nature Biotechnology* 36.6, pp. 547–551. ISSN: 15461696. DOI: 10.1038/nbt.4108. URL: http://www.ncbi.nlm.nih.gov/pubmed/29734293%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5990440.

Cramer, R., Damgård, I. B., and Nielsen, J. B. (2015). *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press. DOI: 10.1017/CBO9781107337756.

Dobraunig, C. et al. (2021). "Pasta: A Case for Hybrid Homomorphic Encryption; Pasta: A Case for Hybrid Homomorphic Encryption". In: pp. 1–42. URL:

https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:32016R0679.

Domingo-Ferrer, J., Sánchez, D., and Blanco-Justicia, A. (2021). "The limits of differential privacy (and its misuse in data release and machine learning)". In: *Communications of the ACM* 64.7, pp. 33–35. ISSN: 0001-0782. DOI: 10.1145/3433638. arXiv: 2011.02352.

Drysdale, R. et al. (2020). "The ELIXIR Core Data Resources : fundamental infrastructure for the life sciences". In: 36.January, pp. 2636–2642. DOI: 10.1093/bioinformatics/btz959.

Dwork, C. and Roth, A. (2014). "The Algorithmic Foundations of Differential Privacy". In: 9.2013, pp. 211–407. DOI: 10.1561/0400000042.

Enthoven, D. and Al-Ars, Z. (2021). "An Overview of Federated Deep Learning Privacy Attacks and Defensive Strategies". In: *Federated Learning Systems: Towards Next-Generation AI*. Ed. by M. H. u. Rehman and M. M. Gaber. Cham: Springer International Publishing, pp. 173–196. ISBN: 978-3-030-70604-3. DOI: 10.1007/978-3-030-70604-3_8. URL: https://doi.org/10.1007/978-3-030-70604-3_8.

Feldman, D. et al. (2017). "Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks". In: *Proceedings - 2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2017*, pp. 3–15. DOI: 10.1145/3055031.3055090.

Fereidooni, H. et al. (2017). "Fitness Trackers: Fit for Health but Unfit for Security and Privacy". In: *Proceedings - 2017 IEEE 2nd International Conference on Connected Health: Applications, Systems and Engineering Technologies, CHASE 2017*, pp. 19–24. DOI: 10.1109/CHASE.2017.54.

Feyissa, G. T. et al. (2019). "Reducing HIV-related stigma and discrimination in healthcare settings: A systematic review of quantitative evidence". In: *PLoS ONE* 14.1, pp. 1–23. ISSN: 19326203. DOI: 10.1371/journal.pone.0211298.

Ford, W. (2014). *Numerical Linear Algebra with Applications : Using MATLAB*. Elsevier Science & Technology.

Fränti, P. and Sieranoja, S. (2019). "How much can k-means be improved by using better initialization and repeats?" In: *Pattern Recognition* 93, pp. 95–112. ISSN: 00313203. DOI: 10.1016/j.patcog.2019.04.014.

Galinsky, K. J. et al. (2016). "Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia". In: *The American Journal of Human Genetics* 98.3, pp. 456–472. ISSN: 00029297. DOI: 10.1016/j.ajhg.2015.12.022. URL: http://dx.doi.org/10.1016/j.ajhg.2015.12.022%20https://linkinghub.elsevier.com/retrieve/pii/S0002929716000033.

Gaye, A. et al. (2014). "DataSHIELD: Taking the analysis to the data, not the data to the analysis". In: *International Journal of Epidemiology* 43.6, pp. 1929–1944. ISSN: 14643685. DOI: 10.1093/ije/dyu188.

Gootjes-Dreesbach, L. et al. (2020). "Variational Autoencoder Modular Bayesian Networks for Simulation of Heterogeneous Clinical Study Data". In: *Frontiers in Big Data* 3.May, pp. 1–15. ISSN: 2624909X. DOI: 10.3389/fdata.2020.00016.

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions". In: pp. 1–74. arXiv: arXiv:0909.4061v2.

Hard, A. et al. (2018). "Federated Learning for Mobile Keyboard Prediction". In: arXiv: 1811.03604. URL: http://arxiv.org/abs/1811.03604.

Hastie, T. T. (2017). "The Elements of Statistical Learning Second Edition". In: *Math. Intell.* 27.2, pp. 83–85. ISSN: 03436993. DOI: 111. eprint: arXiv:1011.1669v3.

Hittmeir, M., Ekelhart, A., and Mayer, R. (2019). "On the utility of synthetic data: An empirical evaluation on machine learning tasks". In: *Pervasive-Health: Pervasive Computing Technologies for Healthcare*. ISSN: 21531633. DOI: 10.1145/3339252.3339281.

Huang, L. et al. (2019). "Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records". In: *Journal of Biomedical Informatics* 99.September, p. 103291. ISSN: 15320464. DOI: 10.1016/j.jbi.2019.103291. URL: https://doi.org/10.1016/j.jbi.2019.103291.

Jolliffe, I. (2002). *Principal Component Analysis*. Springer-Verlag. DOI: 10.1007/b98835. URL: https://doi.org/10.1007/b98835.

Kairouz, P. et al. (2021). "Advances and open problems in federated learning". In: *Foundations and Trends in Machine Learning* 14.1-2, pp. 1–210. ISSN: 19358245. DOI: 10.1561/2200000083. arXiv: 1912.04977.

Karim, M. R. et al. (2021). "Deep learning-based clustering approaches for bioinformatics". In: *Briefings in Bioinformatics* 22.1, pp. 393–415. ISSN: 14774054. DOI: 10.1093/bib/bbz170.

Kerr, K. et al. (2020). "A scoping review and proposed workflow for multi-omic rare disease research". In: *Orphanet Journal of Rare Diseases* 15.1, pp. 1–18. ISSN: 17501172. DOI: 10.1186/s13023-020-01376-x.

Lauter, K., Naehrig, M., and Vaikuntanathan, V. (2011). "Can homomorphic encryption be practical?" In: *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 113–124. ISSN: 15437221. DOI: 10.1145/2046660.2046682.

Li, F. et al. (2021). "Positive-unlabeled learning in bioinformatics and computational biology: a brief review". In: *Briefings in Bioinformatics* 23.September 2021, pp. 1–13. ISSN: 1467-5463. DOI: 10.1093/bib/bbab461.

Li, Q. et al. (2019). "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection". In: *CoRR* abs/1907.09693. arXiv: 1907.09693. URL: http://arxiv.org/abs/1907.09693.

Liu, Y., Fan, T., et al. (2021). "FATE: An industrial grade platform for collaborative learning with data protection". In: *Journal of Machine Learning Research* 22, pp. 1–6. ISSN: 15337928.

Liu, Y., Ma, Z., et al. (2020). "Privacy-preserving fe derate d k -means for proactive caching in next generation cellular networks". In: *Information Sciences* 521, pp. 14–31. DOI: 10.1016/j.ins.2020.02.042. URL: https://doi.org/10.1016/j.ins.2020.02.042.

Lloyd, S. P. (1982). "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. ISSN: 15579654. DOI: 10.1109/TIT.1982.1056489.

Lopez, R. et al. (2018). "Deep generative modeling for single-cell transcriptomics". In: *Nature Methods* 15.12, pp. 1053–1058. ISSN: 15487105. DOI: 10.1038/s41592-018-0229-2.

Lyu, L. et al. (2020). "Threats to Federated Learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12500 LNCS, pp. 3–16. ISSN: 16113349. DOI: 10.1007/978-3-030-63076-8_1. arXiv: arXiv:2003.02133v1.

Machanavajjhala, A. et al. (2006). "L-diversity: privacy beyond k-anonymity". In: *22nd International Conference on Data Engineering (ICDE'06)*, pp. 24–24. DOI: 10.1109/ICDE.2006.1.

MacQueen, J. (1967). "SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS". In: 233.233, pp. 281–297.

Marfoq, O. et al. (2020). "Throughput-optimal topology design for cross-silo federated learning". In: *Advances in Neural Information Processing Systems* 2020-December.NeurIPS. ISSN: 10495258. arXiv: 2010.12229.

Matschinske, J. et al. (2021). "The FeatureCloud AI Store for Federated Learning in Biomedicine and Beyond". In: pp. 1–32.

Mcinnes, L., Healy, J., and Melville, J. (2018). "UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction arXiv : 1802 . 03426v2 [stat . ML] 6 Dec 2018". In: arXiv: arXiv:1802.03426v2.

McMahan, H. B. et al. (2017). "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: 54, p. 10.

Mills, M. C. and Rahal, C. (2019). "A scientometric review of genome-wide association studies". In: *Communications Biology* 2.1. ISSN: 23993642. DOI: 10.1038/s42003-018-0261-x. URL: http://dx.doi.org/10.1038/s42003-018-0261-x.

Nasirigerdeh, R. et al. (2020). "sPLINK: A Federated, Privacy-Preserving Tool as a Robust Alternative to Meta-Analysis in Genome-Wide Association Studies". In: 6.Figure 1, pp. 1–16. DOI: 10.1101/2020.06.05.136382.

Nasr, M., Shokri, R., and Houmansadr, A. (2019). "Comprehensive Privacy Analysis of Deep Learning". In: *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 739–753. ISSN: 10816011. arXiv: arXiv:1812.00910v1.

Nguyen, D. C. et al. (2021). "Federated Learning for Smart Healthcare: A Survey". In: *CoRR* abs/2111.08834. arXiv: 2111.08834. URL: https://arxiv.org/abs/2111.08834.

Paleyes, A., Urma, R.-G., and Lawrence, N. D. (2020). "Challenges in Deploying Machine Learning: a Survey of Case Studies". In: pp. 1–21. arXiv: 2011.09926.

Pallas, F. and Grambow, M. (2018). "Three Tales of Disillusion: Benchmarking Property Preserving Encryption Schemes". In: *Trust, Privacy and Security in Digital Business*. Ed. by S. Furnell, H. Mouratidis, and G. Pernul. Cham: Springer International Publishing, pp. 39–54. ISBN: 978-3-319-98385-1.

Passerat-Palmbach, J. et al. (2019). "A blockchain-orchestrated Federated Learning architecture for healthcare consortia". In: arXiv: 1910.12603. URL: http://arxiv.org/abs/1910.12603.

Pustozerova, A. and Mayer, R. (2021). "Information Leaks in Federated Learning". In: February, pp. 1–6. DOI: 10.14722/diss.2020.23004.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Rees, C. A. et al. (2019). "Noncompletion and nonpublication of trials studying rare diseases: A cross-sectional analysis". In: *PLoS Medicine* 16.11, pp. 1–16. ISSN: 15491676. DOI: 10.1371/journal.pmed.1002966.

Rieke, N. et al. (2020). "The future of digital health with federated learning". In: *npj Digital Medicine*, pp. 1–7. ISSN: 2398-6352. DOI: 10.1038/s41746-020-00323-1. URL: http://dx.doi.org/10.1038/s41746-020-00323-1.

Roy, A. G. et al. (2021). "BrainTorrent : A Peer-to-Peer Environment for Decentralized Federated Learning". In: pp. 1–9. arXiv: arXiv:1905.06731v1.

Ryffel, T. et al. (2018). "A generic framework for privacy preserving deep learning". In: arXiv: 1811.04017. URL: http://arxiv.org/abs/1811.04017.

Sattler, F., Möller, K. R., and Samek, W. (2019). "Clustered federated learning: Model-Agnostic distributed multi-Task optimization under privacy constraints". In: *arXiv*, pp. 1–16. ISSN: 23318422. DOI: 10.1109/tnnls.2020.3015958. arXiv: 1910.01991.

Sheller, M. J., Edwards, B., et al. (2020). "Federated learning in medicine : facilitating multi - institutional collaborations without sharing patient data". In: *Scientific Reports*, pp. 1–12. ISSN: 2045-2322. DOI: 10.1038/s41598-020-69250-1. URL: https://doi.org/10.1038/s41598-020-69250-1.

Sheller, M. J., Reina, G. A., et al. (2019). "Multi-institutional Deep Learning Modeling Without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Ed. by A. Crimi et al. Cham: Springer International Publishing, pp. 92–104. ISBN: 978-3-030-11723-8.

Sirugo, G., Williams, S. M., and Tishkoff, S. A. (2019). "The Missing Diversity in Human Genetic Studies". In: *Cell* 177.1, pp. 26–31. ISSN: 10974172. DOI:

10.1016/j.cell.2019.02.048. URL: https://doi.org/10.1016/j.cell.2019.02.048.

Sluciak, O. et al. (2012). "Distributed Gram-Schmidt orthogonalization based on dynamic consensus". In: *Conference Record - Asilomar Conference on Signals, Systems and Computers*, pp. 1207–1211. ISSN: 10586393. DOI: 10.1109/ACSSC.2012.6489213.

Snyder, P. (2014). "Yao ' s Garbled Circuits : Recent Directions and Implementations". In:

Sun, Z. et al. (2019). "Can You Really Backdoor Federated Learning?" In: ISSN: 2331-8422. arXiv: 1911.07963. URL: http://arxiv.org/abs/1911.07963.

Sweeney, L. (1997). "Weaving Technology and Policy Together to Maintain Confidentiality". In: *Journal of Law, Medicine and Ethics* 25.2-3, pp. 98–110. ISSN: 10731105. DOI: 10.1111/j.1748-720X.1997.tb01885.x.

Tan, A. Z. et al. (2021). "Towards Personalized Federated Learning". In: arXiv: 2103.00710. URL: http://arxiv.org/abs/2103.00710.

Theis, F. J. (2019). "Current best practices in single-cell RNA-seq analysis : a tutorial". In: *Mol Syst Biol*. DOI: 10.15252/msb.20188746.

Torkzadehmahani, R. et al. (2022). "Privacy-Preserving Artificial Intelligence Techniques in Biomedicine". In: *Methods of Information in Medicine*. ISSN: 0026-1270. DOI: 10.1055/s-0041-1740630. arXiv: 2007.11621. URL: http://arxiv.org/abs/2007.11621%20http://dx.doi.org/10.1055/s-0041-1740630%20http://www.thieme-connect.de/DOI/DOI?10.1055/s-0041-1740630.

Traag, V. A., Waltman, L., and Eck, N. J. van (2019). "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific Reports* 9.1, pp. 1–12. ISSN: 20452322. DOI: 10.1038/s41598-019-41695-z. arXiv: 1810.08473.

ukbiobank.ac.uk (2022). *About us — ukbiobank.ac.uk*. https://www.ukbiobank.ac.uk/learn-more-about-uk-biobank/about-us. [Accessed 04-01-2022].

Usynin, D. et al. (2021). "Adversarial interference and its mitigations in privacy-preserving collaborative machine learning". In: *Nature Machine Intelligence* 3.9, pp. 749–758. ISSN: 25225839. DOI: 10.1038/s42256-021-00390-3. URL: http://dx.doi.org/10.1038/s42256-021-00390-3.

Veen, E. B. van (2018). "Observational health research in Europe: understanding the General Data Protection Regulation and underlying debate". In: *European Journal of Cancer* 104, pp. 70–80. ISSN: 18790852. DOI: 10.1016/j.ejca.2018.09.032. URL: https://doi.org/10.1016/j.ejca.2018.09.032.

Wainakh, A. et al. (2021). "Federated Learning Attacks Revisited: A Critical Discussion of Gaps, Assumptions, and Evaluation Setups". In: arXiv: 2111.03363. URL: http://arxiv.org/abs/2111.03363.

Warnat-Herresthal, S. et al. (2021). "Swarm Learning for decentralized and confidential clinical machine learning". In: *Nature* 594.7862, pp. 265–270. ISSN: 14764687. DOI: 10.1038/s41586-021-03583-3.

Wu, S. X. et al. (2018). "A Review of Distributed Algorithms for Principal Component Analysis". In: *Proceedings of the IEEE* 106.8, pp. 1321–1340. ISSN: 00189219. DOI: 10.1109/JPROC.2018.2846568.

Wu, X. et al. (2021). "A novel privacy-preserving federated genome-wide association study framework and its application in identifying potential risk variants in ankylosing spondylitis". In: *Briefings in Bioinformatics* 22.3, pp. 1–10. ISSN: 14774054. DOI: 10.1093/bib/bbaa090.

Xu, J. et al. (2021). "Federated Learning for Healthcare Informatics". In: *Journal of Healthcare Informatics Research* 5.1, pp. 1–19. ISSN: 2509498X. DOI: 10.1007/s41666-020-00082-4. arXiv: 1911.06270.

Yoo, J. H. et al. (2021). "Federated Learning: Issues in Medical Application". In: pp. 3–22. ISSN: 16113349. DOI: 10.1007/978-3-030-91387-8_1. arXiv: 2109.00202.

Zolotareva, O. et al. (2020). "Flimma: a federated and privacy-preserving tool for differential gene expression analysis". In: arXiv: 2010.16403. URL: http://arxiv.org/abs/2010.16403.

Chapter 2

Manuscript 1

Federated Horizontally Partitioned Principal Component Analysis for Biomedical Applications

Journal Title Here, 2019, 1–12
doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

Federated Horizontally Partitioned Principal Component Analysis for Biomedical Applications

Anne Hartebrodt[1,*] and Richard Röttger[1]

[1]Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230 Odense, Denmark

*Corresponding author. hartebrodt@imada.sdu.dk

FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Motivation: Federated learning enables privacy preserving machine learning in the medical domain because the sensitive patient data remains with the owner and only parameters are exchanged between the data holders. The federated scenario introduces specific challenges related to the decentralized nature of the data, such as batch effects and differences in study population between the sites. Here, we investigate the challenges of moving classical analysis methods to the federated domain, specifically principal component analysis, a versatile and widely used tool, often serving as an initial step in machine learning and visualization workflows. We provide implementations of different federated PCA algorithms and evaluate them regarding their accuracy for high dimensional biological data using realistic sample distributions over multiple data sites, and their ability to preserve downstream analyses.

Results: Federated subspace iteration converges to the centralized solution even for unfavorable data distributions, while approximate methods introduce error. Larger sample sizes at the study sites lead to better accuracy of the approximate methods. Approximate methods may be sufficient for coarse data visualization, but is vulnerable to outliers and batch effects. Before the analysis, the version of PCA, as well as the number of eigenvectors should be considered carefully to avoid unnecessary communication overhead.

Availability: Simulation code and notebooks for federated PCA can be found at `https://gitlab.com/roettgerlab/federatedPCA`; the code for the federated app is available at `https://github.com/AnneHartebrodt/fc-federated-pca`

Contact: hartebrodt@imada.sdu.dk

Supplementary information: Supplementary data are available at *Bioinformatics Advances* online.

Key words: PCA, horizontal data partitioning, federated machine learning, unsupervised learning

Introduction

Federated learning (FL) has recently gained attention in the machine learning (ML) community as a privacy preserving alternative to centralized computation. Contrary to classical machine learning, where the data is consolidated into a single machine or cloud, the data stays with the owner during the entire learning process and only model parameters are exchanged between the participants. The concept has potential applications in domains where the volume of data is too large to be stored at a single location, and in domains where the owners have concerns about losing agency over their data or are not allowed to share their data. This is especially important in the medical domain, where, due to patient confidentiality, doctors and hospitals are rightfully unable or unwilling to share their data with a third party. Outside of academic applications or in hybrid settings, federated learning can enable (industry) partners who are unwilling to disclose their raw data, but willing to join an analysis, to contribute to studies. These scenarios are cases of cross-silo federated learning where larger chunks of data are stored in 'data silos'. Another type of federated learning is cross-device federated learning popular in particular for mobile applications, where each participant has only access to their own data (for example on their phone). Another, complementary approach for private data analysis currently discussed is the generation of synthetic data with the same properties as the raw data. This approach is a valid option if sufficiently trustworthy generators can be created. The advantage of FL is that it is a

generic approach while synthetic data could suffer the biases of the training data and crucial, subtle information can potentially be lost in the generation process.

In the last decade, high throughput techniques have been routinely used to generate vast amounts of biomedical data [25]. Nevertheless, to this day, studies are commonly reporting a lack of data as a main limitation of their study, resulting in insufficiently validated, and potentially confounded, or unstable predictors [21]. An investigation of the causes for trial termination in rare diseases showed that 30% of non-completed clinical trials were terminated due to insufficient patient accrual [31]. Furthermore, many diagnostic tools are biased towards the predominant demographic at the site of the study, leading to potentially inapplicable results in other demographic groups posing an ethical problem [41]. A very prominent example of this bias are Genome-Wide Association studies which suffer from a massive bias towards population of European ancestry [34] and very small cohorts otherwise. This problem arises because the data generated in a research facility or hospital may only leave this institution under restricted conditions [37]. To overcome this challenge, federated learning has been brought forward as a solution to work with sensitive medical data without breaching patient privacy.

A popular method for the analysis of biomedical data is principal component analysis (PCA). It is a dimensionality reduction technique frequently used for count data, including bulk and single cell transcriptome data [36]. Several algorithms have been proposed for PCA in a federated setting. However, these algorithms were mainly evaluated using 'standard' test data sets and rarely with biomedical applications in mind. For instance, where the popular `MNIST` [22] data set comprises 60,000 samples with 784 pixels (dimensions), bulk transcriptome data usually only has a few hundred samples but measurements for about 20000 coding genes. The dimensionalities of these data sets are fundamentally different with *large n, moderate d* in the classical case and *small n, large d* in the biomedical scenario.

To enable the routine use of federated PCA in biomedicine, the existing algorithms must be evaluated for their suitability to analyze medical data. A major concern is the accuracy of the methods, given that the outcome of the studies will be included into medical decision making. Here, we will investigate the suitability of various approaches to federated PCA with varying, but realistic sample distributions using data from The Cancer Genome Atlas (TCGA). TCGA is a large scale project with multiple participating research centers [40] which profiled various cancer types using different *OMICS* technologies, including genome and transcriptome sequencing. According to a survey on multi-omics studies on rare diseases many of the cancers included indeed count as a rare disease [20]. The decentralized nature of the TCGA sampling process makes it suitable to study the feasibility of federated learning using real medical data: At the tissue source site (TSS) a sample was collected, and the RNA was extracted. The processed sample was then shipped to a central sequencing center, sequenced, and processed according to a standardized protocol. In this setup, the sequencing was done at a central facility, which is a slight deviation from a truly federated sample acquisition process where every TSS would perform the sequencing itself. However, it is a realistic example with respect to the number and distribution of participants, potential batch effects through different population demographics per sample site, and batch effects due to the sample preparation as it would occur in a truly federated analysis. Additionally, we use simulated single cell data to illustrate the problem in a practical fashion. Single cell studies inherently contain more 'samples', as each individual cell constitutes an observation. While this might alleviate the issue of limited sample availability, other problems such as batch effects in the data remain.

Lastly, many methods exist only on paper, making it hard to judge the practicalities of the suggested approaches. Computational biology is a notoriously heterogeneous research field with many practitioners not having a profound background in computer science and programming. Here, we provide simulation code and a federated app which allows users to run federated PCA. Based on our considerations, we support the users with our derived guidelines to choose the appropriate algorithm for their purpose.

Overall, our contributions are:

- A comprehensive overview of different federated PCA algorithms, including simulation code and a federated app.
- Performance comparisons of those approaches including communication overhead and different accuracy metrics.
- Evaluation of the algorithms using realistic data partitions derived from the sample distributions from TCGA.
- A practical illustration of the application of federated PCA using single-cell data, in particular to highlight the feasibility and potential problems of federated biomedical studies.
- Guidelines for the selection of the best algorithmic approach.

The remainder of this article is structured as follows. In section 2 we discuss the relevant data, algorithms and test setup. In section 3.1 we describe our theoretical and practical findings. Section 4 puts the results into perspective and provides guidelines for the interested reader, and section 5 concludes the work.

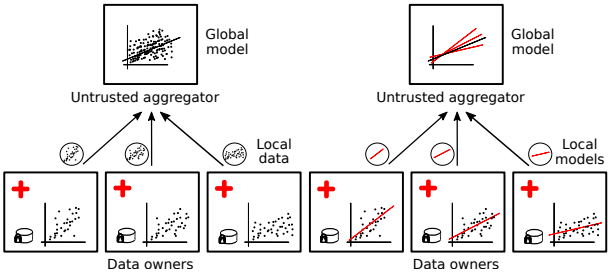


Fig. 1: Comparison of cloud learning and federated learning. In cloud learning (left panel) the data is consolidated at a central server which computes the global model. In federated learning (right panel), the different sites (e.g., hospitals) calculate a local model on their private data and send only the model parameters to an untrusted aggregator. The global model is computed and can be sent back to the local sites.

Material and Methods

Distributed data model

The distributed setting for the remainder of the paper is as follows: The data $\mathbf{A}$ is stored in s distinct subsets $\mathbf{A} = \mathbf{A}_1 \cup \cdots \cup \mathbf{A}_s$ at s different sites (e.g., hospitals) and constitutes a total of n patients with d dimensions. Rows correspond to patients, columns correspond to variables. Every site has a different subset of n_s patients but the full set of observed variables. Using terminology established by [32] and [43], we speak of distributed rows or horizontal partitioning of the data. Due to privacy constraints the sites are only allowed to exchange aggregated parameters. We are assuming a client-server/star-like architecture [35, 17], where sites communicate with a central server which performs the aggregation step. Peer-to-peer architectures, such as proposed in the concept of swarm learning and the personal health train [39, 6] could be used at the cost of additional communication steps and conceptually more involved protocols. The data sets at the distant sites will be called *local data sets* and the parameters or models learned using this data will be called *local parameters* or *local models*, while the final aggregated model will be called *global model* and considered optimal when it equals the result of the conventional model, the *centralized model*, calculated on all data.

High dimensional biomedical data

One common property of *OMICS* data is that it typically contains significantly more features than samples are available, i.e, $d >> n$.The dimensionality of *OMICS* data can be quite unfavorable, with a high number of features compared to the number of available samples ($d > n$). Although the trend towards more granular (e.g. single cell) analyses alleviates this problem, and PCA is a method applied to all types of data, there are applications where the number of samples remains low. Therefore, it is interesting to evaluate, how well federated methods perform on data with a high number of features compared to the number of samples. This setting is rarely considered in typical test scenarios for new algorithms, where usually the sample size of the test data exceeds the number of dimensions. For this study we selected all publicly available gene expression studies on TCGA in form of the processed count tables downloaded from the web repository (`https://www.cancer.gov/tcga`). These contain FPKM normalized counts, according to the unified TCGA pipeline. We scaled and normalized the data to unit variance, but did no further processing. We chose to divide the data according to the cancer type annotated in TCGA. We narrowed down the data selection to studies containing more than 300 individuals. We split the data into subsets according to the TSS. The sample distribution over different sites varies greatly between the studies. Most of the studies have skewed sample distributions with one site contributing considerably more samples than others. Please refer to supplementary figure 1 and table 1 for an overview of the number of samples and the number of TSS per study after filtering. We want to emphasize that this setup is distinctively different than the usual test setup of federated algorithms, where large data sets are split into a few equally sized chunks with iid data distribution with respect to the classes.

Table 1. Summary of number of samples and number of sites per cancer type.

Data set	No. Samples	No. Sites
Kidney	887	24
Thyroid gland	504	11
Liver and intrahepatic bile ducts	404	8
Bladder	408	14
Ovary	377	9
Brain	679	20
Prostate gland	495	14
Corpus uteri	547	12
Breast	1093	19
Cervix uteri	304	8
Colon	458	12
Bronchus and lung	1017	34
Stomach	386	9
Skin	468	11

Principal Component Analysis

Principal component analysis is used to calculate a low dimensional approximation of the data [19]. Intuitively, the data is projected into a lower dimensional representation using the directions which maximise the variance. Let the global data be given as a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$. The PCA is the decomposition of the covariance matrix $\mathbf{M} = \frac{1}{n}\mathbf{A}^\top\mathbf{A}$ into $\mathbf{M} = \mathbf{V}\Sigma\mathbf{V}^\top$. Σ is a diagonal matrix containing the non-negative eigenvalues σ_i in non-increasing order. $\mathbf{V}$ is a matrix containing the eigenvectors $\mathbf{v}_i$ corresponding to the eigenvalues σ_i with $\mathbf{v}_i$ column vectors. Note: As we are solving the eigendecomposition of a $n \times d$ matrix where $d >> n$ the maximal number of non-zero eigenvalues is $n - 1$. The top k- eigenvalues and corresponding eigenvectors are called a k-subspace and denoted as $(\mathbf{V}^k, \Sigma^k)$.

Federated Principal Component Analysis for horizontally partitioned data

In the federated case the data is distributed over s sites with n_s samples, such that $n = \sum_{i=1}^{s} n_i$. The goal of distributed principal component analysis is to find an eigendecomposition of $\mathbf{A}$ without having all the local data sets $\mathbf{A}_s$ at a central site. The data is assumed to be centered, and if applicable scaled to unit variance which can be achieved easily using federated summary statistics. There are broadly two groups of algorithms, single round approaches which communicate only once between the clients and the aggregator and iterative approaches which require multiple communication rounds. The single-round approaches follow the same main idea, but have different implementation details. Generally, a local summary statistic is computed and sent to the central aggregator, where it is merged to a global model.

Reconstitution of the covariance matrix

The reconstitution of the global (approximated) covariance matrix is a popular approach which has been implemented in different variations. They all rely on the observation that the covariance matrix can be computed exactly at the global server by adding up the local covariance matrices. This basic version is, for instance, used in [23]. In this version, the covariance matrices of the local data sets are computed and sent to the aggregator.

At the aggregator the local covariance matrices are summed up element-wise. The Eigendecomposition of this exact covariance matrix is computed and shared with the clients. We denote this version `P-COV`. A means to significantly reduce the transmission costs is to approximate the local subspaces and send these to the aggregator. In this case, a local SVD is computed and the top-k eigenspace is sent to the aggregator, where k is fixed but arbitrary [29, 38, 17, 42, 1, 23, 10]. More precisely, in these algorithms the local subspace $(\mathbf{V}_s^k, \Sigma_s^k)$ is computed at each site and sent to the aggregator (lines 2 to 3) . At the aggregator, a proxy covariance matrix $\mathbf{M}_s^p = \mathbf{V}_s^k \Sigma_s^k \mathbf{V}^\top$ for each site is reconstituted using $(\mathbf{V}_s^k, \Sigma_s^k)$, and added up element-wise such that an approximation of the global covariance matrix $\mathbf{M}^p$ is obtained (line 8). As only a limited number k of eigenvectors is transmitted $\mathbf{M}^p$ is an approximation of the hypothetical global covariance matrix. The global PCA is then computed by the eigendecomposition of the proxy covariance matrix (line 12). This version is denoted `AP-COV`.

Subspace aggregation The federated PCA algorithm proposed by Balcan et al. [4] computes a local subspace such as in the proxy covariance methods above but differs in the aggregation step. The local subspaces $(\mathbf{V}^k, \Sigma^k)$ are concatenated on the vertical axis and the singular value decomposition of this stacked subspace is computed (line 10). It is conceptually the same as computing the proxy covariance matrix but more efficient depending on the dimensions of the input matrices. This version is called `AP-STACK` in the remainder of the article. Please refer to algorithm 1 for a pseudocode description of these algorithms. (They have been merged due to their conceptual overlap).

Intermediate dimensionality `AP-COV` and `AP-STACK` have a parameter k' which determines the number of eigenvectors transferred to the aggregator. This number of intermediate dimensions k' is usually larger than the target dimensions k, the size of the final subspace. Naturally, k' determines the transmission cost of these approaches. This already hints at an issue regarding the dimensionality of the local subspaces: The number of retrieved eigenvectors is limited by the minimum dimension of the data matrix (i.e., either by the number of features, or by the number of samples) at the site. For instance, the subspace of a $20 \times 20,000$ matrix can only have 20 eigenvectors which means higher order global subspaces, that is subspaces where the global k is set to be larger than any of the local k can possibly be, can suffer accuracy loss w. r. t. to the centralized solution.

Algorithm 1 Federated PCA using subspace aggregation

Require: Data matrices $\mathbf{A}_s \in \mathbb{R}^{n_s \times m}$, # eigenvectors k.
1: **Client**
2: $\mathbf{U}_s, \Sigma_s, \mathbf{V}_s^\top \leftarrow$ `svd`$(\mathbf{A}_s)$
3: `send-to-aggregator`$(\mathbf{V}_s^{k\top}, \Sigma_s^k)$
4: **Client**
5: **Aggregator**
6: $[\mathbf{V}_s^{k\top}, \Sigma_s^k] \leftarrow$ **for** $s \in [S]$ `get-from-client`$(\mathbf{V}_s^{k\top}, \Sigma_s^k)$
7: **if** (`P-COV`, `AP-COV`) **then** ▷ Proxy cov. methods
8: $\mathbf{M} \leftarrow \sum_s^S \mathbf{V}_s^k \Sigma_s^k \mathbf{V}_s^{k\top}$
9: **else** ▷ Balcan et al. (`AP-STACK`)
10: $\mathbf{M} \leftarrow$ `stack-vertically`$([\Sigma_s^k \mathbf{V}_s^{k\top}])$
11: **end if**
12: $\mathbf{U}, \Sigma, \mathbf{V}^\top =$ `svd`$(\mathbf{M})$
13: `send-to-client`$(\mathbf{V}^{k\top}, \Sigma^k)$
14: **Aggregator**
15: **Return** $\mathbf{V}^{k\top}, \Sigma^k$ ▷ Return approximate subspace of $\mathbf{A}^\top\mathbf{A}$.

QR based PCA

Bai et al. [2] propose a conceptually different method, which still only requires one communication round per participant. Their method is not designed for a star like architecture, but could be considered for the cross-silo P2P-architectures cited earlier [39, 6], therefore we include it in the accuracy analysis. It can also be adapted to the star-like architecture by modifying the merge procedure. See algorithm 2 for a pseudocode description of this algorithm. At the local sites s a QR factorization of the data matrix is computed and $\mathbf{R}_s$ is sent to the aggregator (line 2). From the local local QR factorizations the global PCA is computed by stacking all $\mathbf{R}_s$ matrices vertically to form $\mathbf{R}' \in \mathbb{R}^{n \times m}$. $\mathbf{R}'$ is decomposed into $\mathbf{Q}, \mathbf{R}''$ (lines 7 to 8). In the final step, the singular value decomposition of $\mathbf{R}'' = \mathbf{U}\Sigma\mathbf{V}^\top$ is computed and the top k eigenvector matrix $\mathbf{V}^{k,\top}$ is returned as the eigenvector of $\mathbf{A}^\top\mathbf{A}$ (line 9).

Algorithm 2 Federated PCA using QR factorization [2]

Require: Data matrices $\mathbf{A}_s \in \mathbb{R}^{n_s \times m}$, # eigenvectors k.
1: **Client**
2: $\mathbf{Q}_s, \mathbf{R}_s \leftarrow$ `orthonormalize`$(\mathbf{A}_s)$
3: `send-to-aggregator`$(\mathbf{R}_s)$
4: **Client**
5: **Aggregator**
6: $[\mathbf{R}_s] \leftarrow$ **for** $s \in [S]$ `get-from-client`$(\mathbf{R}_s)$
7: $\mathbf{R}' \leftarrow$ `stack-vertically`$([\mathbf{R}_s])$
8: $\mathbf{Q}, \mathbf{R}'' \leftarrow$ `orthonormalize`$(\mathbf{R}')$
9: $\mathbf{U}, \Sigma, \mathbf{V}^\top =$ `svd`$(\mathbf{R}'')$
10: `send-to-client`$(\mathbf{V}^{k\top}, \Sigma^k)$
11: **Aggregator**
12: **Return** $\mathbf{V}^k$ ▷ Return eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.

Federated subspace iteration

Federated subspace iteration is a direct extension of the centralized subspace iteration [14] and has been formulated in different versions [15, 3, 28]. It is the extension of power iteration which computes one vector at the time. Subspace iteration is described in algorithm 3. Initially, a random eigenvector estimate $\mathbf{V}_{i=0}$ is generated at the aggregator as the current eigenvector estimate and orthonormalised (lines 1 to 3). The procedure then iteratively refines this estimate. It consists of a local phase and a global phase. In the local phase the current candidate eigenvector matrix $\mathbf{V}_{i-1}$ is multiplied by the covariance matrix of the local data to form $\mathbf{V}_{s,i} = \mathbf{A}_s^\top \mathbf{A}_s \mathbf{V}_{i-1}$ (lines 7 to 9). This candidate matrix $\mathbf{V}_{s,i}$ is sent to the aggregator where the global estimate is computed by adding up the local eigenvector estimates $\mathbf{V}_{s,i}$ element-wise over the local estimates. The candidate eigenvector matrix is normalized using QR factorization and sent back to the clients (lines 12 to 15). This procedure is repeated until convergence.

Algorithm 3 Federated Subspace Iteration

Require: Data matrices $\mathbf{A}_s \in \mathbb{R}^{n_s \times m}$, # eigenvectors k.
Generate $\mathbf{V}_0 \in \mathbb{R}^{m \times k}$ randomly ▷ Initialize candidate eigenvector matrix of $\mathbf{A}^\top \mathbf{A}$.
$\mathbf{V}_0 \leftarrow$ `orthonormalize`$(\mathbf{V}_0)$
$i \leftarrow 1$ ▷ Initialize iteration counter.
while termination criterion not met **do**
  Client
    $V_{i-1} \leftarrow$ `get-from-aggregator`()
    $\mathbf{V}'_{s,i} = \mathbf{A}_s \mathbf{V}_{i-1}$ ▷ Update local eigenvectors $V_{s,i}$
    $\mathbf{V}_{s,i} = \mathbf{A}_s{}^\top \mathbf{V}'_{s,i}$
    `send-to-aggregator`$(\mathbf{V}_{s,i})$
  Client
  Aggregator
    $[\mathbf{V}s, i] \leftarrow$ `get-from-client`()
    $\mathbf{V}_i = \sum_s \mathbf{V}_{s,i}$ ▷ Add up $V_{s,i}$ element wise.
    $\mathbf{V}_i =$ `orthonormalize`$(\mathbf{V}_i)$
    `send-to-client`(V_i)
    $i \leftarrow i + 1$
  Aggregator
end while
$\mathbf{V}^k \leftarrow \mathbf{V}_i^k$
Return $\mathbf{V}^k$ ▷ Return converged eigenvectors of $\mathbf{A}^\top \mathbf{A}$.

Vertical partitioning

In this article, we discuss the algorithms and applications of federated PCA for horizontally partitioned data. Please note, that some applications in computational biology (for instance population stratification) require the decomposition of the sample-by-sample covariance matrix, which cannot be solved directly with every one of the algorithms evaluated in this manuscript. In the vertical case, the computation of the entire covariance matrix is not possible because for two sites i and j with n_i and n_j samples respectively only the partial covariance matrices $M_{i,i} \in \mathbb{R}^{n_i \times n_i}$ and $M_{j,j} \in \mathbb{R}^{n_j \times n_j}$ can be computed while the computation of $M_{i,j} \in \mathbb{R}^{n_i \times n_j}$ and $M_{j,i} \in \mathbb{R}^{n_j \times n_i}$ would require the transfer of the samples of site i to site j. To illustrate this, consider a gene panel as example. Every hospital measures d genes for their n_i patients. At every site s, we can compute the gene-by-gene covariance matrix with the full dimensionality $d \times d$, but we can only compute the partial patient-by-patient covariance matrices of $n_s \times n_s$ at each site without exchanging patient level information. Nasirigerdeh et al. [27] have shown that for federate GWAS piplines, exchanging the entire sample eigenvectors potentially leads to a privacy breach where binary covariates of participants can be disclosed. Therefore, care has to be taken when exchanging the sample eigenvectors. In [16], this problem is presented in greater detail, and an algorithm is developed, which solves this problem efficiently and without materializing the covariance matrix or exchanging the sample eigenvectors at all.

Other related methods

A plethora of algorithms has been designed for distributed sensor networks dealing with both horizontal data partitioning and vertical partitioning, including but not limited to work described in [5, 13, 18, 33] and [43]. These algorithms cover cross-device federated learning. In contrast to cross-silo federated learning where large chunks of data are available at the sites, cross-device federated learning assumes a high number of devices such as sensors or mobile phones with limited compute power and relatively little data belonging to only one user. Due to the differing assumptions on architecture and computational resources, and the frequent use of (randomized) P2P communication, algorithms for this use case will not be considered here. Chen et al. [8] describe a gradient method which uses matrix deflation for the computation of more than one eigenvector which is impractical due to the increased communication effort (c.f. [16]).

Test setup and metrics

In the optimal case the federated PCA produces exactly the centralized solution. In order to estimate the performance of the algorithms on the realistic data from TCGA, we simulate the execution of the federated algorithm with the data distributed according to the TSS as described above. Since some of the sample sites are quite small, in a second experiment we additionally group several sample sites together to form larger 'meta-sample-sites' of approximately the same size each using a greedy heuristic. We chose this strategy to better investigate approaches that compute local subspaces. As explained above, the dimension of such a subspace is strictly limited by the number of samples.

To evaluate the algorithms' performance, we compare the result of the federated PCA to the solution computed on the centralized data. As a reference implementation we use the implementation in `scipy.sparse.linalg` which internally uses the LAPACK package. The comparison of the models is done by calculating the angles between the leading eigenvectors. We chose the angle between the eigenvectors over the subspace reconstruction error, because the 'loadings', the coordinates of the eigenvectors, are routinely used in gene expression analysis, for example to detect correlated genes [12]. Therefore, the individual coordinates of the eigenvectors must be taken into account when comparing the resulting subspaces. For applications which only rely on the projected data, and not on the individual loadings, we compute the subspace reconstruction error. The subspace reconstruction error is the Euclidean distance of the original data from the reconstructed 'denoised' data It is defined as $|\mathbf{A}\mathbf{V}^k\mathbf{V}^{k\top} - \mathbf{A}|_2$.

Complementing the simulated results which established the accuracy of the methods, we also provide a real implementation of the algorithms. We use the FeatureCloud [26] platform for this purpose and implemented an app, that can compute all previously presented algorithms. The application has multiple modes, including a batch mode and a train/test mode allowing for cross validation splits. We then set up a test using the FeatureCloud 'Testbed', which allows to simulate a federated setting by spawning multiple clients on the same machine. The parameters are passed via a remote relay server, meaning the transmission is close to a realistic estimate. For more details on this system, please refer to the website `featurecloud.ai` and the publication [26]. We measure the wall clock time, the number of iterations and the number and size of sent packages. The tests were run on a UNIX server with 502GB RAM and 64 CPUs partially in parallel. `AP-COV` has been omitted because it is as accurate as `AP-STACK` and has the same communication properties as `P-COV`. We used a randomly generated data set

Table 2. The parameter choices for the truly federated implementation of PCA.

Parameter	Choices
algorithm	`P-COV`, `AP-STACK`, `SUB-IT`, `QR-PCA`
clients	3, 5
data sets	random, `MNIST`
epsilon	$10e^{-9}$

with 5000 samples and 10 features, and the `MNIST` data set. They were randomly chunked into 3, and 5 equal chunks and repeated 5 times. We set the termination criterion to $1e^{-9}$ for `SUB-IT`. Table 2 summarizes the investigated parameters.

Practical illustration using integrated Psoriasis data

A possible use of PCA is the vizualization of the data to detect batch effects, systematic shifts in the data distribution due to different processing of the data. We illustrate this use case with a publicly available collection of Psoriasis data sets. The studies were originally not concieved as a federated study, but have been manually curated and preprocessed following the same computational pipeline [11]. We investigate the differences of federated exact PCA, federated approximate PCA and the naïve superimposition of the local PCA spaces to show whether they can be used to accurately determine the presence of batch effects in the data.

Results

Analysis of the exchanged and final parameters

Firstly, we analyze the actually transmitted information and investigate which algorithm discloses the highest amount of information. V denotes the complete eigenvector matrix and V^k and $V^{k'}$ the matrices containing the top k and k' eigenvectors respectively. Table 3 summarizes the parameters known to the clients and the aggregator at the end of the run. Note that V allows the computation of the covariance matrix M and V^k and $V^{k'}$ analogously its approximations M^k and $M^{k'}$. Given the aggregator has all the exact local covariance matrices it can run local subspace iteration and therefore access V_i^k when using `P-COV` and `QR-PCA`, given the same initialisation. Following this reasoning, we claim that in terms of disclosed knowledge, `P-COV` and `QR-PCA` are equivalent and disclose the highest amount of information. `AP-COV` and `AP-STACK` disclose larger V^k subspaces than `SUB-IT` which only discloses the required V^k, however `SUB-IT` might be prone to iterative leakage and disclose the covariance matrix. It is outside the scope of this manuscript to try and attack either algorithm. It is also apparent that there is an asymmetry of the knowledge of the parameters in the chosen architecture which favors the aggregator who gains knowledge of all intermediate steps. This asymmetry is due to to the chosen architecture, and can be trivially resolved by adopting a P2P architecture at the expense of increased network traffic. Another solution to this problem is the use of secure multiparty aggregation [9] which can be used to hide parameters with a additive aggregation strategy. This applies to `P-COV`, `AP-COV` and `SUB-IT` but not trivially to `QR-PCA` and `AP-STACK` which use QR orthogonalisation and singular value decomposition as their respective aggregation strategies.

Table 3. Summary of the transmitted parameters and the computable parameters. s denotes the knowledge of the local parameter, S denotes the knowledge of all local parameters and hence the aggregate. A * indicates which parameters can be hidden via the use of secure addition.

Algorithm	V^k		$V^{k'}$		V		R_s		V_i^k	
	C	A	C	A	C	A	C	A	C	A
`P-COV`	s*	S	s*	S	s*	S	s*	S	s*	S
`AP-COV`	s*	S	s*	S						
`AP-STACK`	s	S	s	S						
`SUB-IT`	s*	S							s*	S
`QR-PCA`	s	S	s	S	s	S	s	S	s	S

Table 4. Summary of the transmitted parameters and the required number of iterations. (C=client, A=aggregator)

Algorithm	Param.	Direction	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{T}$
`P-COV`	$\mathbf{M}$	C $\rightarrow$ A	$d \times d$	1	$\mathcal{O}(d^2)$
	$\mathbf{V}^k$	C $\leftarrow$ A	$d \times k$	1	
`AP-COV`	$\mathbf{V}^{k'}$	C $\rightarrow$ A	$d \times k$	1	$\mathcal{O}(dk')$
	$\mathbf{V}^k$	C $\leftarrow$ A	$d \times k$	1	
`AP-STACK`	$\mathbf{V}^{k'}$	C $\rightarrow$ A	$d \times k$	1	$\mathcal{O}(dk')$
	$\mathbf{V}^k$	C $\leftarrow$ A	$d \times k$	1	
`SUB-IT`	$\mathbf{V}^k$	C $\leftrightarrow$ A	$d \times k$	i	$\mathcal{O}(dki)$
`QR-PCA`	$\mathbf{R}$	C $\rightarrow$ A	$d \times d$	1	$\mathcal{O}(d^2)$
	$\mathbf{V}^k$	C $\leftarrow$ A	$d \times k$	1	

Network traffic

Here, we analyse the communication requirements for federated PCA. Let $\mathcal{D}$ be the dimensionality of the parameters in terms of floats transmitted between client and aggregator and let $\mathcal{N}$ be the number of communication rounds. Let $\mathcal{T}$ be the total transmission cost. Recall that the global data matrix has dimensions $A \in \mathbb{R}^{n \times d}$ and is divided into S local data matrices A_s with n_s samples and d dimensions each. k is the number of eigenvectors of the final decomposition and k' the intermediate dimensionality if applicable. i is the number of iterations for subspace iteration to converge. Table 4 summarizes the parameter and the associated transmission cost exchanged between the sites. All methods assume a centered data matrix, so the exchange of the column sums and the number of samples is required.

Accuracy on a standard image data set

To put the performance of the algorithms into perspective, we first provide accuracy values for the performance on the standard image data set `MNIST`. This data set consists of 60,000 gray scale images containing 784 pixels each. The dimensionality is hence $d < n$. Here, the performance of the algorithms is generally very good. Table 5 summarizes selected angles for each of the selected approaches using the `MNIST` data set split into 20 equal chunks. Using this data all algorithms lead to a good approximation of the subspace with low angular deviations. `SUB-IT`, `P-COV` and `QR-PCA` outperform `AP-COV` and `AP-STACK`, but by a small margin.

Table 5. Reference values for PCA performance using the `MNIST` data set with for the 1st, 5th and 10th eigenvector over 20 randomized splits

PCA version	EV1	EV5	EV10
`AP-STACK`	0.47	1.46	10.35
`AP-COV`	0.47	1.46	10.35
`P-COV`	0	0	0
`QR-PCA`	0	0	0
`SUB-IT`	0	0	0

Accuracy of the selected approaches on realistic biomedical data

Figure 2 summarizes the accuracy of the eigenvectors computed using the simulated federated PCA using TCGA data. As a measure of quality we plot the angle between the eigenvector calculated by the federated approach against the centralized singular value decomposition. The upper panel of the plot shows three sub figures, one for the original data partitioning extracted from TCGA, and one each for the 'meta-sample-sites'. Generally speaking, `SUB-IT`, as well as the `P-COV` and `QR-PCA` perform accurately regardless of the data distribution. The angle between all selected eigenvectors is close to 0. The approximate algorithms `AP-COV` and `AP-STACK` do not perform as well. They improve when creating larger meta-sites which confirms that these algorithms only perform well, when there are sufficiently many samples at each collection site. In order to put these values into perspective, in the lower panel, we also provide the ratio of the subspace reconstruction error achieve by the method divided by the gold standard reconstruction error. It is apparent, that the subspace reconstruction error degenerates less quickly, especially as the number of samples grows. This indicates, that downstream analyses relying on the coordinates of the eigenvectors are likely to suffer from the approximate approaches, while analyses merely relying on the projection of the data are more resilient against the errors introduced by the approximations. This will be further illustrated in the practical experiments.

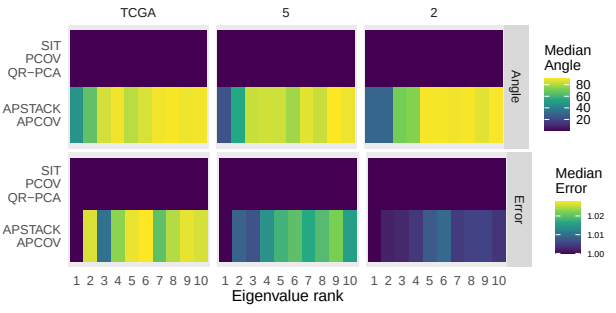


Fig. 2: Comparison of the different PCA algorithms w. r. t. to the angle between the leading eigenvectors with the TCGA data distributed according to the tissue sample site, and combined into 2 and 5 meta sites respectively. Proxy naive and the Power method achieve perfect accuracy both according to the angle between the eigenvectors (upper panel) and the subspace reconstruction error (SRE). With higher rank, the accuracy of the proxy method deteriorates.

Table 6. Results of the federated test runs using randomly generated and `MNIST` data.

Data set	Algorithm	Sites	Time[s]	Iter.	MB
`MNIST`	`P-COV`	5	27	1	10555
		3	20	1	21118
	`AP-STACK`	5	20	1	649
		3	25	1	325
	`QR-PCA`	5	23	1	11242
		3	30	1	5626
	`SUB-IT`	5	1208	1000	296073
		3	1090	1000	148180
random	`P-COV`	5	3.1	1	12
		3	3.4	1	6
	`AP-STACK`	5	2.8	1	16
		3	3.3	1	8
	`QR-PCA`	5	2.8	1	11
		3	4	1	5
	`SUB-IT`	5	778	465	6735
		3	492	439	2966

Application of federated PCA to Psoriasis data

We used the methods with a manually curated multicentric Psoriasis data set where individually performed studies were assembled and reprocessed with the same computational pipeline. In fig. 3, we show the PC plots of the data using centralized PCA which is identical to `SUB-IT`, `AP-STACK` and by superimposing the results of the local computations. The exact PCA shows that there are prominent batch effects in the data, as the data separates according to the experiment, wheras this is not replicated by `AP-STACK` and `AP-COV`. In supl. fig. 2, we show similar results for simulated data.

Practical implementation

Table 6 shows the results of the empirical runtimes of the PCA algorithms for different data sets averaged over 5 runs. `AP-STACK`, `P-COV` and `QR-PCA` have low execution times in the order of seconds. The low number of executions does not allow to rank the algorithms further. `SUB-IT` has longer execution times and requires more data transmission.

Discussion

Choice of the performance criteria

The angle between the eigenvectors is a very stringent criterion for the performance of the algorithm, as with high dimensional vectors very few deviating coordinates can lead to a drastic change of the angle. The subspace reconstruction error measures the distance between the projections of the data and the actual data and is therefore suitable for analyses that solely rely on the projected data. For the analysis of the loadings the subspace reconstruction error might be misleading, therefore we report the angle between the eigenvectors. The wall clock time and number of communication steps are both useful measures as their combination allows to estimate the run times for future studies.

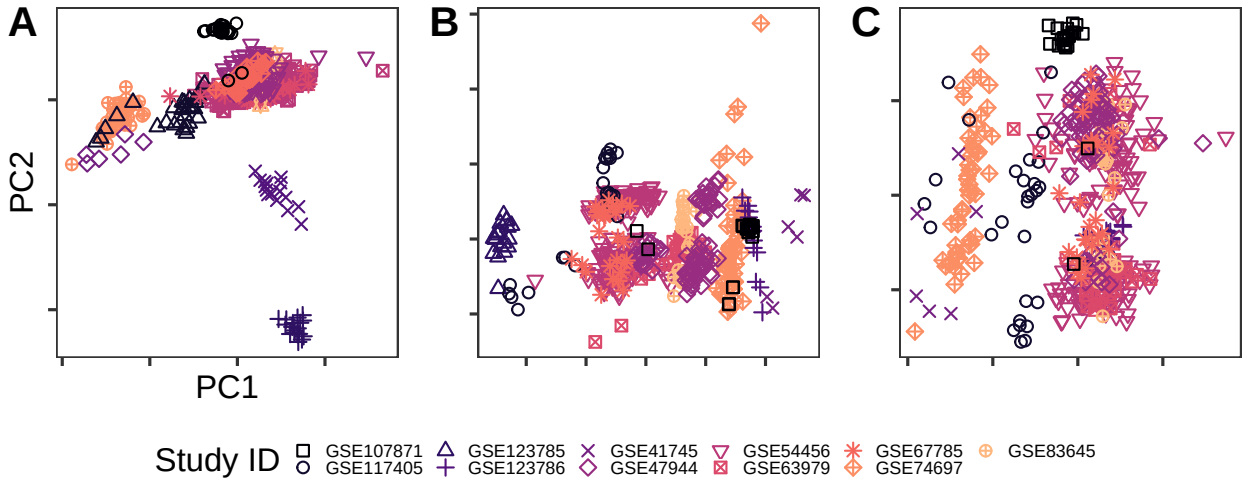


Fig. 3: Comparison of centralized PCA/`SUB-IT`, combined local projections and `AP-STACK`. There pattern for the exact PCA is markedly different from both the naïve combinations of PCA and the approximate PCA. B and C are not suited to detect the batch effects present in the data.

Analytical Performance of the methods

Approximate methods generally perform poorly according to the angular deviation of the eigenvectors, especially when retrieving many eigenvectors. This strong deviation of the federated eigenvectors from the centralized baseline implies, that the analysis of the loadings of the eigenvectors is not possible using the approximate methods and is additionally very vulnerable to outliers. The eigenvector coordinates may deviate significantly even in low ranks, therefore different hypotheses will be generated using approximate PCA compared to centralized PCA. This is further illustrated in fig. 3 where the results of exact and approximate PCA are fundamentally different, and the approximate method does not reproduce the result to an extend where the batch effect in the data cannot be detected.

According to the subspace reconstruction error for the TCGA data, most methods perform reasonably well even on unfavorably distributed data with many small subsets. Due to the centralized processing, TCGA data is still likely to give optimistic estimates of the error incurred due to federated approximate PCA, meaning that the sites do not show a fundamentally different data distribution. It has recently been argued, that UMAP and t-SNE should only be used for the coarse analysis of the data [7]. Furthermore, analyses should be robust to perturbation, thus using approximate PCA should not disturb the signal strongly enough to disrupt the major sources of variation in the data.

Computational performance of the methods

Our practical implementations show that the overhead through the use of the federated methods is acceptable given the long sourcing process of biological data and the potential privacy gain. The FeatureCloud platform introduces a certain overhead through virtualization and encryption techniques. Nonetheless, with the run time of a few seconds to a few minutes for federated principal component analysis researchers can realistically use these methods in practice. The bottleneck is the number of communication steps required, therefore federated power iteration is slower than the single round methods. The advantage of `SUB-IT` is, that it can cope with a higher number of features while performing exact PCA, contrary to `QR-PCA` and `P-COV` which need to load the covariance matrix into memory. The convergence criterion for power iteration is also set very stringent, so a lower number of iterations and thus a decreased run time might be sufficient in practice.

Privacy of federated SVD

Critiques may raise the issue of the privacy of the parameters. Indeed, the amount of data transmitted between the sites is quite large since they are the result of a matrix decomposition. Recall that we are working on the matrix $A \in \mathbb{R}^{n \times d}$, where n is the number of samples and d is the number of features and $A = U\Sigma V^\top$. This means each vector u in $U = [u_1, u_2, ..., u_k]$ contains elements belonging to the samples, whereas $V = [v_1, v_2, ..., v_k]$ summarizes the features across all samples. If all participants are to receive the complete SVD, then the aggregator has to broadcast the final U and V to all the clients. In previous work, it has already been shown that federated pipelines which include the use of the sample eigenvectors U are prone to data leakage when broadcasting the full eigenvectors. Therefore, we highly recommend to not broadcast the sample specific eigenvectors U. Ideally, this would happen in an oracle fashion, where the parties gain knowledge of the output, but none of the intermediate parameters. Unfortunately, this is not the case, therefore section 3.1 we established a hierarchy of the approaches in terms of trivially reconstructable data and parameters.

Several articles discuss privacy preserving PCA or power iteration in a federated setting via encryption and secure multiparty computation (SMPC) techniques [42, 1, 28, 30] or differential privacy (DP) [17, 38, 15, 3]. A few articles [42, 24]

assume that the aggregated covariance matrix is private. The authors of [28] assume that the aggregated eigenvector updates are private. Generally speaking, if the aggregated parameters are considered private, since the methods generally only require additive aggregation, secret sharing by sharding the data or using fixed-point arithmetic, can be added with relatively little overhead [9]. A protocol that does not use the clear-text covariance matrix has been proposed by [1] who use a garbled circuit protocol that allows to compute the eigenvectors securely based on the homomorphically aggregated covariance matrix. The latter approach is quite time intensive on small data sets in simulation. [30] introduces improved primitives required for PCA using homomorphically encrypted centralised data. The empirical evaluation unfortunately does not include data at the scale of high-dimensional biological data ($d = 20$ is the largest dimensions) and a realistic number of iterations ($i = 5$ is the total number of iterations). The extension to the federated setting provides an additional challenge. The high dimensionality of the data is an obstacle for DP as the noise scales with the number of variables. Since the number of variables is large, these approaches cannot be used without severe degradation of the results. Furthermore, the choice of a good ϵ is not easy in practice. A major obstacle in the adoption of these techniques is the lack of ready-to-use libraries implementing the methods. Lastly, while it is possible to retrieve the eigenvectors privately, most of the downstream analyses in bioinformatics use the projections of the data onto the eigenvectors. Therefore, even differentially private eigenvectors are not sufficient to ensure privacy. (Due to the use of the data for the projection, this operation does not fall under the closure under post-processing). This needs to be considered when working with PCA in a federated setting and the potential risks have to be evaluated on a case-by-case basis.

Choice of an appropriate algorithm

In the following section, we will give guidelines for the choice of an appropriate algorithm which we summarize in fig. 4. Due to the various considerations in a federated study, not every algorithm is appropriate for every setting. Non-star like architectures can achieve $\mathcal{O}(log_2(n))$ communication steps when using single round approaches with n the number of clients. In terms of data disclosure and storage requirements, which amounts to the entire covariance matrix, the methods are equal. Since the aggregation method in `QR-PCA` is a QR factorization for which secure aggregation is not immediately possible, `P-COV` should be preferred when secure aggregation is required. If an approximate eigenvector is sufficient, `AP-COV` and `AP-STACK` are useful. However, only `AP-COV` allows secure aggregation because `AP-STACK` uses SVD as its aggregation method. Furthermore, `AP-COV` and `AP-STACK` depend on a high number of samples to achieve good results and may fail catastrophically in practice when data is limited or confounded. `SUB-IT` is exact and only discloses V^k to the aggregator given a limited number of iterations. Our practical implementations show that all methods can achieve reasonable run times. The potential data leakage induced by subspace iteration remains an open question and will be subject to further research.

Other recommendations

In addition to choosing an appropriate algorithm, we suggest to carefully consider the information that is required for downstream analyses. Notably, restricting the number of eigenvectors k keeps the amount of communicated data and the number of communication steps low. It is possible to retrieve further eigenvectors should this be required later on. Before the use of the federated tools, it should be considered whether all local data sets need to be cleaned from obvious outliers and it should be assessed whether the populations at the clients are eligible for joint analysis. The inspection for outliers and their removal are necessary, because outliers have a disproportional effect on the PCA. In practice the presence of outliers in the PCA can warrant a recomputation of the PCA and lead to unnecessary information disclosure if the outliers could have been detected beforehand. On the other hand, local outliers may not be outliers globally. In this case, as sampling approach can be chosen, where instead of using the original data, artificial data points are generated. They represent the variability of the data and allow the clients to locally assess, whether their data points are indeed outliers, or if the are part of a group that is small at one client, but has more samples at another client. A local PCA can be also be performed to assess the information content of the components, including the computation of the eigengap, an indicator for the convergence behavior of subspace iteration (the larger the eigengap, the quicker the convergence). The principle of only transmitting strictly required information is equally true for the transmission of the projections. If a transmission is necessary, users should consider whether approximations of the data such as density estimates or an ellipse might be sufficient to identify batch effects.

Conclusion

In this article, we identified existing methods for federated principal component analysis, evaluated them using a realistic non-iid setting as well as random data distributions, and provide a practical illustration of its application using transcriptomics data. Importantly, we implemented the methods and benchmarked their run time, providing valuable information for the applicability of the algorithms in a real setting. Additionally, we provided easy to follow guidelines for the future users to select the most appropriate algorithm and highlight important considerations before conducting a federated analysis.

Future work on federated PCA will include the reduction of communication cost for exact subspace iteration, which could be achieved by updating only relevant coordinates. Furthermore, it needs to be investigated under which conditions the communication of the projections of the data realistically constitutes a privacy breach, i.e. whether it is possible to infer sufficient information on the individual data sets from the shared statistics.

Competing interests

There is no competing interest.

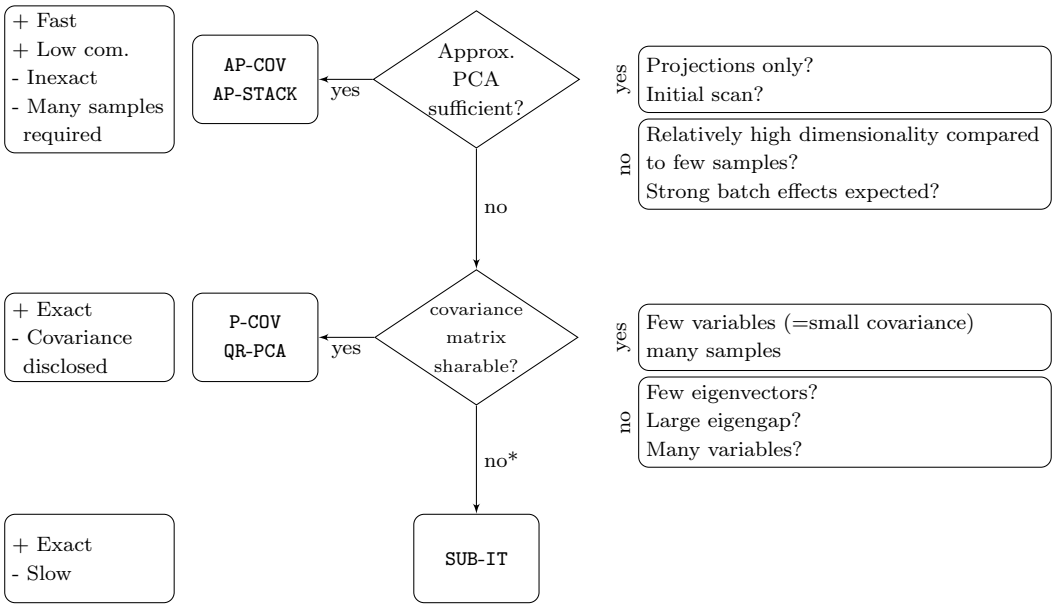


Fig. 4: Decision help for researchers intending to use federated PCA. *It is likely that with sufficient iterations the covariance matrix may be reconstructed.

Author contributions statement

A.H. and R.R. conceived the experiments, A.H. conducted the experiments and analyzed the results. A.H. and R.R. wrote and reviewed the manuscript.

Acknowledgments

The FeatureCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078. This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

References

1. M Al-Rubaie, P.-Y. Wu, J M Chang, and S.-Y. Kung. Privacy-preserving PCA on horizontally-partitioned data. *2017 IEEE Conference on Dependable and Secure Computing*, pages 280–287, 2017.
2. Zheng-Jian Bai, Raymond H. Chan, and Franklin T. Luk. Principal component analysis for distributed data sets with updating. In Jiannong Cao, Wolfgang Nejdl, and Ming Xu, editors, *Advanced Parallel Processing Technologies*, pages 471–483, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
3. Maria-Florina Balcan, Simon Shaolei Du, Yining Wang, and Adams Wei Yu. An improved gap-dependency analysis of the noisy power method. *29th Annual Conference on Learning Theory*, 49:284–309, 23–26 Jun 2016.
4. Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3113–3121, Cambridge, MA, USA, 2014. MIT Press.
5. Alexander Bertrand and Marc Moonen. Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA. *Signal Processing*, 104:120–135, 2014.
6. Oya Beyan, Ananya Choudhury, Johan van Soest, Oliver Kohlbacher, Lukas Zimmermann, Holger Stenzhorn, Md. Rezaul Karim, Michel Dumontier, Stefan Decker, Luiz Olavo Bonino da Silva Santos, and Andre Dekker. Distributed Analytics on Sensitive Medical Data: The Personal Health Train. *Data Intelligence*, 2(1-2):96–107, 2020.
7. Tara Chari, Joeyta Banerjee, and Lior Pachter. The Specious Art of Single-Cell Genomics. *BioRxiv*, pages 1–23, 2021.
8. Xi Chen, Jason D. Lee, He Li, and Yun Yang. Distributed Estimation for Principal Component Analysis: An Enlarged Eigenspace Analysis. *Journal of the American Statistical Association*, 2021.
9. Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.
10. Jianqing Fan, Dong Wang, Kaizheng Wang, and Ziwei Zhu. Distributed estimation of principal eigenspaces. *Annals of Statistics*, 47(6):3009–3031, 2019.
11. Antonio Federico, Veera Hautanen, Nils Christian, Andreas Kremer, Angela Serra, and Dario Greco. Manually curated and harmonised transcriptomics datasets of psoriasis and atopic dermatitis patients. *Scientific Data*, 7(1):5–10, 2020.
12. Rudolf S.N. Fehrmann, Juha M. Karjalainen, Malgorzata Krajewska, Harm Jan Westra, David Maloney, Anton Simeonov, Tune H. Pers, Joel N. Hirschhorn, Ritsert C. Jansen, Erik A. Schultes, Herman H.H.B.M. Van Haagen,

Elisabeth G.E. De Vries, Gerard J. Te Meerman, Cisca Wijmenga, Marcel A.T.M. Van Vugt, and Lude Franke. Gene expression analysis identifies global gene dosage sensitivity in cancer. *Nature Genetics*, 47(2):115–125, 2015.

13. Jerome Fellus, David Picard, and Philippe-Henri Gosselin. Asynchronous gossip principal components analysis. *Neurocomputing*, 169:262–271, dec 2015.
14. N Halko, P G Martinsson, and J A Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. pages 1–74.
15. Moritz Hardt and Eric Price. The noisy power method: A meta algorithm with applications. *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, page 2861–2869, 2014.
16. Anne Hartebrodt, Reza Nasiridergeh, David B. Blumenthal, and Richard Röttger. Federated principal component analysis for genome-wide association studies. ICDM 2021. ICDM 2021, 2021.
17. Hafiz Imtiaz and Anand D. Sarwate. Differentially private distributed principal component analysis. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018-April:2206–2210, 2018.
18. Márk Jelasity, Geoffrey Canright, and Kenth Engø-Monsen. Asynchronous Distributed Power Iteration with Gossip-Based Normalization. In *Euro-Par 2007*, pages 514–525. 2007.
19. Ian T. Joliffe. *Principal Component Analysis*. Springer-Verlag, 2002.
20. Katie Kerr, Helen McAneney, Laura J. Smyth, Caitlin Bailie, Shane McKee, and Amy Jayne McKnight. A scoping review and proposed workflow for multi-omic rare disease research. *Orphanet Journal of Rare Diseases*, 15(1):1–18, 2020.
21. Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis, and Dimitrios I. Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8–17, 2015.
22. Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 2005. [Online; accessed 27-02-2020].
23. Bo Liu, Mohamed Mohandes, Hilal Nuha, Mohamed Deriche, and Faramarz Fekri. A Distributed Principal Component Analysis Compression for Smart Seismic Acquisition Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 56(6):3020–3029, 2018.
24. Yingting Liu, Chaochao Chen, Longfei Zheng, Li Wang, Jun Zhou, Guiquan Liu, and Shuang Yang. Privacy Preserving PCA for Multiparty Modeling. 2020.
25. F. Martin-Sanchez and K. Verspoor. Big data in medicine is driving big changes. *Yearbook of medical informatics*, 9:14–20, 2014.
26. Julian Matschinske, Julian Späth, Reza Nasirigerdeh, Reihaneh Torkzadehmahani, Balázs Orbán, Sándor Fejér, Olga Zolotareva, and Mohammad Bakhtiari. The FeatureCloud AI Store for Federated Learning in Biomedicine and Beyond. pages 1–32.
27. Reza Nasirigerdeh, Reihaneh Torkzadehmahani, Julian Matschinske, Tobias Frisch, Markus List, Julian Späth, Stefan Weiß, Uwe Völker, Dominik Heider, Nina Kerstin Wenke, Tim Kacprowski, and Jan Baumbach. sPLINK: A Federated, Privacy-Preserving Tool as a Robust Alternative to Meta-Analysis in Genome-Wide Association Studies. 6(Figure 1):1–16, 2020.
28. Manas A. Pathak and Bhiksha Raj. Efficient protocols for principal eigenvector computation over private data. *Transactions on Data Privacy*, 4(3):129–146, 2011.
29. Yongming Qu, George Ostrouchov, Nagiza Samatova, and Al Geist. Principal Component Analysis for Dimension Reduction in Massive Distributed Data Sets. *Workshop on High Performance Data Mining at the Second SIAM International Conference on Data Mining*, pages 4–9, 2002.
30. Deevashwer Rathee, Pradeep Kumar Mishra, and Masaya Yasuda. Faster PCA and linear regression through hypercubes in HElib. *Proceedings of the ACM Conference on Computer and Communications Security*, (1):42–53, 2018.
31. Chris A. Rees, Natalie Pica, Michael C. Monuteaux, and Florence T. Bourgeois. Noncompletion and nonpublication of trials studying rare diseases: A cross-sectional analysis. *PLoS Medicine*, 16(11):1–16, 2019.
32. Miguel Ángel Rodríguez, Alberto Fernández, Antonio Peregrín, and Francisco Herrera. *A Review of Distributed Data Models for Learning*. Springer International Publishing, Cham, 2017.
33. Ioannis D. Schizas and Abiodun Aduroja. A Distributed Framework for Dimensionality Reduction and Denoising. *IEEE Transactions on Signal Processing*, 63(23):6379–6394, 2015.
34. Giorgio Sirugo, Scott M. Williams, and Sarah A. Tishkoff. The Missing Diversity in Human Genetic Studies. *Cell*, 177(1):26–31, 2019.
35. Anthony Steed and Manuel Fradinho Oliveira. More than two. *Networked Graphics*, pages 125–168, 2010.
36. Fabian J Theis. Current best practices in single-cell RNA-seq analysis : a tutorial. *Mol Syst Biol*, 2019.
37. Evert Ben van Veen. Observational health research in Europe: understanding the General Data Protection Regulation and underlying debate. *European Journal of Cancer*, 104:70–80, 2018.
38. Sen Wang and J. Morris Chang. Differentially Private Principal Component Analysis over Horizontally Partitioned Data. *DSC 2018 - 2018 IEEE Conference on Dependable and Secure Computing*, pages 1–8, 2019.
39. Stefanie Warnat-Herresthal et al. Swarm Learning for decentralized and confidential clinical machine learning. *Nature*, 594(7862):265–270, 2021.
40. John N. Weinstein et al. The cancer genome atlas pan-cancer analysis project. *Nature Genetics*, 45(10):1113–1120, 2013.
41. Jenna Wiens, Suchi Saria, Mark Sendak, Marzyeh Ghassemi, Vincent X. Liu, Finale Doshi-Velez, Kenneth Jung, Katherine Heller, David Kale, Mohammed Saeed, Pilar N. Ossorio, Sonoo Thadaney-Israni, and Anna Goldenberg. Do no harm: a roadmap for responsible machine learning for health care. *Nature Medicine*, 25(September), 2019.
42. Hee-Sun Won, Sang-Pil Kim, Sanghun Lee, Mi-Jung Choi, and Yang-Sae Moon. Secure principal component analysis in multiple distributed nodes. *Security and Communication Networks*, 9(14):2348–2358, sep 2016.

43. Sissi Xiaoxiao Wu, Hoi To Wai, Lin Li, and Anna Scaglione. A Review of Distributed Algorithms for Principal Component Analysis. *Proceedings of the IEEE*, 106(8):1321–1340, 2018.

Anne Hartebrodt is a PhD candidate at the University of Southern Denmark. She received her M.Sc. degree from Technical University of Munich (TUM) and Ludwig-Maximilians University (LMU). Her main research focus is federated unsupervised machine learning for biomedical data.

Richard Röttger obtained his PhD (Dr. rer. nat.) degree from the University of the Saarland and the Max Planck Institute for Informatics in 2014. Currently, he is Associate Professor for Bioinformatics at the University of Southern Denmark (SDU). In the framework of the EU H2020 project FeatureCloud, his team develops federated machine learning approaches for sensitive biomedical data.

Chapter 3

Manuscript 2

Federated Principal Component Analysis for Genome-Wide Association Studies

Federated Principal Component Analysis for Genome-Wide Association Studies

First Author*, Second Author†, Third Author‡,§ and Fourth Author*,§
*First Organization, First City, First Country
Email: {first.author,fourth.author}@first.organisation
†Second Organization, Second City, Second Country
Email: second.author@second.organisation
‡Third Organization, Third City, Third Country
Email: third.author@third.organisation
§Joint senior authors.

***Abstract*—Federated learning (FL) has emerged as a privacy-preserving alternative to centralized data analysis. Especially for biomedical analyses such as genome-wide association studies (GWAS), the fact that the data remain with the owners has the potential to enable studies which were previously impossible due to privacy protection regulations. Principal component analysis (PCA) is a frequent preprocessing step in GWAS, where the eigenvectors of the sample by sample covariance matrix are used as covariates in the statistical tests. Therefore, a federated version of PCA suitable for vertical data partitioning is required for federated GWAS. Existing federated PCA algorithms exchange the complete sample eigenvectors, a potential privacy breach. In this paper, we present a federated PCA algorithm for vertically partitioned data which does not exchange the sample eigenvectors and is hence suitable for federated GWAS. We prove that our federated algorithm is equivalent to a state-of-the-art centralized method, and empirically show that it outperforms existing federated solutions in terms of convergence behavior and scalability. In addition, we provide a user-friendly privacy-preserving web tool to promote acceptance of federated PCA among non-computer-scientist GWAS researchers.**

I. Introduction

Federated learning (FL) has recently gained attraction as a privacy-preserving alternative to centralized computation. Instead of consolidating the data on a central server, the data holders keep ownership of their data and send only parameters to an aggregation server [1]. An attractive application case for FL are genome-wide association studies (GWAS), which investigate the relationship of genetic variation with phenotypic traits on large cohorts [2], [3]. Genetic data is extremely sensitive in its nature and data holders hence cannot make it publicly available. The practical feasibility of using hybrid-federated learning, a combination of FL with additional privacy-preserving techniques, for GWAS has been demonstrated recently [4], [5] and an implementation is available online.

Since GWAS are often done on populations of mixed ancestry, cryptic population confounders should be controlled for before associating the genetic variants to the phenotypic trait of interest. The standard way for doing this is to compute the leading eigenvectors of the sample covariance matrix via principal component analysis (PCA), and including these eigenvectors as confounding variables to models used for the association tests [6], [7].

For federated GWAS, a PCA algorithm for vertically partitioned data is required for computing the eigenvectors (please refer to section II for a detailed explanation). Although a few such algorithms are available [8]–[11], none of them is suitable for federated GWAS. More precisely, the algorithms reviewed in [11] use client-to-client communication and are therefore unsuitable for the star-like FL architectures used in GWAS, where relatively few data holders collaborate in a static setting. The algorithms presented in [8] and [9] rely on estimating a proxy covariance matrix and hence do not scale to large GWAS datasets, which often contain genetic variation data for hundreds of thousands of individuals. To the best of our knowledge, the only covariance free PCA algorithm suitable for a star-like architecture has been presented in [10]. However, this algorithm broadcasts the complete first $k-1$ sample eigenvectors to the aggregator, which constitutes a privacy leakage that should be avoided in federated GWAS.

Extrapolating from the shortcomings of existing approaches, we can state that, for federated GWAS, a PCA algorithm for vertically partitioned data is required that combines the following properties:

- The algorithm should be suitable for a star-like FL architecture, i. e., require only client-to-aggregator but no client-to-client communication.
- The algorithm should not rely on computing or approximating the covariance matrix.
- The algorithm should not broadcast complete sample eigenvectors to the aggregator or to the clients.

In this paper, we present the first algorithm that combines all of these desirable properties and can hence be used for federated GWAS (and all other applications where these properties are required). We prove that our algorithm is equivalent to centralized vertical subspace iteration [12], a state-of-the-art centralized, covariance-free PCA algorithm. Moreover, we show in an empirical evaluation that the eigenvectors computed by our approach converge to the centrally computed eigenvectors after sufficiently many iterations.

Additionally, we developed a user-friendly web service, which implements our algorithm and hence makes it available to non-computer-scientist researchers working in the GWAS field. To further protect against potential reconstruction attacks at the aggregator, the shared model parameters are protected via secure multi-party computation (SMPC) like parameter obfuscation. The online tool is available at https://xxxxxxxxxxxxxxxxxxxxxxx/. To the best of our knowledge, it is the first ready-to-use implementation of a federated PCA algorithm. Note that providing such an implementation is crucial for federated GWAS solutions to be adopted in practice, because GWAS scientists tend to rely on ready-made software such as PLINK [13]. In sum, this paper contains the following contributions:

- We present the first federated PCA algorithm for vertically partitioned data which meets the requirements that apply in federated GWAS settings.
- We prove that our algorithm is equivalent to centralized power iteration and show that it exhibits an excellent convergence behavior in practice.
- We present a user-friendly privacy-preserving web service that implements the proposed algorithm and thereby makes it available to the GWAS community.

The remainder of this paper is organized as follows: In Section II, we introduce concepts and notations that are used throughout the paper. In Section III, we discuss related work. In Section IV, we describe the proposed algorithm. In Section V, we present the application. In Section VI, we report the results of the experiments. Section VII concludes the paper.

II. Preliminaries

A. Federated Learning and Employed Data Model

Unlike in centralized machine learning where the data is consolidated at a central server and a model is calculated on the combined data, in FL the data remains at the data owners machine. Instead of the data, only model parameters are sent to the (untrusted) aggregator which combines the local models into a global model. No raw data is exchanged in FL. See Figure 1 for a schematic comparison of centralized (cloud) learning and FL. In the cloud-based approach, data contributors send their data to a central server where the model is computed and thereby lose agency over it.

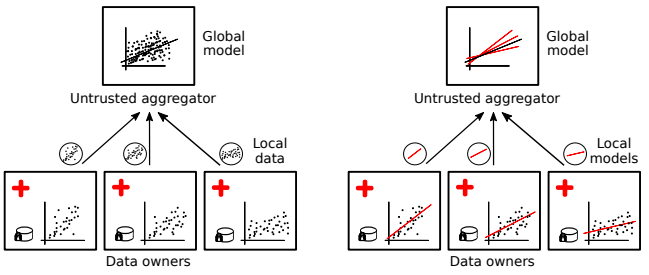


Fig. 1. Schematic comparison of traditional cloud base approaches (left) and federated learning (right).

Typically, a star-like client-aggregator architecture is used in biomedical federated solutions [4], [14], with the data holders acting as clients. The data sets at the client sites will be called *local data sets* and the parameters or models learned using this data will be called *local parameters* or *local models*, while the final aggregated model will be called *pooled model*. The optimal result of the pooled model is achieved when it equals the result of the conventional model calculated on all data, which we call the *global model*.

In federated settings, the data can be distributed in several ways. Either the clients observe a full set of variables for a subset of the samples (horizontal partitioning) or they have a partial set of variables for all samples (vertical partitioning) [11], [15]. In this paper, we assume that we are given a global data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, where m is the number of features (SNPs, in the context of GWAS) and n is the overall number of samples. The data is split across S local sites as $\mathbf{A} = [\mathbf{A}^1 \dots \mathbf{A}^s \dots \mathbf{A}^S]$, where $\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$ and n^s denotes the number of samples available at site s. From a semantic point of view, the partitioning is hence horizontal, since the samples are distributed over the local sites. However, from a technical point of view, the partitioning is vertical, since the samples correspond to the columns of $\mathbf{A}$. The reason for this rather unintuitive setup is that, when using PCA for GWAS, samples are treated as features, as detailed in the following paragraphs.

B. Principal Component Analysis

Given a data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the PCA is the decomposition of the covariance matrix $\mathbf{\Sigma} = \mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{n \times n}$ into $\mathbf{\Sigma} = \mathbf{\Gamma \Lambda \Gamma}^\top$. $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the eigenvalues $(\lambda_i)_{i=1}^n$ of $\mathbf{\Sigma}$ in non-increasing order, and $\mathbf{\Gamma} \in \mathbb{R}^{n \times n}$ is the corresponding matrix of eigenvectors [16]. Usually, one is only interested in the top k eigenvalues and corresponding eigenvectors. Since k is arbitrary but fixed throughout this paper, we let $\mathbf{G} \in \mathbb{R}^{n \times k}$ denote these first k eigenvectors (i. e., $\mathbf{G}$ corresponds to the first k columns of $\mathbf{\Gamma}$). $\mathbf{G}$ is typically used to obtain a lower-dimensional representation $\mathbf{A} \mapsto \mathbf{AG} \in \mathbb{R}^{m \times k}$ of the data matrix $\mathbf{A}$, which can then be used for downstream data analysis tasks. This, however, is not the way PCA is used in GWAS, as we will explain next.

C. Genome-Wide Association Studies

The genome stores the hereditary information that control the phenotype of an individual in interplay with the environment. The genetic information is stored in the DNA encoded as a sequence of bases (A, T, C, G), the positions are called loci. If we observe two or more possible bases at a specific locus in a population, we call this locus a *single nucleotide polymorphism* (SNP). The predominant base in a population is called the *major allele*; bases at lower frequency are called *minor alleles* [3].

Genome wide association studies seek to identify SNPs that are linked to a specific phenotype [2], [3]. Phenotypes of interest can for example be the presence or absence of diseases, or quantitative traits such as height or body mass index. The SNPs for a large cohort of individuals are tested for

association with the trait of interest. Typically, simple models such as linear or logistic regression are used for this [2], [4]. The input to a GWAS is an n-dimensional phenotype column-vector $\mathbf{y}$, a matrix of SNPs $\mathbf{A} \in \mathbb{R}^{m \times n}$, and confounding factors as column vectors. Each SNP $l \in [m]$ is tested in an individual association test

$$\mathbf{y} \sim \beta_0 + \beta_1 \cdot \mathbf{A}_{l,\bullet}^\top + \sum_{r=1}^{R} \beta_{r+1} \cdot \mathbf{x}_r + \epsilon, \tag{1}$$

where $\mathbf{A}_{l,\bullet}$ denotes the l^{th} row of $\mathbf{A}$ and the column vectors $\mathbf{x}_r \in \mathbb{R}^n$ contain confounding factors such as age or sex.

D. Principal Component Analysis for Genome-Wide Association Studies

Confounding factors such as ancestry and population substructure can alter the outcome of an association test and create false hits if not properly controlled for [3]. PCA has emerged as a popular strategy to infer population substructure, but is reported as lacking for decentralized learning [5]. More precisely, the first k (usually $k = 10$) eigenvectors $\mathbf{G} = [\mathbf{g}_1 \dots \mathbf{g}_k] \in \mathbb{R}^{n \times k}$ of the sample covariance matrix $\mathbf{A}^\top\mathbf{A}$ are included into the association test as covariates [6], [7]:

$$\mathbf{y} \sim \beta_0 + \beta_1 \cdot \mathbf{A}_{l,\bullet}^\top + \sum_{r=1}^{R} \beta_{r+1} \cdot \mathbf{x}_r + \sum_{i=1}^{k} \beta_{i+R+1} \cdot \mathbf{g}_i + \epsilon \tag{2}$$

In federated GWAS, each local site s needs to have access only to the partial eigenvector matrix $\mathbf{G}^s$ corresponding to the locally available samples. Consequently, computing the complete eigenvector matrix $\mathbf{G}$ at the aggregator and/or sharing $\mathbf{G}^s$ with other local sites s' should be avoided to reduce the possibility of information leakage. This is especially important because it has been shown that, if $\mathbf{G}$ is available at the aggregator in a federated GWAS pipeline, the aggregator can in principle reconstruct the raw GWAS data $\mathbf{A}_{l,\bullet}$ for SNP l [17]. Federated PCA algorithms that are suitable for GWAS should hence have to respect the following constraint:

Constraint 1 In a GWAS-suitable federated PCA algorithm, the aggregator does not have access to the complete eigenvector matrix $\mathbf{G}$ and each site s has access only to its share $\mathbf{G}^s$ of $\mathbf{G}$.

The PCA in GWAS is usually performed on only a subsample of the SNPs, but there seems to be no consensus as to how many SNPs should be used. Some PCA-based stratification methods rely on a small set of ancestry informative markers [18], while others employ over 100 000 SNPs [19].

Note that PCA for GWAS is conceptually different from "regular" PCA for feature reduction (cf. Figure 2). For feature reduction PCA, we would decompose the $m \times m$ SNP by SNP covariance matrix and compute a set of "meta-SNPs" for each sample. This is not what needs to be done for GWAS. Instead, the matrix which needs to be decomposed is the $n \times n$ sample by sample covariance matrix $\mathbf{A}^\top\mathbf{A}$. In our federated setting where $\mathbf{A}$ is vertically distributed across local sites $s \in [S]$,

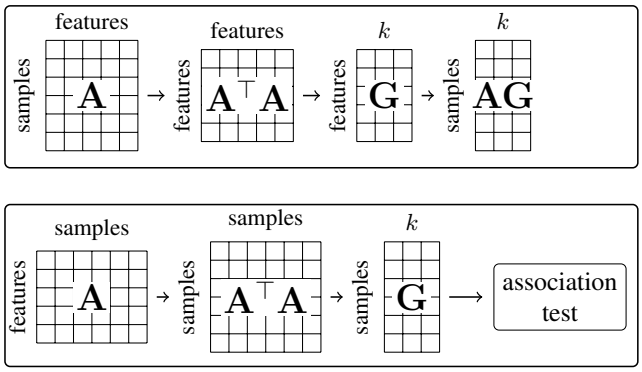


Fig. 2. Regular PCA for dimensionality reduction (upper panel); GWAS PCA for sample stratification (lower panel).

$\mathbf{A}^\top\mathbf{A}$ looks as follows (recall that, unlike in regular PCA, columns correspond to samples and rows to features):

$$\mathbf{A}^\top\mathbf{A} = \begin{pmatrix} \mathbf{A}^{1\top}\mathbf{A}^1 & \mathbf{A}^{1\top}\mathbf{A}^2 & \cdots & \mathbf{A}^{1\top}\mathbf{A}^S \\ \mathbf{A}^{2\top}\mathbf{A}^1 & \mathbf{A}^{2\top}\mathbf{A}^2 & \cdots & \mathbf{A}^{2\top}\mathbf{A}^S \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{S\top}\mathbf{A}^1 & \mathbf{A}^{S\top}\mathbf{A}^2 & \cdots & \mathbf{A}^{S\top}\mathbf{A}^S \end{pmatrix} \tag{3}$$

It is clear that $\mathbf{A}^\top\mathbf{A}$ cannot be computed directly without sharing patient level data. Moreover, with a growing number of samples, this matrix can become very large and computing it becomes infeasible. For instance, the UK Biobank — a large cohort frequently used for GWAS — contains GWAS data for more than 300 000 individuals. Furthermore, approximating the $\mathbf{A}^\top\mathbf{A}$ matrix would introduce an error. These considerations lead to the second constraint for federated PCA algorithms suitable for GWAS:

Constraint 2 A GWAS-suitable federated PCA algorithm works on vertically partitioned data and does not rely on computing or approximating the covariance matrix.

E. Gram-Schmidt Orthonormalization

The Gram-Schmidt algorithm allows to transform a set of linearly independent vectors into a set of mutually orthogonal vectors, see [20] for a proof. Given a matrix $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_k] \in \mathbb{R}^{r \times k}$ of k linearly independent column vectors, a matrix $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_k] \in \mathbb{R}^{r \times k}$ of orthogonal column vectors with the same span can be computed as

$$\mathbf{u}_i = \begin{cases} \mathbf{v}_i & \text{if } i = 1 \\ \mathbf{v}_i - \sum_{j=1}^{i-1} r_{i,j} \cdot \mathbf{u}_j & \text{if } i \in [k] \setminus \{1\} \end{cases}, \tag{4}$$

where $r_{i,j} = \mathbf{u}_j^\top \mathbf{v}_i / n_j$ with $n_j = \mathbf{u}_j^\top \mathbf{u}_j$.

The vectors can then be scaled to unit Euclidean norm as $\mathbf{u}_i \mapsto (1/\sqrt{n_i}) \cdot \mathbf{u}_i$ to achieve a set of orthonormal vectors. In the context of PCA, this can be used to ensure orthonormality of the candidate eigenvectors in iterative procedures, which otherwise suffer from numerical instability in practice [10].

TABLE I
NOTATION TABLE.

Syntax	Semantics
$[N] \subset \mathbb{N}$	index set $[N] = \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$
$S \in \mathbb{N}$	number of sites
$m \in \mathbb{N}$	number of features (i. e. SNPs)
$n \in \mathbb{N}$	total number of samples
$n^s \in \mathbb{N}$	number of samples at site $s \in [S]$
$k \in \mathbb{N}$	number of eigenvectors
$\mathbf{A} \in \mathbb{R}^{m \times n}$	complete data matrix
$\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$	subset of data available at site $s \in [S]$
$\mathbf{G}_i \in \mathbb{R}^{n \times k}$	eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$ at iteration i
$\mathbf{G} \in \mathbb{R}^{n \times k}$	converged eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$
$\mathbf{G}_i^s \in \mathbb{R}^{n^s \times k}$	partial eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$ at iteration i
$\mathbf{G}^s \in \mathbb{R}^{n^s \times k}$	converged partial eigenvector matrices of $\mathbf{A}^\top\mathbf{A}$.
$\mathbf{H}_i \in \mathbb{R}^{m \times k}$	eigenvector matrix of $\mathbf{A}\mathbf{A}^\top$ at iteration i
$\mathbf{H} \in \mathbb{R}^{m \times k}$	converged eigenvector matrix of $\mathbf{A}\mathbf{A}^\top$
$\mathbf{H}_i^s \in \mathbb{R}^{m \times k}$	partial eigenvector matrix of $\mathbf{A}\mathbf{A}^\top$ at iteration i
$\mathbf{V} \in \mathbb{R}^{r \times k}$	a generic column vector matrix
$\mathbf{U} \in \mathbb{R}^{r \times k}$	an orthonormal matrix with $\text{span}(\mathbf{U}) = \text{span}(\mathbf{V})$

F. Notations

Table I provides an overview of notations which are used throughout the paper.

III. Related Work

A. Centralized, Iterative, Covariance-Free Principal Component Analysis

While classical PCA algorithms rely on computing the covariance matrix $\mathbf{A}^\top\mathbf{A}$ [16], there are several covariance-free approaches to iteratively approximate the top k eigenvalues and eigenvectors [21]. Algorithm 1 summarizes the centralized, iterative, covariance-free PCA algorithm suggested in [12], which will serve as point of departure for our federated approach. First, an initial eigenvector matrix is sampled randomly and orthonormalized (lines 1 to 2). In every iteration i, improved candidate eigenvectors $\mathbf{G}_i$ of $\mathbf{A}^\top\mathbf{A}$ are computed (lines 5 to 8). Once a suitably defined termination criterion is met (convergence, maximal number of iterations, time limit, etc.), the last candidate eigenvectors are returned (line 10).

Algorithm 1: Vertical Subspace Iteration [12].

Input: Data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, number of eigenvectors k.
Output: Eigenvector matrix $\mathbf{G} \in \mathbb{R}^{n \times k}$ of $\mathbf{A}^\top\mathbf{A}$.

1 generate $\mathbf{G}_0 \in \mathbb{R}^{n \times k}$ randomly;
2 $\mathbf{G}_0 \leftarrow \texttt{orthonormalize}(\mathbf{G}_0)$;
3 $i \leftarrow 1$;
4 **while** *termination criterion not met* **do**
5 $\quad \mathbf{H}_i = \mathbf{A}\mathbf{G}_{i-1}$;
6 $\quad \mathbf{H}_i = \texttt{orthonormalize}(\mathbf{H}_i)$;
7 $\quad \mathbf{G}_i = \mathbf{A}^\top\mathbf{H}_i$;
8 $\quad \mathbf{G}_i \leftarrow \texttt{orthonormalize}(\mathbf{G}_i)$;
9 $\quad i \leftarrow i + 1$;
10 $\mathbf{G} \leftarrow \mathbf{G}_i$; **return** $\mathbf{G}$;

To update the candidate eigenvector matrices $\mathbf{G}_i = \mathbf{A}^\top\mathbf{H}_i = \mathbf{A}^\top\mathbf{A}\mathbf{G}_{i-1} \in \mathbb{R}^{n \times k}$ of $\mathbf{A}^\top\mathbf{A}$, the algorithm also computes candidate eigenvector matrices $\mathbf{H}_i = \mathbf{A}\mathbf{G}_{i-1} = \mathbf{A}\mathbf{A}^\top\mathbf{H}_{i-1} \in \mathbb{R}^{m \times k}$ of $\mathbf{A}\mathbf{A}^\top$. Since, in the context of GWAS, $\mathbf{A}\mathbf{A}^\top$ corresponds to the "classical" feature by feature covariance matrix, the algorithm can hence be used for both sample stratification and feature reduction. Our federated version will inherit this property.

B. Federated Principal Component Analysis for Vertically Partitioned Data

Only few algorithms to perform federated computation of PCA on vertically partitioned static data sets have been proposed [8]–[11]. However, none of them is suitable for the GWAS use-case considered in this paper: The algorithms reviewed in [11] are specialised for distributed sensor networks and use gossip protocols and peer-to-peer communication. Therefore, they are not suited for the intended FL architecture in the medical setting. The algorithms presented in [8] and [9] rely on estimating a proxy covariance matrix and consequently do not meet Constraint 2 introduced above. Unlike these approaches, the algorithm described in [10] is covariance-free and suitable for the intended star-like architecture. However, it broadcasts the eigenvectors to all sites in violation of Constraint 1.

C. Federated Matrix Orthonormalization

Matrix orthonormalization is a frequently used technique in many applications, including the solution of linear systems of equations and singular value decomposition. There are three main approaches: Householder reflection, Givens rotation, and the Gram-Schmidt algorithm. In distributed memory systems and grid architectures, tiled Householder reflection is a popular approach [22], [23]. However, those algorithms are often highly specialized to the compute system and rely on shared disk storage. For distributed sensor networks, Gram-Schmidt procedures relying on push-sum have been proposed [24]–[26]. However, these methods require peer-to-peer communication and are hence unsuitable for the intended star-like architecture. In other words, no federated orthonormalization algorithms suitable for our setup are available. Below, we present such an algorithm, which is used as a subroutine in our federated PCA algorithm.

D. Federated Principal Component Analysis for Horizontally Partitioned Data

For completeness, we also provide a short overview of existing federated PCA algorithms for horizontally partitioned data. Here, we selected representatives for conceptual groups of algorithms. There are "single-round" approaches, where the eigenvectors are computed locally and sent to the aggregator [27]. At the aggregator, a global subspace is approximated from the local eigenspaces. The higher the number of transmitted intermediate dimensions, the better the global subspace approximation. In these algorithms, the solution quality hence depends on the number of transmitted

dimensions. Furthermore, iterative schemes have been proposed, where locally computed eigenvectors are sent to the aggregator, which performs an aggregation step and sends the obtained candidate subspace back to the clients [28]–[30]. The candidate subspace is then refined iteratively. Furthermore, there are several schemes for specific applications such as streaming [31], [32]. These approaches assume that an approximation of the entire eigenvectors is possible at the clients, or that the global covariance matrix can be approximated. As we have discussed above, these assumptions do not hold in the intended GWAS use case.

IV. Algorithms

In this section, we present a federated PCA algorithm, which is designed for a star-like architecture, meets the requirements of Constraint 1 and Constraint 2, and is hence suitable for federated GWAS. Our algorithm comes in two variants — with and without orthonormalization of the candidate eigenvectors of $\mathbf{A}^\top\mathbf{A}$. In Section IV-A, we describe our algorithm and prove that the version with orthonormalization is equivalent to centralized vertical subspace iteration algorithm [12], which we have summarized in Algorithm 1 above. In Section IV-B, we present a federated Gram-Schmidt algorithm, which can be used as a subroutine in our federated PCA algorithm to ensure that the eigenvectors of $\mathbf{A}^\top\mathbf{A}$ remain at the local sites. Again, we prove that our federated Gram-Schmidt algorithm is equivalent to the centralized counterpart. In Section IV-C, we analyze the network transmission costs of the proposed algorithms.

A. Federated Vertical Subspace Iteration

Algorithm 2 describes our federated vertical subspace iteration algorithm: Initially, the first partial candidate eigenvector matrices $\mathbf{G}_0^s$ of $\mathbf{A}^\top\mathbf{A}$ are generated randomly and orthonormalized (lines 2 to 4).

Inside the main loop, the candidate eigenvectors $\mathbf{H}_i$ of $\mathbf{A}\mathbf{A}^\top$ are updated at the clients, summed up element-wise and orthonormalized at the aggregator, and then sent back to the clients (lines 9 to 11). Next, the clients update the partial candidate eigenvectors $\mathbf{G}_i^s$ of $\mathbf{A}^\top\mathbf{A}$ (line 13). In the version with orthonormalization, the candidate eigenvectors $\mathbf{G}_i$ of $\mathbf{A}^\top\mathbf{A}$ are now normalized by calling the federated Gram-Schmidt orthonormalization algorithm presented in Section IV-B (line 15). Note that this algorithm ensures that the partial candidate eigenvectors $\mathbf{G}_i^s$ remain at the local sites. Finally, the partial eigenvectors are returned at the clients. In practice, the federated orthonormalization of $\mathbf{G}_i$ (line 15) may be omitted to speed up computation. Note, however, that $\mathbf{H}_i$ is still orthonormalized in every iteration.

Like the original centralized version described in Algorithm 1 above, our algorithm can be run with various termination criteria. In our implementation, we use the convergence criterion

$$\mathrm{diag}(\mathbf{H}_i^\top\mathbf{H}_{i-1}) \geq \mathbf{1}_k - \epsilon \tag{5}$$

using the angle as a global measure as suggested in [33], where $\mathbf{1}_k$ is the k-dimensional vector of ones and ϵ is a small positive number. With this criterion, the algorithm terminates once all candidate eigenvectors of $\mathbf{A}\mathbf{A}^\top$ are asymptotically collinear with respect to the eigenvectors of the previous iteration. Other convergence criteria could be used as drop-in replacements.

Algorithm 2: Federated Vertical Subspace Iteration. (Partial) client-side computations are marked in gray.

Input: Partial data matrices $\mathbf{A}^s \in \mathbb{R}^{m\times n^s}$ at sites $s \in [S]$, number of eigenvectors k.
Output: Partial eigenvector matrices $\mathbf{G}^s \in \mathbb{R}^{n^s\times k}$ of $\mathbf{A}^\top\mathbf{A}$ at sites $s \in [S]$.

1 // Initialize partial candidate eigenvector matrices of $\mathbf{A}^\top\mathbf{A}$.
2 **for** $s \in [S]$ **do** generate $\mathbf{G}_0^s \in \mathbb{R}^{n^s\times k}$ randomly;
3 **if** *use orthonormalization* **then**
    // Use approach described in Algorithm 3.
4 federated-gram-schmidt();
5 // Initialize iteration counter.
6 $i \leftarrow 1$;
7 **while** *termination criterion not met* **do**
8 // Update partial candidate eigenvector matrices of $\mathbf{A}\mathbf{A}^\top$.
9 **for** $s \in [S]$ **do** $\mathbf{H}_i^s \leftarrow \mathbf{A}^s\mathbf{G}_{i-1}^s$;
10 $\mathbf{H}_i \leftarrow \sum_{s=1}^{S} \mathbf{H}_i^s$;
11 $\mathbf{H}_i \leftarrow$ orthonormalize$(\mathbf{H}_i)$;
12 // Update partial candidate eigenvector matrices of $\mathbf{A}^\top\mathbf{A}$.
13 **for** $s \in [S]$ **do** $\mathbf{G}_i^s \leftarrow {\mathbf{A}^s}^\top\mathbf{H}_i$;
14 **if** *use orthonormalization* **then**
        // Use approach described in Algorithm 3.
15 federated-gram-schmidt();
16 // Increment iteration counter.
17 $i \leftarrow i + 1$;
18 **for** $s \in [S]$ **do**
19 $\mathbf{G}^s \leftarrow \mathbf{G}_i^s$;
20 **return** $\mathbf{G}^s$;
21

We now prove that the version of Algorithm 2 with orthonormalization is equivalent to the centralized version described in Algorithm 1. Thus, it inherits its convergence behavior from the centralized version. Details on the convergence behavior of centralized vertical subspace iteration can be found in the original publication [12].

Proposition 1: If orthonormalization is used, centralized and federated vertical subspace iteration are equivalent.

Proof: Let $\mathbf{G}_i$ and $\mathbf{H}_i$ denote the eigenvector matrices maintained by the centralized algorithm described in Algorithm 1 at the end of the main while-loop, and $\mathbf{G}_i^s$ be the sub-matrix of $\mathbf{G}_i$ for the samples available at site s. Moreover, let $\tilde{\mathbf{H}}_i$, $\tilde{\mathbf{G}}_i$, $\tilde{\mathbf{G}}_i^s$, and $\tilde{\mathbf{H}}_i^s$ be the (partial) eigenvector matrices maintained by our federated Algorithm 2 at the end of the main while-loop. We will show by induction on the iterations i that $\mathbf{H}_i = \tilde{\mathbf{H}}_i$ and $\mathbf{G}_i^s = \tilde{\mathbf{G}}_i^s$ for all $s \in [S]$ holds throughout the algorithm, if the same random seeds are used for initialization.

For $i = 0$, we only have to show $\mathbf{G}_0^s = \tilde{\mathbf{G}}_0^s$. This directly follows from Proposition 2 and our assumption that the same

random seeds are used for initialization. For the inductive step, note that, before orthonormalization in line 11, we have $\tilde{\mathbf{H}}_i = \sum_{s=1}^{S} \tilde{\mathbf{H}}_i^s = \sum_{s=1}^{S} \mathbf{A}^s \tilde{\mathbf{G}}_{i-1}^s = \sum_{s=1}^{S} \mathbf{A}^s \mathbf{G}_{i-1}^s = \mathbf{A}\mathbf{G}_{i-1} = \mathbf{H}_i$, where the third equality follows from the inductive assumption. Because of Proposition 2, this identity continues to hold at the end of the main while-loop.

Similarly, after updating in line 13 but before orthonormalization, we have $\tilde{\mathbf{G}}_i^s = \mathbf{A}^{s\top} \tilde{\mathbf{H}}_i = \mathbf{A}^{s\top} \mathbf{H}_i = (\mathbf{A}^\top \mathbf{H}_i)^s = \mathbf{G}_i^s$, where the second equality follows the identity $\mathbf{H}_i = \tilde{\mathbf{H}}_i$ shown above and $(\mathbf{A}^\top \mathbf{H}_i)^s$ denotes the sub-matrix of $\mathbf{A}^\top \mathbf{H}_i$ for the samples available at site s. Again, Proposition 2 ensures that the identity continues to hold after orthonormalization. ■

The omission of the orthonormalization of $\mathbf{G}_i$ (line 4 in Algorithm 2) removes provable convergence to algorithm algorithm 1. However, other formulations of centralized power iteration exist which directly operate on the covariance matrix [27]. In these schemes, instead of splitting the iteration into $\mathbf{H}_i$ update (line 5, algorithm 1) and $\mathbf{G}_i$ update (line 7, algorithm 1), the covariance matrix is computed and $\mathbf{H}_i$ is updated as $\mathbf{H}_i = \mathbf{A}\mathbf{A}^\top \mathbf{H}_{i-1}$ at every iteration. Proposition 1 can be formulated and proven analogously for this version.

B. Federated Gram-Schmidt Algorithm

Here, we describe federated Gram-Schmidt orthonormalization for vertically partitioned column vectors. Previous federated PCA algorithms require the complete eigenvectors to be known at all sites for the orthonormalization procedure. The naïve way of orthonormalizing the eigenvector matrices would be to send them to the aggregator which performs the aggregation and then sends the orthonormal matrices back to the clients. However, in this naïve scheme, the transmission cost scales with the number of variables (individuals in GWAS) and all eigenvectors are known to the aggregator.

To address these two problems, we suggest a federated Gram-Schmidt orthonormalization algorithm, summarized in Algorithm 3. The algorithm exploits the fact that the computations of the squared norms n_i and of the residuals r_{ij} can be decomposed into independent computations of summands n_i^s and r_{ij}^s computable at the local sites $s \in [S]$. The clients compute the local summands and send them to the aggregator, where the squared norm of the first orthogonal vector is computed and sent to the clients (lines 2 to 2). Subsequently, the remaining $k-1$ vectors are orthogonalized. For the i^{th} vector $\mathbf{v}_i$, the algorithm computes the residuals r_{ij} w. r. t. all already computed orthogonal vectors $\mathbf{u}_j$, using the fact that the corresponding squared norms n_j are already available (lines 8 to 10). The residuals are aggregated by the central server (lines 12 to 13). Next, $\mathbf{v}_i$ is orthogonalized at the clients, the local norms are computed (lines 15 to 17), and the squared norm of the resulting orthogonal vector $\mathbf{u}_i$ is computed at the aggregator and sent back to the clients (line 19). After orthogonalization, all orthogonal vectors are scaled to unit norm at the clients (lines 21 to 23).

Proposition 2: Centralized and federated Gram-Schmidt orthonormalization are equivalent.

Algorithm 3: Federated Gram-Schmidt. Client-side computations are marked in gray.

Input: Data matrices $\mathbf{V}_s$ at sites $s \in [S]$.
Output: Orthonormalized data matrices $\mathbf{U}_s$ at sites $s \in [S]$.

1 // Compute squared norm of first orthogonal vector.
2 **for** $s \in [S]$ **do**
3 $\quad \mathbf{u}_1^s \leftarrow \mathbf{v}_1^s$;
4 $\quad n_1^s \leftarrow \mathbf{u}_1^{s\top} \mathbf{u}_1^s$;
5 $n_1 \leftarrow \sum_{s=1}^{S} n_1^s$;
// Orthogonalize all subsequent vectors.
6 **for** $i \in [k] \setminus \{1\}$ **do**
7 $\quad$ // Compute client residuals for vector being orthogonalized.
8 $\quad$ **for** $s \in [S]$ **do**
9 $\quad\quad$ **for** $j \in [i-1]$ **do**
10 $\quad\quad\quad r_{ij}^s \leftarrow \mathbf{u}_j^{s\top} \mathbf{v}_i^s / n_j$;
11 $\quad$ // Compute global residuals for vector being orthogonalized.
12 $\quad$ **for** $j \in [i-1]$ **do**
13 $\quad\quad r_{ij} \leftarrow \sum_{s=1}^{S} r_{ij}^s$;
14 $\quad$ // Orthogonalize the vector and compute squared norm.
15 $\quad$ **for** $s \in [S]$ **do**
16 $\quad\quad \mathbf{u}_i^s \leftarrow \mathbf{v}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s$;
17 $\quad\quad n_i^s \leftarrow \mathbf{u}_i^{s\top} \mathbf{u}_i^s$
18 $\quad$ // Compute squared norm of orthogonalized vector.
19 $\quad n_i \leftarrow \sum_{s=1}^{S} n_i^s$;
20 // After orthogonalization, scale all k vectors to unit norm.
21 **for** $s \in [S]$ **do**
22 $\quad$ **for** $i \in [k]$ **do** $\mathbf{u}_i^s \leftarrow \frac{1}{\sqrt{n_i}} \cdot \mathbf{u}_i^s$;
23 $\quad \mathbf{U}^s \leftarrow [\mathbf{u}_1^s \dots \mathbf{u}_s^k]$; **return** $\mathbf{U}^s$;
24

Proof: Let $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_k]$ be the matrix that should be orthonormalized, $\mathbf{v}_i^s$ be the restriction of the i^{th} columns vector to the samples available at side s, and $\mathbf{u}_i^s$ be the restriction of the i^{th} orthogonal vector computed by the centralized Gram-Schmidt algorithm before normalization to the samples available at site s. Moreover, let n_i and $r_{i,j}$ be the centrally computed norms and residuals, and $\tilde{n}_i$, $\tilde{r}_{i,j}$, and $\tilde{\mathbf{u}}_i^s$ be the locally computed norms, residuals, and partial orthogonal vectors before normalization. We show by induction on i that $n_i = \tilde{n}_i$, $r_{ij} = \tilde{r}_{ij}$, and $\mathbf{u}_i^s = \tilde{\mathbf{u}}_i^s$ holds for all $i \in [k]$ and all $j \in [i-1]$. This implies the proposition.

For $i = 1$, we have $\mathbf{u}_1^s = \mathbf{v}_1^s = \tilde{\mathbf{u}}_1^s$ and $n_1 = \mathbf{u}_1^\top \mathbf{u}_1 = \sum_{s=1}^{S} \mathbf{u}_1^{s\top} \mathbf{u}_1^s = \sum_{s=1}^{S} \tilde{\mathbf{u}}_1^{s\top} \tilde{\mathbf{u}}_1^s = \tilde{n}_1$. For the inductive step, note that $r_{ij} = \mathbf{u}_j{}^\top \mathbf{v}_i / n_j = \sum_{s=1}^{S} \mathbf{u}_j^{s\top} \mathbf{v}_i^s / n_j = \sum_{s=1}^{S} \tilde{\mathbf{u}}_j^{s\top} \mathbf{v}_i^s / \tilde{n}_j = \tilde{r}_{ij}$, where the third identity follows from the inductive assumption. Moreover, we have $\mathbf{u}_i^s = \mathbf{v}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s = \mathbf{v}_i^s - \sum_{j=1}^{i-1} \tilde{r}_{ij} \cdot \tilde{\mathbf{u}}_j^s = \tilde{\mathbf{u}}_i^s$, where the second identity follows from the inductive assumption and the identities $r_{ij} = \tilde{r}_{ij}$ established before. We hence obtain

$n_i = \mathbf{u}_i^\top \mathbf{u}_i = \sum_{s=1}^{S} \mathbf{u}_i^{s\top} \mathbf{u}_i^s = \sum_{s=1}^{S} \tilde{\mathbf{u}}_i^{s\top} \tilde{\mathbf{u}}_i^s = \tilde{n}_i$, which completes the proof. ■

C. Network Transmission Costs

The main bottleneck in FL is the amount of data transmitted between the different sites and the number of network communications. The following Proposition 3 specifies these quantities for our federated PCA algorithm. Recall that S, k, m, and n denote, respectively, the numbers of sites, eigenvectors, features, and samples.

Proposition 3: Let $\mathcal{D}$ be the total amount of data transmitted by our federated PCA algorithm, $\mathcal{N}$ be the total number of network communications, and I be the total number of iterations of the main while-loop. Then the following statements hold:

- If the $\mathbf{G}_i$ matrices are not orthonormalized, it holds that $\mathcal{D} = \mathcal{O}(I \cdot S \cdot k \cdot m)$ and $\mathcal{N} = \mathcal{O}(I \cdot S)$.
- If federated Gram-Schmidt orthonormalization is used, it holds that $\mathcal{D} = \mathcal{O}(I \cdot (S \cdot k \cdot m + k^2))$ and $\mathcal{N} = \mathcal{O}(I \cdot S \cdot k)$.

Proof: In each iteration i of our federated vertical subspace iteration algorithm, the matrices $\mathbf{H}_i^s \in \mathbb{R}^{m \times k}$ have to be sent from the clients to the aggregator and the matrix $\mathbf{H}_i \in \mathbb{R}^{m \times k}$ has to be sent back to the clients. In iteration i, the amount of transmitted data and the number of communications due to $\mathbf{H}_i$ is hence $\mathcal{O}(S \cdot k \cdot m)$ and $\mathcal{O}(S)$, respectively. For orthonormalizing the eigenvector matrices $\mathbf{G}_i \in \mathbb{R}^{n \times k}$, we need to transmit a data volume of $\mathcal{O}(S \cdot k^2)$ and the number of communications increases to $\mathcal{O}(S \cdot k)$. By summing over the iterations i, this yields the statement of the proposition. ■

If our federated Gram-Schmidt algorithm is used, the overall volume of transmitted data is hence independent of the number of samples n. This is especially important in the intended GWAS setting, since here we can achieve $n \gg m$ by pre-filtering the SNPs (i. e., features) before carrying out the PCA [18], [34]. Moreover, k is small (typically, $k = 10$ is used for GWAS PCA), which implies that the additional factor k in the complexities of $\mathcal{D}$ and $\mathcal{N}$ can be neglected. Therefore, using the suggested scheme is preferable over sending the eigenvectors to the aggregator for orthonormalization both in terms of privacy and expected transmission cost.

V. Web Service

Researchers from the biomedical sciences performing GWAS typically rely on ready-made software such as PLINK [13]. Consequently, a user-friendly tool is required in order to make federated PCA available to these researchers in order to facilitate adoption in practice.

To render the federated PCA algorithm presented in Section IV accessible to the biomedical community, we have developed a web-based user interface available at https://anonymous.4open.science/r/hyfed-pca-22F6. The implementation is based on HyFed [35], a Python package for privacy-preserving federated web services. To the best of our knowledge, our web tool is the first ready-to-use privacy-preserving implementation of a federated PCA algorithm. It consists of four components:

- A client software which runs at the local sites and computes all parameters using local data.
- An aggregation software which runs on a central server and aggregated the local (noisy) parameters.
- A compensator module which aggregates the noise such that the true local parameters are hidden from the aggregation software.
- A web interface which can be used to set up the study. This includes the generation of tokens for all participants to ensure that only authenticated users participate in the study.

To protect the local data from reconstruction attacks at the aggregation server, the HyFed backend provides an SMPC-like parameter obfuscation scheme, where noise is added to the intermediate parameters and subtracted from the global parameters. HyFed assumes an honest-but-curious risk model, i. e., assumes that all participants (aggregator, compensator, as well as all clients) strictly adhere to the protocol but try to extract as much information as possible from legally acquired data. For more details, we refer to the original publication [35].

VI. Empirical Evaluation

A. Test Datasets

To evaluate our federated PCA algorithm, we used three publicly available datasets: chromosome 1 and chromosome 2 from a genetic dataset from the 1000 Genomes Project [36], as well as the MNIST database of handwritten digits [37] (Table II). The two genetic data sets contain data for 2502 patients (samples) and 100 000 SNPs (features), after applying standard pre-processing steps (MAF filtering, LD pruning, SNP sub-sampling). MNIST contains 60 000 grayscale images of handwritten numerals (samples), each of which has 784 pixels (features). To the best of our knowledge, publicly available genetic data sets with large numbers of patients are not readily available. Therefore, we decided to use MNIST in addition to the genetic datasets, because the higher number of samples allowed us to evaluate our algorithm in terms of scalability. Moreover, although motivated by federated GWAS, our federated PCA algorithm is actually generically applicable. The experiments on MNIST demonstrate its usefulness for a more general audience. The MNIST data set is available at http://yann.lecun.com/exdb/mnist/, the genetic data can be obtained from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/.

TABLE II
Datasets used in the study.

Dataset	Samples	Features
MNIST	60 000	784
1000 Genomes – Chrom. 1	2502	100 000
1000 Genomes – Chrom. 2	2502	100 000

B. Compared Methods

We tested two variants of our federated PCA algorithm detailed in Algorithm 2: NO-GS, which omits Gram-Schmidt

orthonormalization, and FED-GS, which uses federated Gram-Schmidt orthonormalization as described in Algorithm 3. We compared NO-GS and FED-GS to the federated PCA algorithm suggested by Guo and colleagues [10] (called GUO in the sequel), the only available federated PCA algorithm for vertically partitioned data suitable for a star-like architecture. However, GUO shares the sample eigenvectors with the aggregator, which should be avoided in federated GWAS as emphasized earlier. For all compared methods, we set the convergence criterion in eq. (5) to $\epsilon = 10^{-9}$, which corresponds to a change of the angle between two consecutive eigenvectors updates of about 0.0026 degrees. Note that this angle does not equal the angle w. r. t. centrally computed eigenvectors, which we used as a test metric for measuring the quality of the compared methods (cf. next paragraph).

C. Test Metrics

For measuring the quality of the compared methods, we computed the angles between the eigenvectors obtained from a reference implementation of a centralized PCA and their counterparts computed in a federated fashion. An angle of 0 between two eigenvectors of the same rank is the desired result. As a reference, we chose the version implemented in `scipy.sparse.linalg`, which internally interfaces LAPACK. The amount of transmitted data is estimated by calculating the number of transmitted floats and multiplying it by a factor of 4 bytes (single precision IEEE 754). We choose this metric to remain agnostic with respect to the transmission protocol. Times measures are wall clock times using Python's `time` module.

D. Implementation, Availability, and Hardware Specifications

All methods except the web interface are written in Python, using mainly, but not exclusively `numpy` and `scipy`. They are available online at https://anonymous.4open.science/r/federated-pca-simulation-5C79/. The tool is implemented in Python (Django) and Angular, and is available at https://anonymous.4open.science/r/hyfed-pca-22F6. The simulation tests were run on a compute server with 48 CPUs and 502 GB available RAM in a total of 319 minutes, however the tests can also be run on a standard laptop (8 CPUs, 16GB RAM) . The tool was tested using a commodity laptop with a high-speed Ethernet connection (100Mbit/s) running the clients and the remote server running the aggregation software, meaning the parameters had to be transmitted over the network.

E. Results of the Experiments

a) Convergence Behavior: To test the convergence behavior of the compared federated algorithms, we split the three data sets into 5 equally sized chunks of samples. For each of the first ten eigenvectors (i. e., the number of eigenvectors typically used for sample stratification in GWAS) and each compared algorithm, we then recorded the mean angles at each iteration w. r. t. the fully converged reference averaged across 20 runs of the algorithm with random initialization of the eigenvector guess and the data split into different randomized chunks.

Figure 3 shows the results of the experiments. Note that, unlike the versions FED-GS and NO-GS of our algorithm, the competitor GUO computes the eigenvectors sequentially (i. e., eigenvector k has to converge before starting the computation of the eigenvector $k+1$), which means that, for all but the first eigenvector, the plots for GUO start with a horizontal line. The most important result is that, for all algorithms, the eigenvectors perfectly converge to the reference after a sufficient number of iterations. The two versions of FED-GS and NO-GS of our algorithm show essentially the same convergence behavior. They both clearly outperform the competitor GUO in terms of convergence speed.

b) Scalability: For the scalability tests, we randomly sampled collections containing $p\,\%$ of the MNIST images, for $p \in \{10, 20, \ldots, 90\}$, and ran the methods on the collections until convergence or until a maximum of 500 iterations was reached. For NO-GS and FED-GS the total number of iterations was limited and for GUO the number of iterations per eigenvector. We repeated this 10 times for each fraction of the data, and recorded the net volumes of transmitted data, as well as the numbers of iterations at convergence.

Figure 4 shows the volume of transmitted data in MB depending on the size of the data subset. For GUO the volume of transmitted data increases linearly with increasing number of samples. This is not the case for FED-GS and NO-GS, where the amount of transmitted data is constant with increasing sample size. This is due to the federated Gram-Schmidt orthonormalization since the eigenvectors of the sample by sample covariance matrix do not need to be transmitted.

Naturally, federated QR orthogonalization introduces more communication steps. However, this would also be the case would one implement GUO with the additional constraint to not broadcast the eigenvectors. When imposing this constraint, the number of communication rounds per iteration increases by a factor of $2k$ where k is the number of eigenvectors.

c) Performance of the Web Tool: Finally, we tested the performance of our web interface. For this, we recorded the run times and transmission costs for computing the first eight eigenvectors for the three test data sets, when using either of the two variants FED-GS and NO-GS of our federated PCA algorithm. The convergence threshold was again set to $\epsilon = 10^{-9}$, and the data was split into 2 equal chunks with the clients running on the same laptop. We chose a slightly smaller number in these experiments, because the explanatory power decreases with higher rank, but the convergence time increases due to the smaller eigengap. Table IV shows the results. Using NO-GS, the tool needed fewer than 22 minutes to compute the eigenvectors in all three instances. Using FED-GS increased the run time to about 2 hours due to the increased network communication (cf. Proposition 3) and the fact that the tool uses a https-based transmission protocol which relies on polling. These run times are absolutely reasonable, given that collecting GWAS data often takes years. The transmission

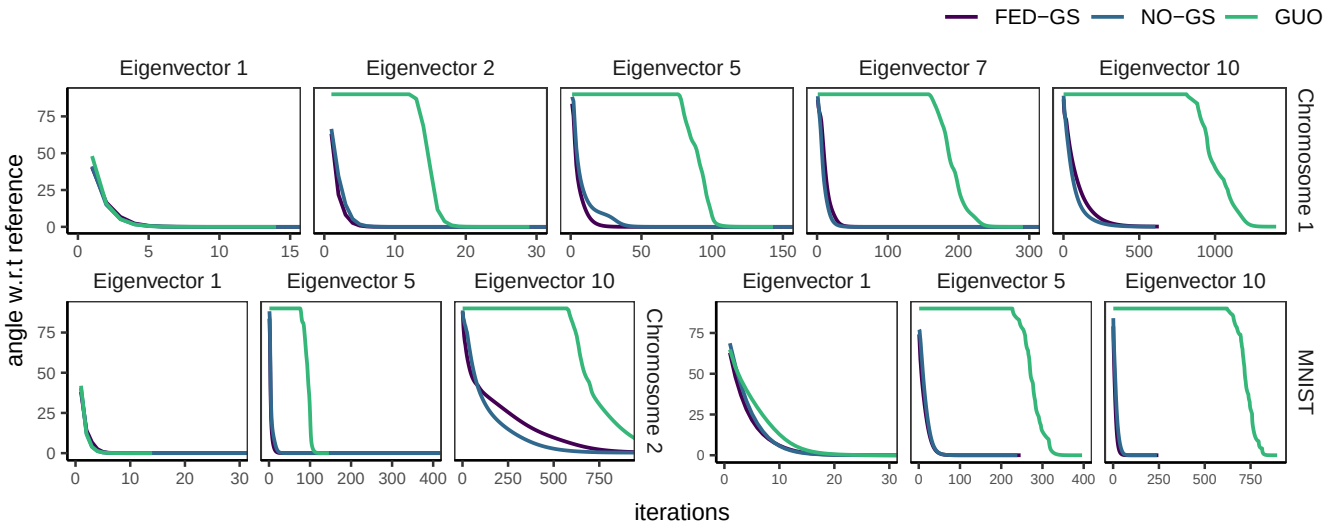


Fig. 3. Angles between selected reference eigenvectors and the federated eigenvectors on chromosome 1 and 2, as well as for MNIST. The omitted eigenvectors show similar behaviors.

TABLE IV
RUNTIMES OF THE WEB TOOL (IN min) ON THE THREE TEST DATASETS.

Algorithm	MNIST		Chrom. 1		Chrom. 2	
	clear	noisy	clear	noisy	clear	noisy
NO-GS	9	10	18	23	11	13
FED-GS	125	125	134	157	148	104

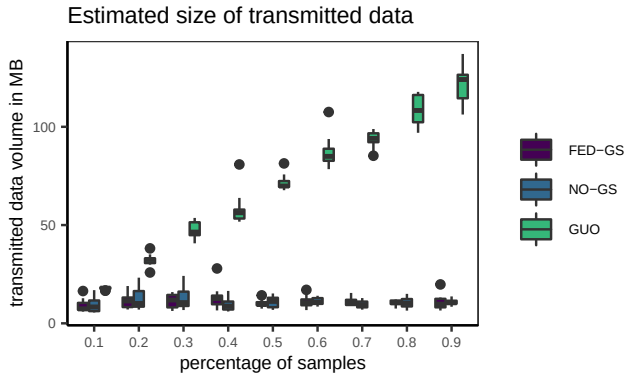


Fig. 4. Number of samples versus estimated amount of transmitted data.

TABLE III
TRANSMISSION COST OF THE WEB TOOL (IN GB) ON THE TEST DATASETS.

Algorithm	MNIST		Chrom. 1		Chrom. 2	
	clear	noisy	clear	noisy	clear	noisy
NO-GS	0.048	0.082	6.33	10.59	3.8	5.28
FED-GS	0.055	0.089	5.56	10.93	6.16	7.22

costs are summarized in table III. Since GWAS PCA is used in cross-silo federated learning, the relatively high volume of transmitted data is acceptable, given the feasible compute time. Independently, we conclude that the number of communication steps is a crucial factor to optimize in iterative federated learning.

VII. CONCLUSIONS AND OUTLOOK

In this paper, we presented an improved federated PCA algorithm for vertically partitioned data. Although our algorithm is primarily designed to meet the requirements of population stratification in federated GWAS, it can be used for any vertically or horizontally partitioned data. We proved that our algorithm is equivalent to a state-of-the-art centralized PCA algorithm and showed empirically that it indeed converges to the centrally computed solutions. The key advantage of our algorithm is that, unlike in existing federated PCA algorithms, the sample eigenvectors remain at the local sites, due to the use of fully federated Gram-Schmidt orthonormalization which improves the privacy of the algorithm. We also provide a user-friendly web interface to promote FL in the less technically inclined GWAS community. In future work, we intend to further decrease the amount of transmitted data and the number of communication rounds by only updating those eigenvectors that have not converged yet or offering premature termination to avoid computing eigenvectors explaining little of the overall variance.

ACKNOWLEDGMENTS

Acknowledgment placeholder
Acknowledgment placeholder
Acknowledgment placeholder
Acknowledgment placeholder
Acknowledgment placeholder
Acknowledgment placeholder

References

[1] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, p. 619–640, Feb 2021.

[2] P. M. Visscher, N. R. Wray, Q. Zhang, P. Sklar, M. I. McCarthy, M. A. Brown, and J. Yang, "10 Years of GWAS Discovery: Biology, Function, and Translation," *American Journal of Human Genetics*, vol. 101, no. 1, pp. 5–22, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ajhg.2017.06.005

[3] V. Tam, N. Patel, M. Turcotte, Y. Bossé, G. Paré, and D. Meyre, "Benefits and limitations of genome-wide association studies," *Nature Reviews Genetics*, vol. 20, no. 8, pp. 467–484, 2019. [Online]. Available: http://dx.doi.org/10.1038/s41576-019-0127-1

[4] R. Nasirigerdeh, R. Torkzadehmahani, J. Matschinske, T. Frisch, M. List, J. Späth, S. Weiss, U. Völker, N. K. Wenke, T. Kacprowski, and J. Baumbach, "splink: A federated, privacy-preserving tool as a robust alternative to meta-analysis in genome-wide association studies," *bioRxiv*, 2020.

[5] H. Cho, D. J. Wu, and B. Berger, "Secure genome-wide association analysis using multiparty computation," *Nature Biotechnology*, vol. 36, no. 6, pp. 547–551, 2018. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/29734293http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5990440

[6] A. L. Price, N. J. Patterson, R. M. Plenge, M. E. Weinblatt, N. A. Shadick, and D. Reich, "Principal components analysis corrects for stratification in genome-wide association studies," *Nature Genetics*, vol. 38, no. 8, pp. 904–909, 2006.

[7] K. J. Galinsky, G. Bhatia, P.-R. Loh, S. Georgiev, S. Mukherjee, N. J. Patterson, and A. L. Price, "Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia," *The American Journal of Human Genetics*, vol. 98, no. 3, pp. 456–472, mar 2016. [Online]. Available: http://dx.doi.org/10.1016/j.ajhg.2015.12.022https://linkinghub.elsevier.com/retrieve/pii/S0002929716000033

[8] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson, "Distributed Clustering Using Collective Principal Component Analysis," *Knowledge and Information Systems*, 2001.

[9] H. Qi, T. W. Wang, and J. D. Birdwell, "Global principal component analysis for dimensionality reduction in distributed data mining," in *Statistical Data Mining and Knowledge Discovery*. Chapman and Hall/CRC, jul 2003, pp. 323–338. [Online]. Available: http://www.crcnetbase.com/doi/10.1201/9780203497159.ch19

[10] Y. F. Guo, X. Lin, Z. Teng, X. Xue, and J. Fan, "A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data," *Pattern Recognition*, vol. 45, no. 3, pp. 1211–1219, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2011.09.002

[11] S. X. Wu, H. T. Wai, L. Li, and A. Scaglione, "A Review of Distributed Algorithms for Principal Component Analysis," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1321–1340, 2018.

[12] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, p. 217–288, Jan 2011.

[13] C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee, "Second-generation PLINK: Rising to the challenge of larger and richer datasets," *GigaScience*, vol. 4, no. 1, pp. 1–16, 2015.

[14] A. Steed and M. Fradinho Duarte de Oliveira, "More than two," *Network Graphics*, pp. 125–168, 12 2010.

[15] M. Á. Rodríguez, A. Fernández, A. Peregrín, and F. Herrera, *A Review of Distributed Data Models for Learning*. Cham: Springer International Publishing, 2017.

[16] I. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 2002. [Online]. Available: https://doi.org/10.1007/b98835

[17] R. Nasirigerdeh, R. Torkzadehmahani, J. Baumbach, and D. B. Blumenthal, "On the privacy of federated pipelines," in *SIGIR 2021*. New York: ACM, 2021, p. 5.

[18] Y. Li, J. Byun, G. Cai, X. Xiao, Y. Han, O. Cornelis, J. E. Dinulos, J. Dennis, D. Easton, I. Gorlov, M. F. Seldin, and C. I. Amos, "FastPop: A rapid principal component derived method to infer intercontinental ancestry using genetic data," *BMC Bioinformatics*, vol. 17, no. 1, pp. 1–8, 2016. [Online]. Available: http://dx.doi.org/10.1186/s12859-016-0965-1

[19] H. G. Gauch, S. Qian, H. P. Piepho, L. Zhou, and R. Chen, "Consequences of PCA graphs, SNP codings, and PCA variants for elucidating population structure," *PLoS ONE*, vol. 14, no. 6, pp. 1–26, 2019.

[20] R. A. Beezer, "A first course in linear algebra," http://linear.ups.edu/fcla/section-O.html, 2016, [Online book; accessed 2020-11-18].

[21] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, ser. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan 2011. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611970739

[22] B. Hadri, H. Ltaief, E. Agullo, and J. Dongarra, "Tile qr factorization with parallel panel processing for multicore architectures," in *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, 2010, pp. 1–10.

[23] M. Hoemmen, "A communication-avoiding, hybrid-parallel, rank-revealing orthogonalization method," in *2011 IEEE International Parallel Distributed Processing Symposium*, 2011, pp. 966–977.

[24] O. Sluvciak, H. Straková, M. Rupp, and W. Gansterer, "Distributed Gram-Schmidt orthogonalization with simultaneous elements refinement," *Eurasip Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–13, 2016. [Online]. Available: http://dx.doi.org/10.1186/s13634-016-0322-6

[25] O. Sluciak, H. Strakova, M. Rupp, and W. N. Gansterer, "Distributed Gram-Schmidt orthogonalization based on dynamic consensus," *Conference Record - Asilomar Conference on Signals, Systems and Computers*, pp. 1207–1211, 2012.

[26] H. Straková, W. N. Gansterer, and T. Zemen, "Distributed qr factorization based on randomized algorithms," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 235–244.

[27] M. F. Balcan, V. Kanchanapally, Y. Liang, and D. Woodruff, "Improved distributed principal component analysis," *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3113–3121, 2014. [Online]. Available: http://arxiv.org/abs/1408.5823

[28] M. F. Balcan, S. S. Du, Y. Wang, and A. W. Yu, "An improved gap-dependency analysis of the noisy power method," *Journal of Machine Learning Research*, vol. 49, no. June, pp. 284–309, 2016. [Online]. Available: http://arxiv.org/abs/1602.07046

[29] X. Chen, J. D. Lee, H. Li, and Y. Yang, "Distributed estimation for principal component analysis: a gap-free approach," *CoRR*, vol. abs/2004.02336, 2020. [Online]. Available: https://arxiv.org/abs/2004.02336

[30] H. Imtiaz and A. D. Sarwate, "Differentially private distributed principal component analysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Apr 2018, p. 2206–2210. [Online]. Available: https://ieeexplore.ieee.org/document/8462519/

[31] A. Sanchez-Fernandez, M. Fuente, and G. Sainz-Palmero, "Fault detection in wastewater treatment plants using distributed pca methods," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, Sep 2015, p. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7301504/

[32] A. Grammenos, R. Mendoza Smith, J. Crowcroft, and C. Mascolo, "Federated principal component analysis," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6453–6464. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/47a658229eb2368a99f1d032c8848542-Paper.pdf

[33] Q. Lei, K. Zhong, and I. S. Dhillon, "Coordinate-wise power method," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016, p. 2064–2072. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/8b40665547 30ddfaa0266346bdc1b202-Paper.pdf

[34] E. R. Londin, M. A. Keller, C. Maista, G. Smith, L. A. Mamounas, R. Zhang, S. J. Madore, K. Gwinn, and R. A. Corriveau, "Coaims: A cost-effective panel of ancestry informative markers for determining continental origins," *PLoS One*, vol. 5, p. e13443, Oct 2010.

[35] R. Nasirigerdeh, R. Torkzadehmahani, J. O. Matschinske, J. Baumbach, D. Rueckert, and G. Kaissis, "HyFed: A hybrid federated framework for privacy-preserving machine learning," *CoRR*, vol. abs/2105.10545, 2021. [Online]. Available: https://arxiv.org/abs/2105.10545

[36] The 1000 Genomes Project Consortium., Corresponding authors., Auton, A. et al., "A global reference for human genetic variation," *Nature*, vol. 526, no. 7571, pp. 68–74, 2015.

[37] Y. LeCun, C. Cortes, and C. J. Burges, "MNNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 2005, [Online; accessed 27-02-2020].

Chapter 4

Manuscript 3

Federated Singular Value Decomposition for High Dimensional Data

Federated singular value decomposition for high dimensional data

Anne Hartebrodt[1*], Richard Röttger[1†] and David B. Blumenthal[2†]

[1*]Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, Odense, 5230, , Denmark.
[2]Department Artificial Intelligence in Biomedical Engineering (AIBE), Friedrich-Alexander University Erlangen-Nürnberg (FAU), Konrad-Zuse-Str. 3/5, 91052 Erlangen, Germany.
[a]ORCID:0000-0002-9172-3137.
[b]ORCID:0000-0003-4490-5947.
[c]ORCID:0000-0001-8651-750X.

*Corresponding author(s). E-mail(s): hartebrodt@imada.sdu.dk;
Contributing authors: roettger@imada.sdu.dk;
david.b.blumenthal@fau.de;
†Joint last authors

Abstract

Federated learning (FL) is emerging as a privacy-aware alternative to classical cloud-based machine learning. In FL, the sensitive data remains in data silos and only aggregated parameters are exchanged. Hospitals and research institutions which are not willing to share their data can join a federated study without breaching confidentiality. In addition to the extreme sensitivity of biomedical data, the high dimensionality poses a challenge in the context of federated genome-wide association studies (GWAS). In this article, we present a federated singular value decomposition (SVD) algorithm, suitable for the privacy-related and computational requirements of GWAS. Notably, the algorithm has a transmission cost independent of the number of samples and is only weakly dependent on the number of features, because the singular vectors associated with the samples are never exchanged and the vectors associated with the features only for a fixed number of iterations. Although motivated by GWAS, the algorithm is generically applicable for both horizontally and vertically partitioned data.

Keywords: Singular value decomposition, Federated learning, Principal component analysis, Genome-wide association studies

1 Introduction

Federated learning (FL) has recently gained attention as a privacy-aware alternative to centralized computation. Unlike in centralized machine learning where the data is consolidated at a central server and a model is calculated on the combined data, in FL, the data remains at the data owners machine (Mothukuri et al, 2021). Instead of the data, only model parameters are sent to the (untrusted) aggregator which combines the local models into a global model. No raw data is exchanged in FL. See Figure 1 for a schematic comparison of centralized (cloud) learning and FL. In the cloud-based approach, data contributors send their private data to a central server where the model is computed and thereby lose agency over it.

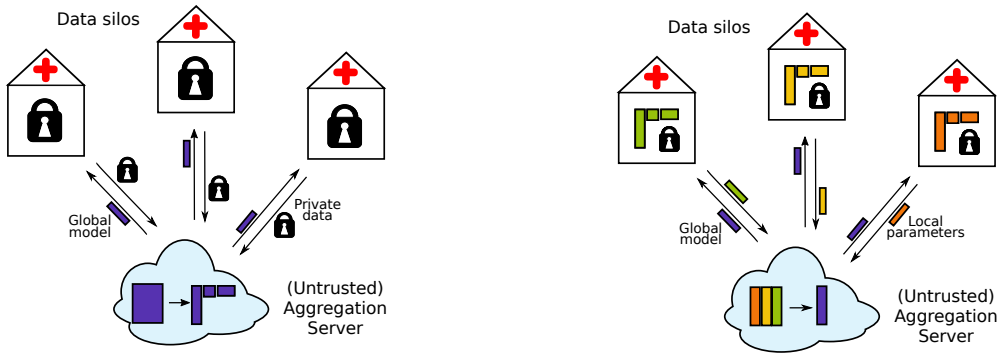


Fig. 1 Schematic comparison of traditional cloud base approaches (left) and federated learning (right).

FL is subdivided in cross-device and cross-silo FL. Cross-device FL assumes a high number of devices, such as mobile phones or sensors with limited compute power connected in a dynamic fashion, meaning clients are expected join and drop out during the learning process. Cross-silo FL has a lower number of participants which hold a larger amount of data, have higher compute power and are connected in a more static fashion. Clients are not expected to join and drop out during the learning process randomly (Kairouz et al, 2021).

An attractive application case for cross-silo FL are genome-wide association studies (GWAS), which investigate the relationship of genetic variation with phenotypic traits on large cohorts (Visscher et al, 2017; Tam et al, 2019). Since genetic data are extremely sensitive, data holders cannot make it publicly available. The practical feasibility of using FL for GWAS has been demonstrated recently (Nasirigerdeh et al, 2020; Cho et al, 2018).

Since GWAS are often done on populations of mixed ancestry, cryptic population confounders should be controlled for before associating the genetic variants to the phenotypic trait of interest. The standard way for doing this is to compute the leading eigenvectors of the sample covariance matrix via principal component analysis (PCA), and to include these eigenvectors as

confounding variables to the models used for the association tests (Price et al, 2006; Galinsky et al, 2016).

For federated GWAS, a PCA algorithm for vertically partitioned data is required for computing the eigenvectors (see Section 2 for a detailed explanation). Although a few such algorithms are available (Kargupta et al, 2001; Qi et al, 2003; Guo et al, 2012; Wu et al, 2018), none of them is suitable for federated GWAS. More precisely, the algorithms reviewed by Wu et al (2018) use client-to-client communication and are therefore unsuitable for the star-like FL architectures used in GWAS, where relatively few data holders collaborate in a static setting. The algorithms presented by Kargupta et al (2001) and Qi et al (2003) rely on estimating a proxy covariance matrix and hence do not scale to large GWAS datasets, which often contain genetic variation data for hundreds of thousands of individuals. One of the few covariance free PCA algorithm suitable for a star-like architecture has been presented by Guo et al (2012). However, this algorithm broadcasts the complete first $k-1$ sample eigenvectors to the aggregator, which constitutes a privacy leakage that should be avoided in federated GWAS (Nasirigerdeh et al, 2021). Cho et al (2018) present a secure multiparty protocol for GWAS which includes PCA relying on householder reflections. The protocol includes three external parties and potential physical shipping of data. The setup is fundamentally different: the data holders are individuals who only have access to one record. They create secret shares which are processed by two computing parties.

In previous algorithms, including algorithms designed for horizontally partitioned data, such as described by Balcan et al (2014), the exchanged parameters scale with the number of genetic variants (features) in the data set as the feature eigenvectors are exchanged. At the scale of GWAS with several million genetic variants this is another challenge for the existing algorithms. Furthermore, due to the iterative nature of the algorithm, the process is prone to information leakage, a problem previously not investigated. More precisely, the feature eigenvector updates exchanged during the learning process can be used to compute the feature-covariance matrix given a sufficient number of iterations. This makes the algorithm equivalent with algorithms exchanging the entire covariance matrix in terms of disclosed information. The feature covariance matrix is a summary statistic over all samples, but due to its size contains a high amount of information and can be used to generate realistically looking samples. Therefore, the communication of the entire feature eigenvectors should also be avoided as far as possible.

Extrapolating from the shortcomings of existing approaches, we can state that, for federated GWAS, a PCA algorithm for vertically partitioned data is required that combines the following properties:

- The algorithm should be suitable for a star-like FL architecture, i. e., require only client-to-aggregator but no client-to-client communication.
- The algorithm should not rely on computing or approximating the covariance matrix.
- The algorithm should be communication efficient.

- The algorithm should avoid the communication of the sample eigenvectors and reduce the communication of the feature eigenvectors.

In this paper, we present the first algorithm that combines all of these desirable properties and can hence be used for federated GWAS (and all other applications where these properties are required). We prove that our algorithm is equivalent to centralized vertical subspace iteration (Halko et al, 2011) — a state-of-the-art centralized, covariance-free SVD algorithm — and therefore generically applicable to any kind of data. Thereby, we show that the notion of "horizontally" and "vertically" partitioned data are irrelevant for SVD. Furthermore, we apply two strategies to make the algorithm more communication efficient, both in terms of communication rounds and transmitted data volume. More specifically, we employ approximate PCA (Balcan et al, 2014) and randomized PCA (Halko et al, 2011). We show in an empirical evaluation that the eigenvectors computed by our approaches converge to the centrally computed eigenvectors after sufficiently many iterations. In sum, the article contains the following main contributions:

- We present the first federated PCA algorithm for vertically partitioned data which meets the requirements that apply in federated GWAS settings.
- We prove that our algorithm is equivalent to centralized power iteration and show that it exhibits an excellent convergence behavior in practice.
- This algorithm is generically applicable for federated singular value decomposition on both "horizontally" and "vertically" partitioned data.

This article is an extended and consolidated version of a previous conference publication (Hartebrodt et al, 2021) with the following additional contributions: a demonstration how iterative leakage can pose a problem for federated power iteration; a further reduction in transmission cost, and increase in privacy, due to the use of randomized PCA; a data dependent speedup due to the use of approximate PCA. The remainder of this paper is organized as follows: In Section 2, we introduce concepts and notations that are used throughout the paper. In Section 3, we discuss related work. In Section 4, we describe the proposed algorithms. We then describe how to extract the covariance matrix from the updates in Section 5. In Section 6, we report the results of the experiments. Section 7 concludes the paper.

2 Preliminaries

2.1 Federated learning and employed data model

Typically, a star-like client-aggregator architecture is used in biomedical federated solutions (Steed and Fradinho Duarte de Oliveira, 2010; Nasirigerdeh et al, 2020), with the data holders acting as clients. The data sets at the client sites will be called *local data sets* and the parameters or models learned using this data will be called *local parameters* or *local models*, while the final aggregated model will be called *pooled model*. The optimal result of the pooled model is achieved when it equals the result of the conventional model calculated on all data, which we call the *global model*.

In federated settings, the data can be distributed in several ways. Either the clients observe a full set of variables for a subset of the samples (horizontal partitioning) or they have a partial set of variables for all samples (vertical partitioning) (Rodríguez et al, 2017; Wu et al, 2018). In this paper, we assume that we are given a global data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, where m is the number of features (genetic variants, in the context of GWAS) and n is the overall number of samples. The data is split across S local sites as $\mathbf{A} = [\mathbf{A}^1 \dots \mathbf{A}^s \dots \mathbf{A}^S]$, where $\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$ and n^s denotes the number of samples available at site s. From a semantic point of view, the partitioning is hence horizontal, since the samples are distributed over the local sites. However, from a technical point of view, the partitioning is vertical, since the samples correspond to the columns of $\mathbf{A}$. The reason for this rather unintuitive setup is that, when using PCA for GWAS, samples are treated as features, as detailed in the following paragraphs.

2.2 Principal component analysis and singular value decomposition

Given a data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the PCA is the decomposition of the covariance matrix $\mathbf{M} = \mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{n \times n}$ into $\mathbf{M} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^\top$. $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the eigenvalues $(\sigma_i)_{i=1}^n$ of $\mathbf{M}$ in non-increasing order, and $\mathbf{V} \in \mathbb{R}^{n \times n}$ is the corresponding matrix of eigenvectors (Jolliffe, 2002). Singular value decomposition (SVD) is closely related to PCA and an extension of PCA to non-square matrices. Many of the PCA algorithms actually call SVD to do the actual computation, because it is more efficient. Given a data matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the SVD is its decomposition into $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vector matrices. Usually, one is only interested in the top k eigenvalues and corresponding eigenvectors. Since k is arbitrary but fixed throughout this paper, we let $\mathbf{G} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{m \times k}$ denote these first k eigenvectors (i. e., $\mathbf{G}$ corresponds to the first k columns of $\mathbf{V}$). $\mathbf{G}$ is typically used to obtain a low-dimensional representation $\mathbf{A} \mapsto \mathbf{A}\mathbf{G} \in \mathbb{R}^{m \times k}$ of the data matrix $\mathbf{A}$, which can then be used for downstream data analysis tasks. This, however, is not the way PCA is used in GWAS, as we will explain next.

2.3 Genome-wide association studies

The genome stores hereditary information that controls the phenotype of an individual in interplay with the environment. The genetic information is stored in the DNA encoded as a sequence of bases (A, T, C, G), the positions are called loci. If we observe two or more possible bases at a specific locus in a population, we call this locus a *single nucleotide polymorphism* (SNP). The predominant base in a population is called the *major allele*; bases at lower frequency are called *minor alleles* (Tam et al, 2019).

Genome wide association studies seek to identify SNPs that are linked to a specific phenotype (Visscher et al, 2017; Tam et al, 2019). Phenotypes of interest can for example be the presence or absence of diseases, or quantitative traits such as height or body mass index. The SNPs for a large cohort of individuals are tested for association with the trait of interest. Typically, simple models such as linear or logistic regression are used for this (Visscher et al, 2017; Nasirigerdeh et al, 2020). The input to a GWAS is an n-dimensional phenotype vector $\mathbf{y}$, a matrix of SNPs $\mathbf{A} \in \mathbb{R}^{m \times n}$, and confounding factors such as age or sex, given as column vectors $\mathbf{x}_r \in \mathbb{R}^n$. The SNPs are encoded as categorical values between 0 and 2, representing the number of minor alleles observed in the individual at the respective position. Each SNP $l \in [m]$ is tested in an individual association test

$$\mathbf{y} \sim \beta_0 + \beta_1 \cdot \mathbf{A}_{l,\bullet}^\top + \sum_{r=1}^{R} \beta_{r+1} \cdot \mathbf{x}_r + \epsilon, \tag{1}$$

where $\mathbf{A}_{l,\bullet}$ denotes the l^{th} row of $\mathbf{A}$.

2.4 Principal component analysis for genome-wide association studies

Confounding factors such as ancestry and population substructure can alter the outcome of an association test and create false hits if not properly controlled for (Tam et al, 2019). PCA has emerged as a popular strategy to infer population substructure and a SMPC based protocol has been presented by (Cho et al, 2018). More precisely, the first k (usually $k = 10$) eigenvectors $\mathbf{G} = [\mathbf{g}_1 \dots \mathbf{g}_k] \in \mathbb{R}^{n \times k}$ of the sample covariance matrix $\mathbf{A}^\top \mathbf{A}$ are included into the association test as covariates (Galinsky et al, 2016; Price et al, 2006):

$$\mathbf{y} \sim \beta_0 + \beta_1 \cdot \mathbf{A}_{l,\bullet}^\top + \sum_{r=1}^{R} \beta_{r+1} \cdot \mathbf{x}_r + \sum_{i=1}^{k} \beta_{i+R+1} \cdot \mathbf{g}_i + \epsilon \tag{2}$$

In federated GWAS, each local site s needs to have access only to the partial eigenvector matrix $\mathbf{G}^s$ corresponding to the locally available samples. Consequently, computing the complete eigenvector matrix $\mathbf{G}$ at the aggregator and/or sharing $\mathbf{G}^s$ with other local sites s' should be avoided to reduce the possibility of information leakage. This is especially important because

Nasirigerdeh et al (2021) have shown that, if $\mathbf{G}$ is available at the aggregator in a federated GWAS pipeline, the aggregator can in principle reconstruct the raw GWAS data $\mathbf{A}_{l,\bullet}$ for SNP l. Federated PCA algorithms that are suitable for GWAS should hence have to respect the following constraint:

Constraint 1 *In a GWAS-suitable federated PCA algorithm, the aggregator does not have access to the complete eigenvector matrix* $\mathbf{G}$ *and each site* s *has access only to its share* $\mathbf{G}^s$ *of* $\mathbf{G}$.

The PCA in GWAS is usually performed on only a subsample of the SNPs, but there seems to be no consensus as to how many SNPs should be used. Some PCA-based stratification methods rely on a small set of ancestry informative markers (Li et al, 2016), while others employ over 100 000 SNPs (Gauch et al, 2019).

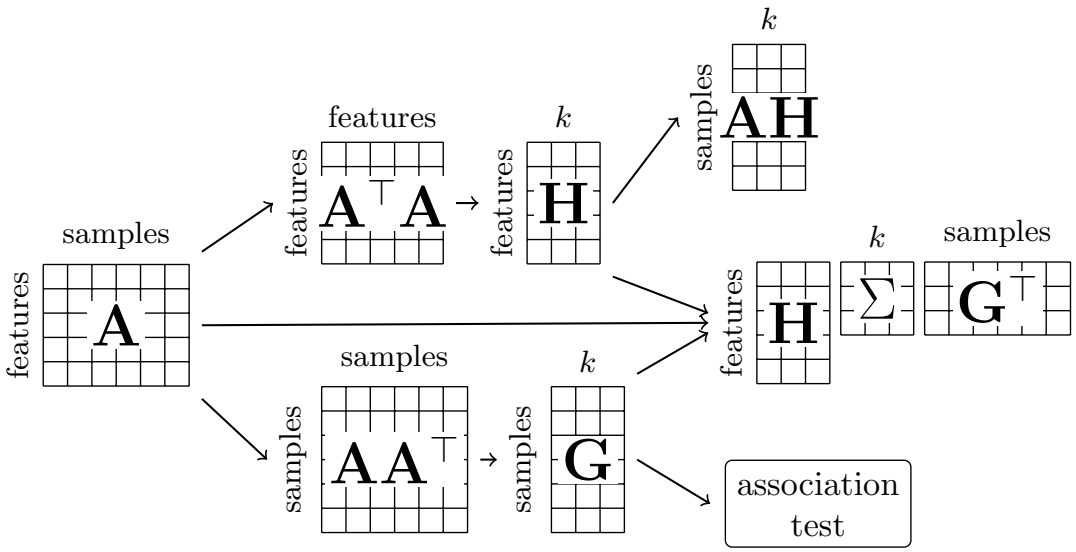


Fig. 2 Regular PCA for dimensionality reduction (top); GWAS PCA for sample stratification (bottom); and SVD (middle).

Note that PCA for GWAS is conceptually different from "regular" PCA for feature reduction (cf. Figure 2). For feature reduction, we would decompose the $m \times m$ SNP by SNP covariance matrix and compute a set of "meta-SNPs" for each sample. This is not what is required for GWAS. Instead, the $n \times n$ sample by sample covariance matrix $\mathbf{A}^\top\mathbf{A}$ is decomposed. In our federated setting where $\mathbf{A}$ is vertically distributed across local sites $s \in [S]$, $\mathbf{A}^\top\mathbf{A}$ looks as follows (recall that, unlike in regular PCA, columns correspond to samples

and rows to features):

$$\mathbf{A}^\top\mathbf{A} = \begin{pmatrix} {\mathbf{A}^1}^\top\mathbf{A}^1 & {\mathbf{A}^1}^\top\mathbf{A}^2 & \cdots & {\mathbf{A}^1}^\top\mathbf{A}^S \\ {\mathbf{A}^2}^\top\mathbf{A}^1 & {\mathbf{A}^2}^\top\mathbf{A}^2 & \cdots & {\mathbf{A}^2}^\top\mathbf{A}^S \\ \vdots & \vdots & \ddots & \vdots \\ {\mathbf{A}^S}^\top\mathbf{A}^1 & {\mathbf{A}^S}^\top\mathbf{A}^2 & \cdots & {\mathbf{A}^S}^\top\mathbf{A}^S \end{pmatrix} \tag{3}$$

It is clear that $\mathbf{A}^\top\mathbf{A}$ cannot be computed directly without sharing patient level data. Moreover, with a growing number of samples, this matrix can become very large and computing it becomes infeasible. For instance, the UK Biobank — a large cohort frequently used for GWAS — contains more than 4 million SNPs for more than 500 000 individuals. Following directly from the definition of PCA, an exact computation of the covariance matrix would furthermore violate Constraint 1. These considerations lead to the second constraint for federated PCA algorithms suitable for GWAS:

Constraint 2 *A GWAS-suitable federated PCA algorithm must work on vertically partitioned data and does not rely on computing or approximating the covariance matrix.*

2.5 Gram-Schmidt orthonormalization

The Gram-Schmidt algorithm transforms a set of linearly independent vectors into a set of mutually orthogonal vectors. Given a matrix $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_k] \in \mathbb{R}^{r\times k}$ of k linearly independent column vectors, a matrix $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_k] \in \mathbb{R}^{r\times k}$ of orthogonal column vectors with the same span can be computed as

$$\mathbf{u}_i = \begin{cases} \mathbf{v}_i & \text{if } i = 1 \\ \mathbf{v}_i - \sum_{j=1}^{i-1} r_{i,j} \cdot \mathbf{u}_j & \text{if } i \in [k] \setminus \{1\} \end{cases}, \tag{4}$$

where $r_{i,j} = \mathbf{u}_j^\top \mathbf{v}_i / n_j$ with $n_j = \mathbf{u}_j^\top \mathbf{u}_j$.

The vectors can then be scaled to unit Euclidean norm as $\mathbf{u}_i \mapsto (1/\sqrt{n_i}) \cdot \mathbf{u}_i$ to achieve a set of orthonormal vectors. In the context of PCA, this can be used to ensure orthonormality of the candidate eigenvectors in iterative procedures, which otherwise suffer from numerical instability in practice (Guo et al, 2012).

2.6 Notations

Table 1 provides an overview of notations which are used throughout the paper.

Table 1 Notation table.

Syntax	Semantics
$[N] \subset \mathbb{N}$	index set $[N] = \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$
$S \in \mathbb{N}$	number of sites
$m \in \mathbb{N}$	number of features (i. e. SNPs)
$n \in \mathbb{N}$	total number of samples
$n^s \in \mathbb{N}$	number of samples at site $s \in [S]$
$k \in \mathbb{N}$	number of eigenvectors
$\mathbf{A} \in \mathbb{R}^{m \times n}$	complete data matrix
$\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$	subset of data available at site $s \in [S]$
$\mathbf{G}_i \in \mathbb{R}^{n \times k}$	right singular matrix of $\mathbf{A}$ at iteration i
$\mathbf{G} \in \mathbb{R}^{n \times k}$	right singular matrix of $\mathbf{A}$
$\mathbf{G}_i^s \in \mathbb{R}^{n^s \times k}$	partial right singular matrix of $\mathbf{A}$ at iteration i
$\mathbf{G}^s \in \mathbb{R}^{n^s \times k}$	converged partial right singular matrix $\mathbf{A}$.
$\mathbf{H}_i \in \mathbb{R}^{m \times k}$	left singular matrix of $\mathbf{A}$ at iteration i
$\mathbf{H} \in \mathbb{R}^{m \times k}$	converged left singular matrix of $\mathbf{A}$
$\mathbf{V} \in \mathbb{R}^{r \times k}$	a generic column vector matrix
$\mathbf{U} \in \mathbb{R}^{r \times k}$	an orthonormal matrix with $\text{span}(\mathbf{U}) = \text{span}(\mathbf{V})$
$\mathbf{M} \in \mathbb{R}^{m \times m}$	exact covariance matrix
$\hat{\mathbf{A}}, \hat{\mathbf{M}}, \hat{\mathbf{H}}, \hat{\mathbf{G}}$	approximations of $\mathbf{A}$, $\mathbf{M}$, $\mathbf{H}$ and $\mathbf{G}$

3 Related work

3.1 Centralized, iterative, covariance-free principal component analysis

While classical PCA algorithms rely on computing the covariance matrix $\mathbf{A}^\top\mathbf{A}$ (Jolliffe, 2002), there are several covariance-free approaches to iteratively approximate the top k eigenvalues and eigenvectors (Saad, 2011). Algorithm 1 summarizes the centralized, iterative, covariance-free PCA algorithm suggested by Halko et al (2011), which will serve as point of departure for our federated approach. First, an initial eigenvector matrix is sampled randomly and orthonormalized (lines 1 to 2). In every iteration i, improved candidate eigenvectors $\mathbf{G}_i$ of $\mathbf{A}^\top\mathbf{A}$ are computed (lines 5 to 8). Once a suitable termination criterion is met (e.g., convergence, maximal number of iterations, time limit, etc.), the last candidate eigenvectors are returned (line 10).

Algorithm 1: Vertical Subspace Iteration Halko et al (2011).

Input: Data matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, number of eigenvectors k.
Output: Singular matrices $\mathbf{G} \in \mathbb{R}^{n\times k}$ and $\mathbf{H} \in \mathbb{R}^{m\times k}$ of $\mathbf{A}$.

1 Generate $\mathbf{G}_0 \in \mathbb{R}^{n\times k}$ randomly;
2 $\mathbf{G}_0 \leftarrow \texttt{orthonormalize}(\mathbf{G}_0)$;
3 $i \leftarrow 1$;
4 **while** *termination criterion not met* **do**
5 $\quad \mathbf{H}_i = \mathbf{A}\mathbf{G}_{i-1}$;
6 $\quad \mathbf{H}_i = \texttt{orthonormalize}(\mathbf{H}_i)$;
7 $\quad \mathbf{G}_i = \mathbf{A}^\top\mathbf{H}_i$;
8 $\quad \mathbf{G}_i \leftarrow \texttt{orthonormalize}(\mathbf{G}_i)$;
9 $\quad i \leftarrow i + 1$;
10 **return** $\mathbf{G}_i$, $\mathbf{H}_i$;

To update the candidate eigenvector matrices $\mathbf{G}_i = \mathbf{A}^\top\mathbf{H}_i = \mathbf{A}^\top\mathbf{A}\mathbf{G}_{i-1} \in \mathbb{R}^{n\times k}$ of $\mathbf{A}^\top\mathbf{A}$, the algorithm also computes candidate eigenvector matrices $\mathbf{H}_i = \mathbf{A}\mathbf{G}_{i-1} = \mathbf{A}\mathbf{A}^\top\mathbf{H}_{i-1} \in \mathbb{R}^{m\times k}$ of $\mathbf{A}\mathbf{A}^\top$. Since, in the context of GWAS, $\mathbf{A}\mathbf{A}^\top$ corresponds to the "classical" feature by feature covariance matrix, and $\mathbf{A}^\top\mathbf{A}$ to the sample covariance matrix, the algorithm computes left and right singular vectors at the same time. This means, the present algorithm is actually an SVD algorithm. In this article, we will sometimes refer to the left singular vector as the feature eigenvector and the right singular vector as the sample eigenvector.

3.2 Federated principal component analysis for vertically partitioned data

Only few algorithms are designed to perform federated computation of PCA on vertically partitioned siloed data sets (Guo et al, 2012; Kargupta et al, 2001; Qi et al, 2003; Wu et al, 2018). However, none of them is suitable for the GWAS use-case considered in this paper: The algorithms reviewed by Wu et al (2018) are specialised for distributed sensor networks and use gossip protocols and peer-to-peer communication. Therefore, they are not suited for the intended FL architecture in the medical setting. The algorithms presented by Kargupta et al (2001) and Qi et al (2003) rely on estimating a proxy covariance matrix and consequently do not meet Constraint 2 introduced above. Unlike these approaches, the algorithm proposed by Guo et al (2012) is covariance-free and suitable for the intended star-like architecture. However, it broadcasts the eigenvectors to all sites in violation of Constraint 1.

3.3 Federated matrix orthonormalization

Matrix orthonormalization is a frequently used technique in many applications, including the solution of linear systems of equations and singular value decomposition. There are three main approaches: Householder reflection, Givens rotation, and the Gram-Schmidt algorithm. In distributed memory systems and grid architectures, tiled Householder reflection is a popular approach (Hadri et al, 2010; Hoemmen, 2011). However, those algorithms are often highly specialized to the compute system and rely on shared disk storage. For distributed sensor networks, Gram-Schmidt procedures relying on push-sum have been proposed (Sluciak et al, 2016; Straková et al, 2012). However, these methods require peer-to-peer communication and are hence unsuitable for the intended star-like architecture. Consequently, no federated orthonormalization algorithm suitable for our setup is available. In Section 4.2, we present our own version of a federated orthonormalization algorithm fulfilling all constraints and subsequently utilize it as a subroutine in our federated PCA algorithm.

3.4 Federated principal component analysis for horizontally partitioned data

Previously, federated PCA algorithms have been described for horizontal and vertical data partitioning. In the remainder of this article, we establish an algorithm which is capable of both, which allows us to borrow ideas from previously described algorithms for horizontally federated PCA. There are "single-round" approaches, where the eigenvectors are computed locally and sent to the aggregator (Balcan et al, 2014). At the aggregator, a global subspace is approximated from the local eigenspaces. The higher the number of transmitted intermediate dimensions, the better the global subspace approximation. In these algorithms, the solution quality hence depends on the number of transmitted dimensions. This algorithm is a more memory efficient version of the naive algorithm [e. g. (Liu et al, 2020)], where the entire covariance matrix is processed

by the aggregator. Since only the top k left singular values are transmitted, this algorithm fulfills constraint 2. Furthermore, iterative schemes have been proposed, where locally computed eigenvectors are sent to the aggregator, which performs an aggregation step and sends the obtained candidate subspace back to the clients (Balcan et al, 2016; Chen et al, 2020; Imtiaz and Sarwate, 2018). The candidate subspace is then refined iteratively. Furthermore, there are several schemes for specific applications such as streaming (Sanchez-Fernandez et al, 2015; Grammenos et al, 2020). These approaches assume that an approximation of the entire eigenvectors is possible at the clients, or that the global covariance matrix can be approximated. As we have discussed above, these assumptions do not hold in the intended GWAS use case.

3.5 Randomized principal component analysis

In the context of GWAS, a randomized PCA algorithm (Halko et al, 2011; Galinsky et al, 2016) is popular as it speeds-up the computation compared to traditional algorithms. Here, we briefly present the version implemented by Galinsky et al (2016). The algorithm starts with I' iterations of subspace iteration on the full-dimensional data matrix, resulting in feature eigenvetor matrices H_i for all iterations $i \in \{1, \ldots, I'\}$. Next, the data is projected on the concatenation of all $\mathbf{H}_i$, forming approximate principal components which approximate the data matrices. Then, subspace iteration is performed on these proxy data matrices. In practice, $I' = 10$ iterations are sufficient. This reduces the dimensionality of the data from m to $k \cdot I'$. In Section 4.3 we will present a fully federated version of this algorithm in detail. Note that this is a randomized approach, because subspace iteration on the full-dimensional data is initialized randomly. Since it is interrupted before convergence after I' iterations, the feature eigenvetor matrices H_i inherit this randomness.

4 Algorithms

In this section, we present a federated SVD algorithm, which is designed for a star-like architecture, meets the requirements of Constraint 1 and Constraint 2, and is hence suitable for federated GWAS. Our base algorithm comes in two variants — with and without orthonormalization of the candidate right singular vectors of $\mathbf{A}$. In Section 4.1, we describe our algorithm and prove that the version with orthonormalization is equivalent to centralized vertical subspace iteration algorithm (Halko et al, 2011), which we have summarized in Algorithm 1 above. In Section 4.2, we present a federated Gram-Schmidt algorithm, which can be used as a subroutine in our federated SVD algorithm to ensure that right singular vectors of $\mathbf{A}$ remain at the local sites. Again, we prove that our federated Gram-Schmidt algorithm is equivalent to the centralized counterpart. We then show how approximate horizontal PCA can be used to compute approximate principal components for immediate use or as initialization for subspace iteration in Section 4.3. In Section 4.4, we present randomized federated subspace iteration as a means to reduce the transmission cost in federated SVD. In addition to decreasing the communication cost, the use of the two latter strategies also prevents potential iterative leakage (detailed in Section 5). In Section 4.5, we analyze the network transmission costs of the proposed algorithms, and Section 4.6 provides an overview of possible configurations of our federated SVD algorithm.

4.1 Federated vertical subspace iteration

Algorithm 2 describes our federated vertical subspace iteration algorithm: Initially, the first partial candidate right singular matrices $\mathbf{G}_0^s$ of $\mathbf{A}$ are generated randomly and orthonormalized (lines 4 to 5). Inside the main loop, the left singular vectors are updated at the clients, summed up element-wise and orthonormalized at the aggregator, and then sent back to the clients (lines 9 to 11). Next, the clients update the partial right singular vectors (line 13). In the version with orthonormalization, the candidate right singular vectors are now normalized by calling the federated Gram-Schmidt orthonormalization (line 14) algorithm presented in Section 4.2 (Algorithm 3). Note that this algorithm ensures that the partial singular vectors $\mathbf{G}_i^s$ remain at the local sites. Finally, the full left singular matrices $\mathbf{H}$ and the orthonormalized partial right singular matrices $\mathbf{G}_s$ are returned to the clients (line 19). In practice, the federated orthonormalization of $\mathbf{G}_i$ (line 14) may be omitted to speed up computation. Note, however, that $\mathbf{H}_i$ is still orthonormalized in every iteration and that the final orthonormalization (line 18) is required.

Like the original centralized version described in Algorithm 1 above, our algorithm can be run with various termination criteria. In our implementation, we use the convergence criterion

$$\operatorname{diag}(\mathbf{H}_i^\top \mathbf{H}_{i-1}) \geq \mathbf{1}_k - \epsilon \tag{5}$$

Algorithm 2: Federated vertical subspace iteration. (Partial) client-side computations are marked in gray.

Input: Partial data matrices $\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$ at sites $s \in [S]$, number of eigenvectors k, number of iterations I and/or convergence threshold ϵ.

Output: Partial right singular matrices $\mathbf{G}^s \in \mathbb{R}^{n^s \times k}$ and full left singular matrices $\mathbf{H} \in \mathbb{R}^{m \times k}$ of $\mathbf{A}$ at sites $s \in [S]$.

if *use approximate initialisation* **then**
  // Call subroutine Algorithm 4 (AI-FULL).
  $\mathbf{G}_0^s \leftarrow$ `init-approximative`()
else
  // Use random initialisation if no initial eigenvector is available (RI-FULL).
  for $s \in [S]$ **do** generate $\mathbf{G}_0^s \in \mathbb{R}^{n^s \times k}$ randomly;
  // Use approach described in Algorithm 3 (FED-GS).
  if *use orthonormalization* **then** `federated-gram-schmidt`($[\mathbf{G}_0^s]$);
$i \leftarrow 1$;
// Suggested criterion: $i \geq I$ or convergence as specified in eq. (5).
while *termination criterion not met* **do**
  // Update left singular matrix of A.
  for $s \in [S]$ **do** $\mathbf{H}_i^s \leftarrow \mathbf{A}^s \mathbf{G}_{i-1}^s$;
  $\mathbf{H}_i \leftarrow \sum_{s=1}^{S} \mathbf{H}_i^s$;
  $\mathbf{H}_i \leftarrow$ `orthonormalize`($\mathbf{H}_i$);
  // Update partial right singular matrices of $\mathbf{A}$.
  for $s \in [S]$ **do** $\mathbf{G}_i^s \leftarrow {\mathbf{A}^s}^\top \mathbf{H}_i$;
  // Use approach described in Algorithm 3 (FED-GS).
  if *use orthonormalization* **then** `federated-gram-schmidt`($[\mathbf{G}_i^s]$);
  $i \leftarrow i + 1$;
for $s \in [S]$ **do**
  $\mathbf{G}^s \leftarrow \mathbf{G}_i^s$;
$\mathbf{G}^s \leftarrow$ `federated-gram-schmidt`($[\mathbf{G}^s]$);
return $\mathbf{G}^s, \mathbf{H}$;

using the angle as a global measure as suggested by Lei et al (2016), where $\mathbf{1}_k$ is the k-dimensional vector of ones and ϵ is a small positive number. With this criterion, the algorithm terminates once all right singular vectors of $\mathbf{A}$ are asymptotically collinear with respect to the eigenvectors of the previous iteration. Other convergence criteria could be used as drop-in replacements.

We now prove that the version of Algorithm 2 with orthonormalization is equivalent to the centralized version described in Algorithm 1. Thus, it inherits its convergence behavior from the centralized version. Details on the convergence behavior of centralized vertical subspace iteration can be found in the original publication by Halko et al (2011).

Proposition 1 *If orthonormalization is used, centralized and federated vertical subspace iteration are equivalent.*

Proof Let $\mathbf{G}_i$ and $\mathbf{H}_i$ denote the eigenvector matrices maintained by the centralized algorithm described in Algorithm 1 at the end of the main while-loop, and $\mathbf{G}_i^s$ be the sub-matrix of $\mathbf{G}_i$ for the samples available at site s. Moreover, let $\tilde{\mathbf{H}}_i$, $\tilde{\mathbf{G}}_i$, $\tilde{\mathbf{G}}_i^s$, and $\tilde{\mathbf{H}}_i^s$ be the (partial) eigenvector matrices maintained by our federated algorithm 2 at the end of the main while-loop. We will show by induction on the iterations i that $\mathbf{H}_i = \tilde{\mathbf{H}}_i$ and $\mathbf{G}_i^s = \tilde{\mathbf{G}}_i^s$ for all $s \in [S]$ holds throughout the algorithm, if the same random seeds are used for initialization.

For $i = 0$, we only have to show $\mathbf{G}_0^s = \tilde{\mathbf{G}}_0^s$. This directly follows from proposition 2 and our assumption that the same random seeds are used for initialization. For the inductive step, note that, before orthonormalization in line 11, we have $\tilde{\mathbf{H}}_i = \sum_{s=1}^S \tilde{\mathbf{H}}_i^s = \sum_{s=1}^S \mathbf{A}^s \tilde{\mathbf{G}}_{i-1}^s = \sum_{s=1}^S \mathbf{A}^s \mathbf{G}_{i-1}^s = \mathbf{A}\mathbf{G}_{i-1} = \mathbf{H}_i$, where the third equality follows from the inductive assumption. Because of Proposition 2, this identity continues to hold at the end of the main while-loop.

Similarly, after updating in line 13 but before orthonormalization, we have $\tilde{\mathbf{G}}_i^s = {\mathbf{A}^s}^\top \tilde{\mathbf{H}}_i = {\mathbf{A}^s}^\top \mathbf{H}_i = (\mathbf{A}^\top \mathbf{H}_i)^s = \mathbf{G}_i^s$, where the second equality follows the identity $\mathbf{H}_i = \tilde{\mathbf{H}}_i$ shown above and $(\mathbf{A}^\top \mathbf{H}_i)^s$ denotes the sub-matrix of $\mathbf{A}^\top \mathbf{H}_i$ for the samples available at site s. Again, proposition 2 ensures that the identity continues to hold after orthonormalization. □

The omission of the orthonormalization of $\mathbf{G}_i$ (line 5 and line 14 in Algorithm 2) removes provable identity to algorithm 1. However, other formulations of centralized power iteration exist which directly operate on the covariance matrix (Balcan et al, 2016). In these schemes, instead of splitting the iteration into $\mathbf{H}_i$ update (line 5, algorithm 1) and $\mathbf{G}_i$ update (line 7, algorithm 1), the covariance matrix is computed and $\mathbf{H}_i$ is updated as $\mathbf{H}_i = \mathbf{A}\mathbf{A}^\top \mathbf{H}_{i-1}$ at every iteration. Proposition 1 can be formulated and proven analogously for this version.

4.2 Federated Gram-Schmidt algorithm

Here, we describe federated Gram-Schmidt orthonormalization for vertically partitioned column vectors. Previous federated PCA algorithms require the complete eigenvectors to be known at all sites for the orthonormalization procedure. The naïve way of orthonormalizing the eigenvector matrices in a federated fashion would be to send them to the aggregator which performs the aggregation and then sends the orthonormal matrices back to the clients. However, in this naïve scheme, the transmission cost scales with the number of variables (individuals in GWAS) and all eigenvectors are known to the aggregator.

To address these two problems, we suggest a federated Gram-Schmidt orthonormalization procedure, summarized in Algorithm 3. The algorithm exploits the fact that the computations of the squared norms n_i and of the residuals r_{ij} can be decomposed into independent computations of summands n_i^s and r_{ij}^s computable at the local sites $s \in [S]$. The clients compute the local summands and send them to the aggregator, where the squared norm of the first orthogonal vector is computed and sent to the clients (lines 2 to 5). Subsequently, the remaining $k-1$ vectors are orthogonalized. For the i^{th}

vector $\mathbf{v}_i$, the algorithm computes the residuals r_{ij} w. r. t. all already computed orthogonal vectors $\mathbf{u}_j$, using the fact that the corresponding squared norms n_j are already available (lines 8 to 10). The residuals are aggregated by the central server (lines 12 to 13). Next, $\mathbf{v}_i$ is orthogonalized at the clients, the local norms are computed (lines 15 to 17), and the squared norm of the resulting orthogonal vector $\mathbf{u}_i$ is computed at the aggregator and sent back to the clients (line 19). After orthogonalization, all orthogonal vectors are scaled to unit norm at the clients (lines 21 to 23).

Algorithm 3: Federated Gram-Schmidt. Client-side computations are marked in gray.

Input: Data matrices $\mathbf{V}_s$ at sites $s \in [S]$.
Output: Orthonormalized data matrices $\mathbf{U}_s$ at sites $s \in [S]$.

1 // Compute squared norm of first orthogonal vector.
2 **for** $s \in [S]$ **do**
3 $\quad \mathbf{u}_1^s \leftarrow \mathbf{v}_1^s$;
4 $\quad n_1^s \leftarrow \mathbf{u}_1^{s\top} \mathbf{u}_1^s$;
5 $n_1 \leftarrow \sum_{s=1}^{S} n_1^s$;
// Orthogonalize all subsequent vectors.
6 **for** $i \in [k] \setminus \{1\}$ **do**
7 $\quad$ // Compute client residuals for vector being orthogonalized.
8 $\quad$ **for** $s \in [S]$ **do**
9 $\quad\quad$ **for** $j \in [i-1]$ **do**
10 $\quad\quad\quad r_{ij}^s \leftarrow \mathbf{u}_j^{s\top} \mathbf{v}_i^s / n_j$;
11 $\quad$ // Compute global residuals for vector being orthogonalized.
12 $\quad$ **for** $j \in [i-1]$ **do**
13 $\quad\quad r_{ij} \leftarrow \sum_{s=1}^{S} r_{ij}^s$;
14 $\quad$ // Orthogonalize the vector and compute squared norm.
15 $\quad$ **for** $s \in [S]$ **do**
16 $\quad\quad \mathbf{u}_i^s \leftarrow \mathbf{v}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s$;
17 $\quad\quad n_i^s \leftarrow \mathbf{u}_i^{s\top} \mathbf{u}_i^s$
18 $\quad$ // Compute squared norm of orthogonalized vector.
19 $\quad n_i \leftarrow \sum_{s=1}^{S} n_i^s$;
20 // After orthogonalization, scale all k vectors to unit norm.
21 **for** $s \in [S]$ **do**
22 $\quad$ **for** $i \in [k]$ **do** $\mathbf{u}_i^s \leftarrow \frac{1}{\sqrt{n_i}} \cdot \mathbf{u}_i^s$;
23 $\quad \mathbf{U}^s \leftarrow [\mathbf{u}_1^s \dots \mathbf{u}_s^k]$;
24 **return** $\mathbf{U}^s$;

Proposition 2 *Centralized and federated Gram-Schmidt orthonormalization are equivalent.*

Proof Let $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_k]$ be the matrix that should be orthonormalized, $\mathbf{v}_i^s$ be the restriction of the i^{th} columns vector to the samples available at side s, and $\mathbf{u}_i^s$ be the restriction of the i^{th} orthogonal vector computed by the centralized Gram-Schmidt algorithm before normalization to the samples available at site s. Moreover, let n_i and $r_{i,j}$ be the centrally computed norms and residuals, and $\tilde{n}_i$, $\tilde{r}_{i,j}$, and $\tilde{\mathbf{u}}_i^s$ be the locally computed norms, residuals, and partial orthogonal vectors before normalization. We show by induction on i that $n_i = \tilde{n}_i$, $r_{ij} = \tilde{r}_{ij}$, and $\mathbf{u}_i^s = \tilde{\mathbf{u}}_i^s$ holds for all $i \in [k]$ and all $j \in [i-1]$. This implies the proposition.

For $i = 1$, we have $\mathbf{u}_1^s = \mathbf{v}_1^s = \tilde{\mathbf{u}}_1^s$ and $n_1 = \mathbf{u}_1^\top \mathbf{u}_1 = \sum_{s=1}^S \mathbf{u}_1^{s\top} \mathbf{u}_1^s = \sum_{s=1}^S \tilde{\mathbf{u}}_1^{s\top} \tilde{\mathbf{u}}_1^s = \tilde{n}_1$. For the inductive step, note that $r_{ij} = \mathbf{u}_j{}^\top \mathbf{v}_i / n_j = \sum_{s=1}^S \mathbf{u}_j^{s\top} \mathbf{v}_i^s / n_j = \sum_{s=1}^S \tilde{\mathbf{u}}_j^{s\top} \mathbf{v}_i^s / \tilde{n}_j = \tilde{r}_{ij}$, where the third identity follows from the inductive assumption. Moreover, we have $\mathbf{u}_i^s = \mathbf{v}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s = \mathbf{v}_i^s - \sum_{j=1}^{i-1} \tilde{r}_{ij} \cdot \tilde{\mathbf{u}}_j^s = \tilde{\mathbf{u}}_i^s$, where the second identity follows from the inductive assumption and the identities $r_{ij} = \tilde{r}_{ij}$ established before. We hence obtain $n_i = \mathbf{u}_i^\top \mathbf{u}_i = \sum_{s=1}^S \mathbf{u}_i^{s\top} \mathbf{u}_i^s = \sum_{s=1}^S \tilde{\mathbf{u}}_i^{s\top} \tilde{\mathbf{u}}_i^s = \tilde{n}_i$, which completes the proof. □

4.3 Approximate initialization

One major concern of iterative PCA is information leakage through the repeated transmission of updated eigenvectors. This is presented in more detail in Section 5, because knowledge of the subspace iteration algorithm is required to understand the attack. Briefly, the conclusion is that the number of iterations needs to be strictly limited. Therefore, we suggest to use federated approximate horizontal PCA as an initialization strategy to limit the number of iterations, and thereby prevent the possible leakage of the covariance matrix.

Balcan et al (2014) presented a memory efficient version of federated approximate PCA for horizontally partitioned data. We provide a minor modification which allows us to compute the sample eigenvectors. The algorithm can be used "as is" to compute the federated approximate vertical PCA by projecting the approximate left eigenvector to the data; or as an initialization strategy for federated subspace iteration. For the latter, instead of initializing $\mathbf{G}_0^s$ randomly (line 4, Algorithm 2), G_0^s is computed using the approximate algorithm described here (line 2, Algorithm 2).

Algorithm 4 describes this approach. The algorithm proceeds as follows: At the clients, a local PCA is computed and the top $2k$ eigenvectors are shared with the aggregator with c a constant multiplicative factor (line 2). At the aggregator, the local eigenvectors are stacked such that a new approximate covariance matrix $\hat{\mathbf{M}}$ with $\dim(\hat{\mathbf{M}}) = c \cdot k \cdot S \times m$ is formed. $\hat{\mathbf{M}}$ is then decomposed using singular value decomposition leading to a new eigenvector estimate $\hat{\mathbf{H}}$ (lines 4 to 5). At the clients, the feature eigenvector estimate $\hat{\mathbf{H}}$ can be projected onto the data to form an approximation of the sample eigenvector $\hat{\mathbf{G}}_s$. The vectors $\hat{\mathbf{G}}$ and $\hat{\mathbf{H}}$ represent an "educated guess" of the final singular vectors.

Algorithm 4: Slightly modified federated horizontal SVD (Balcan et al, 2014). Referred to as AI-ONLY in this article.

Input: Data matrices $\mathbf{A}_s \in \mathbb{R}^{m \times n}$ at sites $s \in [S]$, number of eigenvectors k, constant approximation factor c.
Output: Approximate singular vector matrices $\hat{\mathbf{G}}_s \in \mathbb{R}^{n_s \times k}$ and $\hat{\mathbf{H}} \in \mathbb{R}^{m \times k}$ of $\mathbf{A}$.

1 **for** $s \in [S]$ **do**
  // Retrieve top $k \cdot c$ eigenvectors.
2 $\mathbf{H}_s, \mathbf{\Sigma}_s, \mathbf{G}_s \leftarrow$ `singular-value-decomposition`$(\mathbf{A}_s, c \cdot k)$;
3 // Aggregate local subspaces to obtain approximate covariance matrix $\hat{\mathbf{M}}$ with $\dim(\hat{\mathbf{M}}) = c \cdot k \cdot S \times m$.
4 $\hat{\mathbf{M}} \leftarrow$ `stack-vertically`$([\mathbf{H}_s^\top])$;
// Use final dimensionality k
5 $\hat{\mathbf{H}} \leftarrow$ `singular-value-decomposition`$(\hat{\mathbf{M}}, k)$;
6 **for** $s \in [S]$ **do**
7 $\hat{\mathbf{G}}_s \leftarrow \mathbf{A}^{s\top}\hat{\mathbf{H}}$;
8 // Return approximate singular vector matrices of $\mathbf{A}$
9 **return** $\hat{\mathbf{G}}_s, \hat{\mathbf{H}}$

4.4 Federated randomized principal component analysis

Another mitigation strategy for the aforementioned information leakage is the use of randomized SVD. In randomized SVD, a reduced representation of the data is computed and subspace iteration is applied on this reduced data matrix instead of the full data. By using the proxy data, only "reduced" eigenvectors become available at the aggregator which makes the attack in Section 5 impossible given not too many initial iteration I' have been executed. Notably, I' needs to be restricted depending on the number of features in the original data. Here, we describe how to modify randomized SVD, such that it can be run in a federated environment, without sharing the random projections of the data or the sample eigenvectors.

We proceed according to Halko et al (2011) and Galinsky et al (2016). First, I' iterations of federated vertical subspace iteration are run using the full data matrices $\mathbf{A}_s$. In order to do so, Algorithm 2 is called as a subroutine. The intermediate matrices $\mathbf{H}_1, \ldots, \mathbf{H}'_I$ are stored (line 1) and concatenated to form $\mathbf{P} \in \mathbb{R}^{k \cdot I' \times m}$ (line 2). The data matrices $\mathbf{A}_s$ are then projected onto $\mathbf{P}$ to form proxy data matrices $\hat{\mathbf{A}}_s \in \mathbb{R}^{k \cdot I' \times n}$ (line 4). Finally, federated subspace iteration (Algorithm 2) is called as a subroutine again and run until convergence using the proxy data matrices $\hat{\mathbf{A}}_s$ (line 6). The subroutine returns the correct right singular vectors $\mathbf{G}$ but only proxy vectors for $\mathbf{H}$. Therefore, in the last step $\mathbf{H}$ can be reconstructed by projecting the data onto $\mathbf{G}$, aggregating and normalizing $\mathbf{H}^s$ at the aggregator and returning the final left singular vectors $\mathbf{H}$ to the clients (lines 8 to 10).

We would like to highlight two properties of this algorithm. Firstly, given that $m > I/k$, it is not possible to construct the covariance matrix using the initial eigenvector updates (see Section 5). Secondly, since the computation of the final right singular vectors utilizes the projected data matrices, it is not possible to construct the original covariance matrix naïvely using the additional I iterations until convergence.

Algorithm 5: Federated randomized SVD (RI-RAND, AI-RAND)

Input: Data matrices $\mathbf{A}_s \in \mathbb{R}^{m \times n}$ at sites $s \in [S]$, number of eigenvectors k, number of intermediate iterations I', number of total iterations I.

Output: Singular vector matrices $\mathbf{H} \in \mathbb{R}^{m \times k}$ and partial $\mathbf{G}_s \in \mathbb{R}^{n_s \times k}$ of $\mathbf{A}$.

// Run I' iterations of Algorithm 2 and store H_i.

$[\mathbf{H}_1, ..., \mathbf{H}_{I'}] \leftarrow$ `federated-subspace-iteration`$([\mathbf{A}_s], k, I')$;

$\mathbf{P} \leftarrow$ `stack-vertically`$([\mathbf{H}_1^\top, ..., \mathbf{H}_{I'}^\top])$;

for $s \in [S]$ **do**

  $\hat{\mathbf{A}}_s \leftarrow \mathbf{P}\mathbf{A}_s$

// Run Algorithm 2 until convergence using proxy matrix $\hat{\mathbf{A}}$. Use Algorithm 2 with approximate initialisation (AI-RAND) or random initialisation (RI-RAND).

$\mathbf{G}_s \leftarrow$ `federated-subspace-iteration`$([\hat{\mathbf{A}}_s], k, I)$;

// Compute the full left singular vectors of A

for $s \in [S]$ **do** $\mathbf{H}^s \leftarrow \mathbf{A}^s\mathbf{G}^s$;

$\mathbf{H} \leftarrow \sum_{s=1}^{S} \mathbf{H}^s$;

$\mathbf{H} \leftarrow$ `orthonormalize`$(\mathbf{H})$;

// Return singular vector matrix of $\mathbf{A}$.

return $\mathbf{G}_s, \mathbf{H}$

4.5 Network transmission costs

The main bottleneck in FL is the amount of data transmitted between the different sites and the number of network communications and the volume of transmitted date (Kairouz et al, 2021). The following Proposition 3 specifies these quantities for our federated PCA algorithm. Recall that S, k, m, n, and c denote, respectively, the numbers of sites, eigenvectors, features, samples, and a constant multiplicative factor.

Proposition 3 *Let $\mathcal{D}$ be the total amount of data transmitted by our federated SVD algorithm, $\mathcal{N}$ be the total number of network communications, and I be the total number of iterations of the main while-loop. Let further I' be the number of inital iteration for randomized SVD and k' the intermediate dimensionality of the subspace for the approximate algorithm. Then the following statements hold:*

- *If the* $\mathbf{G}_i$ *matrices are not orthonormalized, then* $\mathcal{D} = \mathcal{O}(I \cdot S \cdot k \cdot m)$ *and* $\mathcal{N} = \mathcal{O}(I \cdot S)$.
- *If federated Gram-Schmidt orthonormalization is used, then* $\mathcal{D} = \mathcal{O}(I \cdot (S \cdot k \cdot m + k^2))$ *and* $\mathcal{N} = \mathcal{O}(I \cdot S \cdot k)$.
- *If federated randomized subspace iteration is used, then* $\mathcal{D} = \mathcal{O}(I' \cdot S \cdot k \cdot (m + I \cdot k))$ *and* $\mathcal{N} = \mathcal{O}((I + I') \cdot S)$.
- *Approximate initialization itself has a complexity of* $\mathcal{D} = \mathcal{O}(S \cdot k \cdot c \cdot m)$ *and* $\mathcal{N} = \mathcal{O}(S)$*, hence the other algorithms remain in the same complexity class if used in combination with approximate initialization.*

Proof In each iteration i of our federated vertical subspace iteration algorithm, the matrices $\mathbf{H}_i^s \in \mathbb{R}^{m \times k}$ have to be sent from the clients to the aggregator and the matrix $\mathbf{H}_i \in \mathbb{R}^{m \times k}$ has to be sent back to the clients. In iteration i, the amount of transmitted data and the number of communications due to $\mathbf{H}_i$ is hence $\mathcal{O}(S \cdot k \cdot m)$ and $\mathcal{O}(S)$, respectively. For orthonormalizing the eigenvector matrices $\mathbf{G}_i \in \mathbb{R}^{n \times k}$, we need to transmit a data volume of $\mathcal{O}(S \cdot k^2)$ and the number of communications increases to $\mathcal{O}(S \cdot k)$. By summing over the iterations i, this yields the statement of the proposition. In the randomized iteration the first I' iterations have the same communication complexity that regular subspace iteration. Then the dimensionality of the matrix is reduced to $k \cdot I' \times n$ and the decomposition of $\hat{\mathbf{A}}$ with k eigenvectors has complexity $k \cdot I' \cdot k$ for a single iteration. Thereby, the total complexity is $\mathcal{D} = \mathcal{O}(I' \cdot S \cdot k \cdot m + I \cdot S \cdot I' \cdot k^2) = \mathcal{O}(I' \cdot S \cdot k \cdot (m + I \cdot k))$. Approximate initialization has a complexity of one round of subspace iteration, as $\mathbf{H}_i$ needs to be communicated once to the aggregator and back. The complexity classes hence remain the same. □

If our algorithms are used, the overall volume of transmitted data is hence independent of the number of samples n. This is especially important in the intended GWAS setting, since here we can achieve $n \gg m$ by pre-filtering the SNPs (i. e., features) before carrying out the PCA (Li et al, 2016; Londin et al, 2010). Moreover, k is small (typically, $k = 10$ is used for GWAS PCA), which implies that the additional factor k in the complexities of $\mathcal{D}$ and $\mathcal{N}$ can be neglected. Therefore, using the suggested scheme is preferable over sending the eigenvectors to the aggregator for orthonormalization both in terms of privacy and expected transmission cost. (In practice, it is advisable to perform the orthonormalization only at the end). Guo et al (2012)'s algorithm has a complexity of $\mathcal{D} = \mathcal{O}(I \cdot S \cdot k)$ and $\mathcal{N} = \mathcal{O}(I \cdot S)$ per eigenvector. The use of randomized SVD additionally partially removes the dependency of the algorithm from the number of SNPs/features which can be quite large in practice. (The worst case complexity class does not change due to the first iterations). Additionally, only a few iterations of the true feature eigenvectors are transmitted. Therefore, randomized SVD is preferable in terms of privacy and transmission cost.

4.6 Summary

To conclude this section, we provide a brief summary of the main points and introduce a naming scheme for the configurations evaluated in Section 6. We

presented federated vertical subspace iteration with random (RI-FULL) initialization. To avoid the sharing of the sample eigenvector matrix, we introduced federated Gram-Schmidt orthonormalization (FED-GS) which can be run at every iteration, but should be run only at the end. In order to speed up the computation in terms of communication rounds, we suggest to use a modified version of the approximate algorithm (AI-ONLY) by Balcan et al (2014) as an initialization strategy for federated subspace iteration (AI-FULL). To reduce the transmitted data volume and the sharing of the feature eigenvectors, we suggest to use federated randomized subspace iteration (RI-RAND), which calls the regular federated subspace iteration as a subroutine for initialization and as a subroutine for the computation of the final eigenvectors based on a reduced representation of the data matrix. The second call of Algorithm 2 can be initialized with approximate initialization to speed up convergence (AI-RAND). GUO is the reference algorithm. We summarize the asymptotic communication costs in Table 2.

Table 2 Algorithm overview and complexity.

Algorithm(s)	Name	$\mathcal{D}$	$\mathcal{N}$
Algorithm 2	RI-FULL	$\mathcal{O}(I \cdot S \cdot k \cdot m)$	$\mathcal{O}(I \cdot S)$.
Algorithm 2+4	AI-FULL	$\mathcal{O}(I \cdot S \cdot k \cdot m)$	$\mathcal{O}(I \cdot S)$.
Algorithm 2+3	FED-GS	$\mathcal{O}(I \cdot (S \cdot k \cdot m + k^2))$	$\mathcal{O}(I \cdot S \cdot k)$.
Algorithm 5+2	RI-RAND	$\mathcal{O}(I' \cdot S \cdot k \cdot (m + I \cdot k))$	$\mathcal{O}((I + I') \cdot S)$.
Algorithm 5+2+4	AI-RAND	$\mathcal{O}(I' \cdot S \cdot k \cdot (m + I \cdot k))$	$\mathcal{O}((I + I') \cdot S)$.
Algorithm 4	AI-ONLY	$\mathcal{O}(S \cdot k \cdot m)$	$\mathcal{O}(S)$.
Guo et al (2012)	GUO	$\mathcal{O}(I \cdot S \cdot k \cdot m)$	$\mathcal{O}(I \cdot S)$.

5 Iterative leakage at the aggregator

In this section, we describe how the iterative process discloses the covariance matrix when using sufficiently many iterations. We first introduce the problem (Section 5.1) and then discuss how it can be addressed with the algorithms introduced Sections 4.3 and 4.4 above (Section 5.2). Practical results are illustrated in Section 6.7 below, using a simulation study.

5.1 Iterative leakage of the covariance matrix

Iterative leakage at the aggregator might disclose the entire covariance matrix during the execution of the algorithm, as many updates of the variables become available. Figure 3 visualizes the update process in power iteration, and the information used to reconstruct a single row of the covariance matrix at one iteration. Notably, the aggregated vector $\mathbf{H}_i$ becomes known in clear text at the aggregator in every iteration. The aggregator can store the sequence of vectors $\mathbf{H}_i$. In the following we will show, how it is possible to construct a system of linear equations which will leak the covariance matrix. For the sake of this description, we will assume the eigenvector $\mathbf{H}_i$ is updated as $\mathbf{H}_i = \mathbf{K}\mathbf{H}_{i-1}$, where $\mathbf{K} = \mathbf{D}^\top\mathbf{D}$ is the feature-by-feature covariance matrix of the federated data matrix $\mathbf{D}$, which are both unknown to the aggregator. This is equivalent to the two-step update from the aggregator's perspective, but improves the readability.

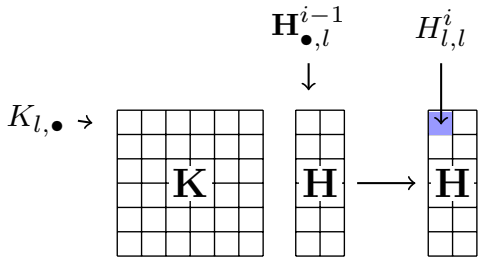


Fig. 3 Eigenvector update using the feature-by-feature covariance matrix.

Proposition 4 *Let* $\mathbf{D} \in \mathbb{R}^{n\times m}$ *be the data matrix and denote* $\mathbf{K} = \mathbf{D}^\top\mathbf{D}$ *the feature-by-feature covariance matrix, which is unknown to the aggregator. Let* k *be the number of eigenvectors retrieved. When applying federated subspace iteration, the aggregator can reconstruct* $\mathbf{K}$ *after* m/k *distinct eigenvector updates by solving a system of linear equations of the form* $\mathbf{K}_{l,\bullet}\mathbf{A} = \mathbf{b}$ *for each row* $\mathbf{K}_{l,\bullet}$ *of* $\mathbf{K}$*, where* $\mathbf{A} \in \mathbb{R}^{m\times m}$ *and* $\mathbf{b} \in \mathbb{R}^m$ *are known parameters.*

Proof Let $\mathbf{K}_{l,\bullet}$ denote a row of the covariance matrix $\mathbf{K} \in \mathbb{R}^{m\times m}$. First, we show how series $(\mathbf{H}^i_{\bullet,1})_{i=1}^m$ of m updates of the first eigenvector can be used to retrieve the row $\mathbf{K}_{l,\bullet}$ of $\mathbf{K}$. Susequently, we show that m/k updates are sufficient if all k eigenvectors are used.

Since $\mathbf{K}_{l,\bullet}$ is a row vector of length m, one needs m equations, which can be derived from m consecutive updates of the column vector $\mathbf{H}^i_{\bullet,1}$. The aggregator can store the consecutive updates of $\mathbf{H}^i_{\bullet,1}$ and, for each i, store an equation of the form $\mathbf{K}_{l,\bullet}\mathbf{H}^{i-1}_{\bullet,1} = H^i_{l,1}$. After m iterations, the aggregator is able to formulate the following fully determined system of linear equations, given that the eigenvectors have not converged:

$$\mathbf{K}_{l,\bullet}\begin{bmatrix}\mathbf{H}^0_{\bullet,1} & \cdots & \mathbf{H}^{m-1}_{\bullet,1}\end{bmatrix} = \begin{bmatrix}H^1_{l,1} & \cdots & H^m_{l,1}\end{bmatrix}$$

In order to reduce the number of required iterations, the aggregator can use all vectors in $\mathbf{H}$ to formulate the linear system and thereby divide the number of required iterations by k:

$$\mathbf{K}_{l,\bullet}\overbrace{\begin{bmatrix}\mathbf{H}^0_{\bullet,1} & \cdots & \mathbf{H}^0_{\bullet,k} & \cdots & \mathbf{H}^{\frac{m}{k}-1}_{\bullet,1} & \cdots & \mathbf{H}^{\frac{m}{k}-1}_{\bullet,k}\end{bmatrix}}^{\mathbf{A}} = \overbrace{\begin{bmatrix}H^1_{l,1} & \cdots & H^1_{l,k} & \cdots & H^{\frac{m}{k}}_{l,1} & \cdots & H^{\frac{m}{k}}_{l,k}\end{bmatrix}}^{\mathbf{b}}$$

The rows of $\mathbf{K}$ can be computed simultaneously, by forming a system for all $\mathbf{K}_{l,\bullet}$ at the same time. Therefore, in theory, this means that after m/k iterations, one has the full system and can solve it as

$$\mathbf{K}_{l,\bullet} = \mathbf{b}\mathbf{A}^{-1}, \tag{6}$$

which completes the proof of the proposition. □

These theoretical results require to invert $\mathbf{A}$, which may pose a problem in numerical applications, especially once the $\mathbf{H}^i$ grow large. By using a linear least squares solver, the inversion of the matrix $\mathbf{A}$ can be prevented at the cost of possibly sub-optimal solutions. Furthermore, in practice, more care needs to be taken when constructing the system, because, once converged, the eigenvectors do not provide a new equation to be added to the system anymore and hence lead to a singular system. In Section 6.7, we show that our approach works on small data.

5.2 Mitigation strategies

Recall that we claimed that Algorithms 4 and 5 improve the privacy of subspace iteration. After having established that the full global covariance matrix can be reconstructed after sufficiently many iterations, it becomes clear that reducing the number of iterations makes this attack more difficult. Algorithm 4 achieves this by using a better initial eigenvector guess and thus reduces the number of iteration until convergence. The randomized Algorithm 5 shares the initial eigenvector updates, but then shares only proxy eigenvectors, whose entries do not correspond to real features in the data, effectively reducing the number of useful iterations for the aggregator to a constant number I'. Therefore, these algorithms provide a algorithmic privacy improvement over previous solutions.

The attack approach described above is possible even when secure multiparty computation (SMPC) (Cramer et al, 2015) is used, as the aggregated updates still become available in clear text at the aggregator. SMPC does however prevent the disclosure of the local covariance matrices, so using it is beneficial in

truly federated implementations of this algorithm. Apart from the approximate algorithm, which uses SVD as an aggregation strategy, all algorithms are trivially compatible with secure aggregation as employed according to Cramer et al (2015). Naturally, perturbation techniques like differential privacy (Balcan et al, 2016) can be used to prevent the presented attack at the cost of decreased result accuracy. However, the high dimensionality m of the data might prove prohibitive, as the noise scales with m.

6 Empirical evaluation

6.1 Test datasets

To evaluate our federated PCA algorithm, we used three publicly available datasets: chromosome 1 and 2 from a genetic dataset from the 1000 Genomes Project (The 1000 Genomes Consortium, Auton, A., 2015), as well as the MNIST database of handwritten digits (LeCun et al, 2005) (Table 3). The two genetic data sets contain data for 2502 individuals (samples). After applying standard pre-processing steps (MAF filtering, LD pruning) we created 3 data set versions for each chromosome, with 100 000, 500 000 and >1 000 000 SNPs, respectively. MNIST contains 60 000 grayscale images of handwritten numerals (samples), each of which has 784 pixels (features). This data set was split into 5 and 10 equal chunks. To the best of our knowledge, publicly available genetic data sets with large numbers of patients are not readily available. However, although motivated by federated GWAS, our federated SVD algorithm is actually generically applicable. The experiments on MNIST demonstrate its usefulness for a more general audience. The MNIST data set is available at http://yann.lecun.com/exdb/mnist/, the genetic data can be obtained from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/.

Table 3 Datasets used in the study.

Dataset	Samples	Features
MNIST	60 000	784
1000 Genomes – Chrom. 1	2502	100 000
1000 Genomes – Chrom. 1	2502	500 000
1000 Genomes – Chrom. 1	2502	1 069 419
1000 Genomes – Chrom. 2	2502	100 000
1000 Genomes – Chrom. 2	2502	500 000
1000 Genomes – Chrom. 2	2502	1 140 556

6.2 Compared methods

We compare several configurations against each other: Federated subspace iteration with random initialization (RI-FULL), federated subspace iteration with approximate initialization (AI-FULL), federated randomized subspace iteration with random initialization (RI-RAND), and federated randomized iteration using approximate SVD for the reduced data matrix (AI-RAND). Federated subspace iteration which employs federated orthonormalization in every round (FED-GS) is extremely communication inefficient by adding $2k$ additional rounds per iteration which has been proven a major bottleneck in preliminary studies. Therefore, we omit this algorithm from the empirical evaluation, because the practical use for SVD is limited. We compare ourselves to the algorithm presented by Guo et al (2012), denoted GUO, as they present a solution which omits the covariance matrix and a way to deal with vertical data

partitioning. However, GUO shares the right singular vectors $\mathbf{G}$ and all updates of the left singular vectors $\mathbf{H}$ with the aggregator, which should be avoided in federated GWAS as emphasized in Section 4 and Section 5. Furthermore, as the number of features grows large, the transmission cost increases. We tested other configurations, including the use of approximate initialization for randomized PCA, but excluded them in this article as they did not bring a gain in performance in practice. For all compared methods, we set the convergence criterion in eq. (5) to $\epsilon = 10^{-9}$, which corresponds to a change of the angle between two consecutive eigenvectors updates of about 0.0026 degrees. Note that this angle does not equal the angle w. r. t. centrally computed eigenvectors, which we used as a test metric for measuring the quality of the compared methods (cf. next subsection).

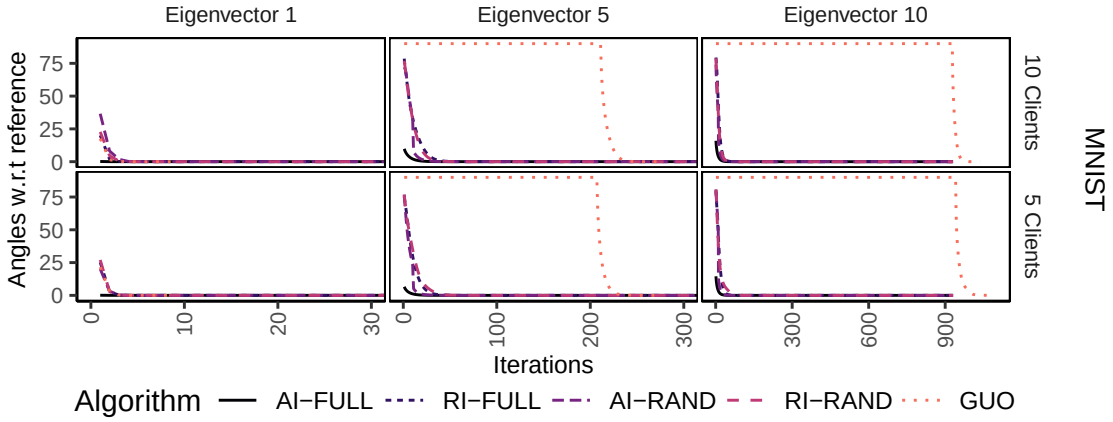


Fig. 4 Angles between selected reference eigenvectors and the federated eigenvectors on chromosome 1 and 2, as well as for MNIST. The omitted eigenvectors show similar behaviors.

6.3 Test metrics

For measuring the quality of the compared methods, we computed the angles between the eigenvectors obtained from a reference implementation of a centralized PCA and their counterparts computed in a federated fashion. An angle of 0 between two eigenvectors of the same rank is the desired result. As a reference, we chose the version implemented in `scipy.sparse.linalg`, which internally interfaces LAPACK. The amount of transmitted data is estimated by calculating the number of transmitted floats and multiplying it by a factor of 4 bytes (single precision IEEE 754). We choose this metric to remain agnostic with respect to the transmission protocol. Times measures are wall clock times using Python's `time` module. We chose to measure the runtime for matrix operations only, as they are the most important contributor to the overall runtime apart from communication related runtime.

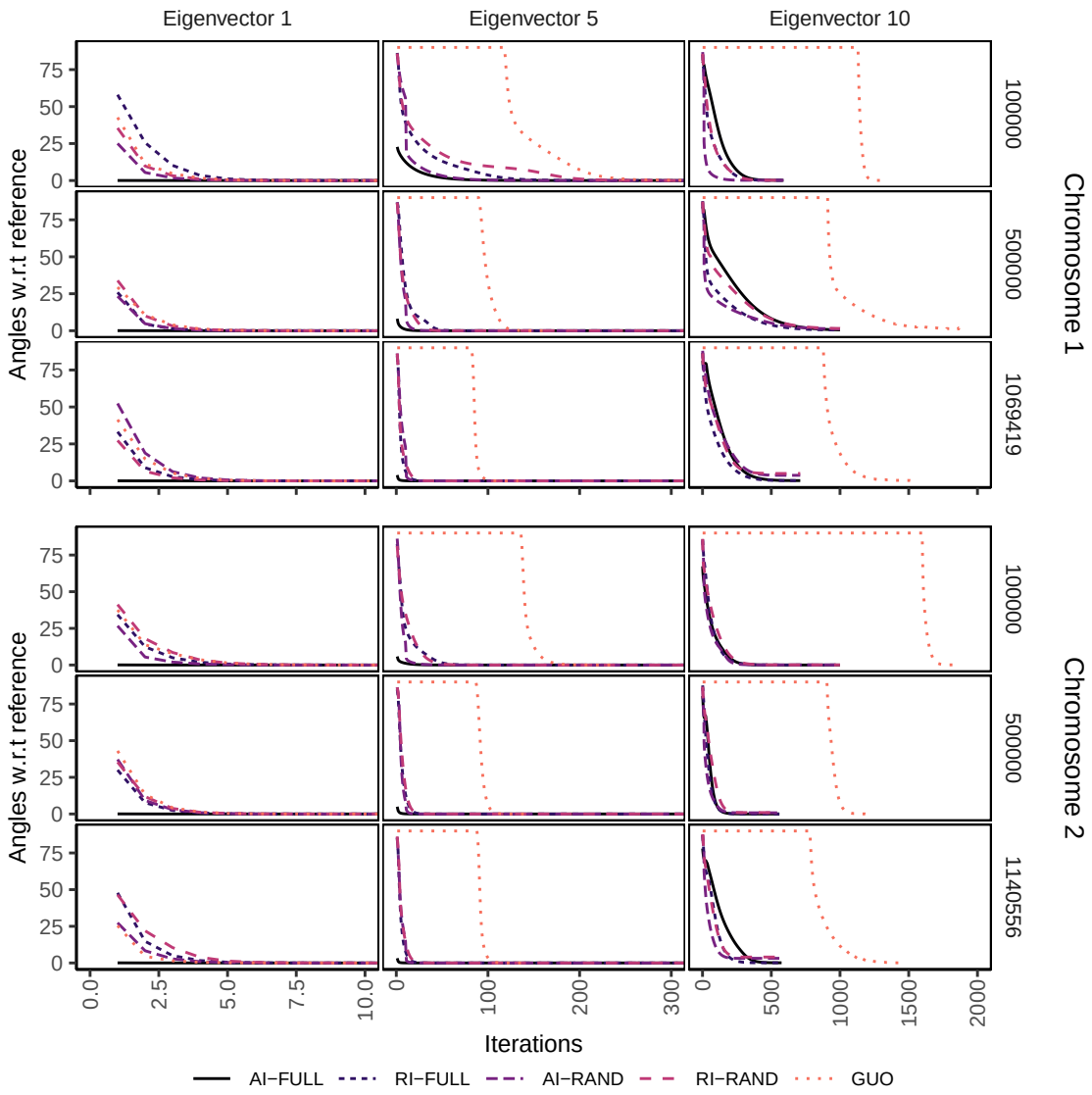


Fig. 5 Angles between selected reference eigenvectors and the federated eigenvectors on chromosome 1 and 2, as well as for MNIST. The omitted eigenvectors show similar behaviors.

6.4 Implementation, availability, and hardware specifications

All methods except the web interface are written in Python, using mainly, but not exclusively `numpy` and `scipy`. They are available online at https://gitlab.com/roettgerlab/federatedPCA. The simulation tests were run on a compute server with 48 CPUs and 502 GB available RAM due to the size of the genetic data sets. A federated tool compatible with the FeatureCloud (Matschinske et al, 2021b) ecosystem (featurecloud.ai) is available on the platform. The corresponding source code can be found at https://github.com/AnneHartebrodt/fc-federated-pca. We also created an AIME report (Matschinske et al, 2021a) to promote accessibility of machine learning research (Hartbrodt, 2022).

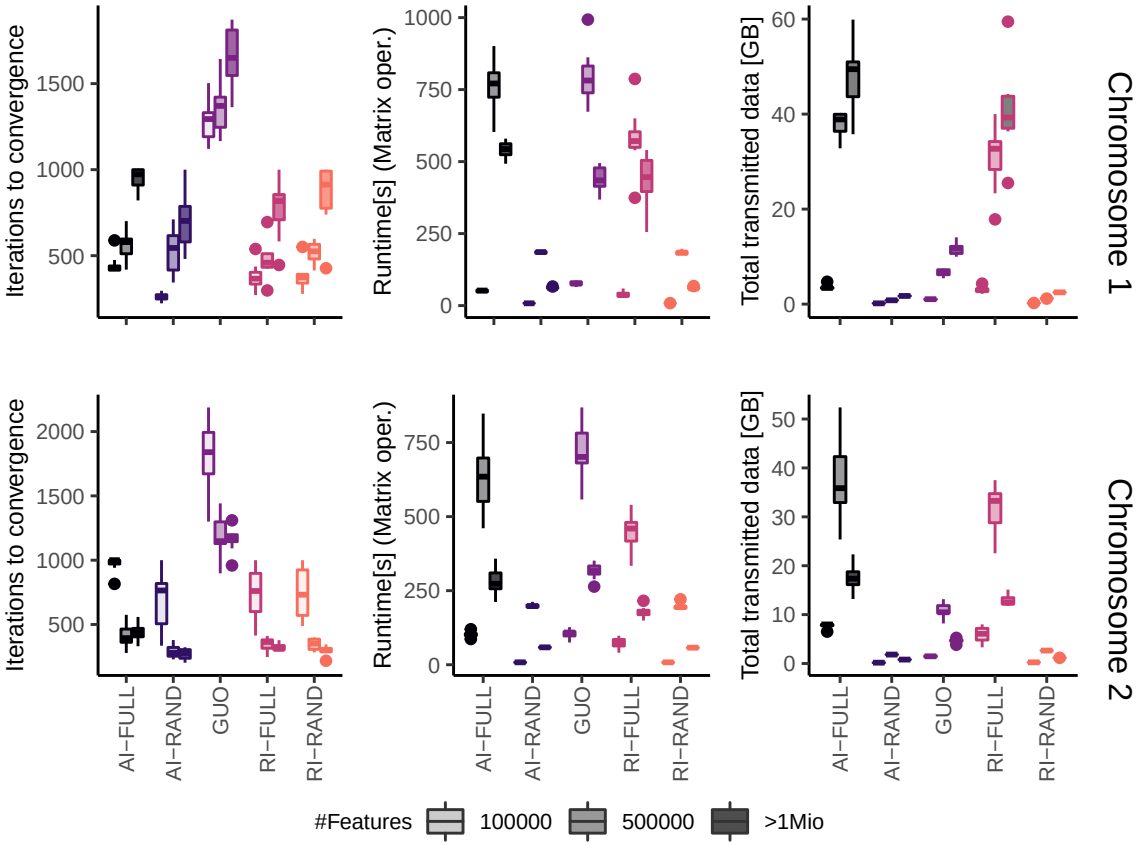


Fig. 6 Iterations, Runtime for matrix computations, and total transmitted data for each algorithm and each of the three data sets. The boxplots are grouped, the shading indicates the number of features ($0.1 * 10^6$, $0.5 * 10^6$, and roughly $1 * 10^6$).

6.5 Convergence behavior

To test the convergence behavior of the compared federated algorithms, we split the genetic data sets into 5 equally sized chunks; and the MNIST data set into 5 and 10 chunks. For every algorithm, we then recorded the angles between the first 10 eigenvectors w. r. t. the fully converged references at each iteration averaged across 10 runs.

Figures 4 to 7 show the results of the experiments. Note that, unlike the versions RI-FULL, AI-FULL, RI-RAND, AI-RAND of our algorithm, the competitor GUO computes the eigenvectors sequentially (i. e., eigenvector k has to converge before starting the computation of the eigenvector $k + 1$), which means that, for all but the first eigenvector, the plots for GUO start with a horizontal line. The most important result is that, for all algorithms, the eigenvectors perfectly converge to the reference eventually.

For low ranking eigenvectors, the approximate initialization speeds up the computation, because these eigenvectors can be well approximated and start with angles to the reference close to 0 (see Figures 4 and 5). The gain decreases in higher dimensions. Therefore, AI-RAND shows the overall best convergence behavior across all data sets and dimensions.

The number of sites does not influence the convergence behavior, as can be seen in the test with the MNIST data (Figures 4 and 7) where the convergence

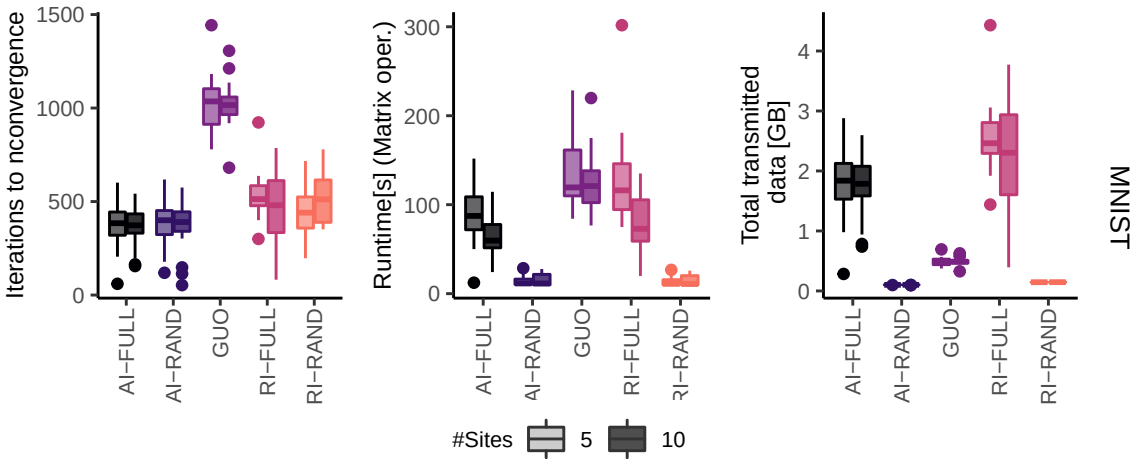


Fig. 7 Iterations, Runtime for matrix computations, and total transmitted data for each algorithm and each of the three data sets. The boxplots are grouped, the shading indicates the number of simulated clients (5, and 10).

curves and the required number of iterations are similar for the simulations with 5 clients and 10 clients. The transmitted data is shown only from the aggregator's perspective, to make the runs comparable. In federated SVD, the transmission cost scales with the number of clients.

The number of features/SNPs in the data does not show a clear trend. Although in the convergence plots in Figure 5 the larger data sets seem to converge more quickly, the overall number of iterations in fig. 6 does not confirm this trend. The reason for this is the dependence of the convergence speed on the eigengaps (the difference between two consecutive eigenvalues), an inherent property of each data set. The smaller the eigengap, the worse the convergence behavior. Table 4 shows the eigengaps for the eigengaps for Chromosome 2. The higher ranking eigengaps are generally quite small, indicating generally bad convergence for all datasets. Eigengap 8 for the data set containing 100000 SNPs specifically, is comparably even smaller which could explain the especially poor convergence.

Table 4 Eigengaps for Chromosome 2

SNPs	EG1	EG2	EG3	EG4	EG5	EG6	EG7	EG8	EG9
100000	15.14	16.02	1.19	3.95	0.9	1.87	0.32	0.05	0.12
500000	26.68	18.41	3.36	12.65	2.71	0.62	0.91	0.2	0.28
1140556	31.5	24.79	4.02	15.59	3.3	3.41	0.39	1.2	0.25

6.6 Scalability

To gauge the scalability of the methods with respect to runtime and transmission cost, we recorded the number of iterations until convergence, the runtime for matrix operations, and the estimated total amount of transmitted data for the selected algorithms. In Figure 6 we see that the amount of transmitted data is

the smallest for the randomized algorithms RI-RAND and AI-RAND, followed by GUO and with a significant distance AI-FULL and RI-FULL. AI-RAND and RI-RAND also spend the least time on the matrix operations which are the major contributor in to the runtime. The number of required iterations is the smallest for RI-RAND and AI-RAND, albeit being closely followed by AI-FULL and RI-FULL and GUO on the last place. Since the required iterations correspond to the number of communication steps, this factor significantly contributes to the overall runtime. Overall, RI-RAND and AI-RAND perform the best in all three measured categories. Generally, the bottleneck in federated learning is the number of transmission steps during the learning process, as this involves network communication. However, with increasing data set size like in the presented GWAS case, also the reduction in local runtime shows considerable impact on the overall runtime.

6.7 Covariance reconstruction experiment

We implemented the covariance reconstruction scheme presented in Section 5 and applied in on small example data, demonstrating its practicality. Using the breast cancer data and the diabetes data set from the UCI repository (Dua and Graff, 2017) which have 442 and 569 samples and 10 and 30 features respectively, we computed the centralized covariance matrix. Then we ran federated subspace iteration and recorded the eigenvector updates. After m/k iterations, we used the recorded matrices to form the linear system described in Section 5.1. Instead of inverting the matrix as described in eq. (6), we used a linear least squares solver (`scipy.lstsq`) to compute the solution. We then computed the Pearson correlation between the true and the reconstructed covariance matrix, with a perfect outcome of 1 (see Table 5) in negligible time.

Table 5 Reconstruction of covariance matrix.

Dataset	Samples	Features	Correlation	Time[s]
Breast Cancer	442	10	1	0.001
Diabetes	569	30	0.997	0.004

7 Conclusions and outlook

In this paper, we presented an improved federated SVD algorithm which is applicable to both vertically and horizontally partitioned data and, at the same time, increases the privacy compared to previous solutions.

Although our algorithm is motivated by the requirements of population stratification in federated GWAS, it is generically applicable. We proved that a first version of our algorithm is equivalent to a state-of-the-art centralized SVD algorithm and demonstrated empirically that it indeed converges to the centrally computed solutions. Subsequently, we improved the algorithm by including techniques from other federated and centralized algorithms to increase scalability and reduce the number of required communications.

There are two key advantages of our algorithm: Firstly, unlike in existing federated PCA algorithms, the sample eigenvectors remain at the local sites, due to the use of fully federated Gram-Schmidt orthonormalization, which improves the privacy of the algorithm. Secondly, the algorithm limits the amount of transmitted data (via smart initialization and data approximation) and is thereby more scalable and further prevents information leakage. In particular, the transmission cost of the randomized algorithm is not dependent on the number of samples and only partially dependent on the number of features.

In future work, we intend to further decrease the amount of transmitted data and the number of communication rounds further by only updating those eigenvectors or coordinates that have not converged yet or offering premature termination to avoid computing eigenvectors explaining little of the overall variance.

Chapter 5

Manuscript 4

Federated QR decomposition – algorithms, privacy, and applications

Federated QR decomposition – algorithms, privacy, and applications

Anne Hartebrodt
hartebrodt@imada.sdu.dk
University of Southern Denmark
Odense, Denmark

Richard Röttger
reottger@imada.sdu.dk
University of Southern Denmark
Odense, Denmark

ABSTRACT

Federated learning (FL) is a privacy-aware data mining strategy keeping the private data on the owners' machine and thereby confidential. The clients compute local models and send them to an aggregator which computes a global model. In hybrid FL, the local parameters are additionally masked using secure aggregation, such that only the global aggregated statistics become available in clear text, not the client specific updates. Federated QR decomposition has not been studied extensively in the context of cross-silo federated learning. In this article, we investigate the suitability of three QR decomposition algorithms for cross-silo FL and suggest a privacy-aware QR decomposition scheme which does not blatantly leak raw data. We investigate, if privacy can be gained via the application of this algorithm as a subroutine in federated PCA. In this context, we show that there is a critical data leak when using federated QR decomposition on upper triangular matrices, making secure aggregation an imperative for this algorithm. We show how the federated QR decomposition can be used to solve federated systems of linear equations.

CCS CONCEPTS

• **Theory of computation → Design and analysis of algorithms**; • **Security and privacy → Information accountability and usage control**.

KEYWORDS

QR decomposition, Federated Learning, Privacy

ACM Reference Format:
Anne Hartebrodt and Richard Röttger. 2022. Federated QR decomposition – algorithms, privacy, and applications. In *Proceedings of SIGKDD CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (KDD).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Federated learning has risen in popularity following the seminal article by McMahan et al., and possibly accelerated by a search for new privacy preserving data analysis techniques following the introduction of the GDPR in Europe. Federated learning is a data analysis paradigm, where the data stays on the data owners' machine and only aggregated parameters are exchanged with the other participants or a central aggregator. There are two main versions of federated learning, cross-silo federated learning and cross-device federated learning. Cross-device FL connects many devices with relatively low computational power, such as mobile phones or sensors in a learning process. The devices have access to limited data, for example for one user. Cross-silo federated learning, the learning paradigm adopted in this article, joins multiple data silos containing records for a larger group of participants together [11]. The federated setting adopted in this article is a type of hybrid federated learning which relies on secure-parameter aggregation (SMPC). This means the computations at the client sides are done on clear text, but the aggregation is performed using secure multiparty computation. Therefore, only the aggregated parameters become known to the aggregator, not the individual client's updates. The participants are honest-but-curious, following the protocol, but trying to infer as much information as possible from the updates they receive [6]. Since we "only" use secure aggregation and allow the disclosure of intermediate and final results, the advantage is, that we can directly chain together different algorithms into pipelines. Modern data analysis workflows rarely only use a single tool, therefore the use of secure aggregation allows reasonable privacy guarantees, without the need to develop new protocols for every workflow.

Recently, federated QR orthonormalization has been identified as a contributor to a more privacy preserving principal component analysis (PCA) algorithm [10]. The authors show that using federated QR orthonormalization for singular value decomposition allows the right, patient-associated singular vectors to remain private when using federated power iteration. QR decomposition is a versatile tool used for many more applications in linear algebra, including the solution of systems of linear equations [9]. Therefore, in this article we study this technique in more detail regarding its potential application in federated learning.

In centralized learning, the traditional machine learning setup, where all data is on a global server, three algorithms for QR factorization are available. They are based on Householder reflection, Givens rotation and the Gram-Schmidt procedure. The Householder algorithm is the most efficient for general applications, while Givens rotation is advantageous for sparse matrices and parallel computing architectures [16]. Gram-Schmidt orthonormalization is not used as much in practice due to numerical instabilities on special matrices [9]. However, a stabilized version of the algorithm exists and privacy considerations may take precedence over numerical issues. Consequently, it is interesting to evaluate the algorithms with regards to their suitability for federated learning. In an earlier article, an orthonormalization procedure based on the Gram-Schmidt

KDD, August 14–18, 2022, Washington DC

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

algorithm has been introduced [10], but it does not return the full decomposition. Therefore, it needs to be extended to return a full QR factorization. The other algorithms have not been explicitly introduced for cross-silo FL, therefore we have developed prototypes of their federated versions. We show that Householder reflection and Givens rotation have properties that render them unsuitable for federated computation, even when secure aggregation is used.

After presenting the privacy-aware QR decomposition algorithm, we demonstrate its application to PCA and for the solution of systems of linear equations: In order to further illustrate potential privacy leakages induced through federated QR decomposition, we study a federated PCA algorithm which has been introduced by Bai et al. and which we extend using the orthonormalization scheme developed in this article. Notably, the question we want to answer is whether the introduction of the federated QR scheme increases the privacy of the algorithm. Furthermore, to highlight the versatility and use for this federated procedure in other applications, we apply federated QR decomposition to compute linear regression. This could be used as an alternative solver for the federated linear regression computation suggested for instance in [14]. Our experiments demonstrate the same accuracy of the federated linear regression than standard standalone tools.

To summarize, our contributions are the following:

- We analyze the Householder, Givens and Gram-Schmidt algorithms for QR decomposition with respect to their suitability for hybrid federated learning, and argue, why Gram-Schmidt is the most private algorithm.
- We present the first descriptions of federated Householder reflection and Givens rotation.
- We provide a detailed description of the newly developed federated Gram-Schmidt based QR factorization.
- We present a modified version of a QR-based federated PCA algorithm to illustrate data leaks with special matrices.
- We show how to used federated QR decomposition to compute linear regression including the residuals and p-values.

The remainder of this manuscript is organized as follows: in section 2, the preliminaries, including the three centralized QR algorithms are introduced. Based on these descriptions, in section 3, we develop federated QR schemes for all algorithms, and a more detailed description for the most suitable QR algorithm, the federated Gram-Schmidt procedure (section 3.3). In section 4, we extend an existing PCA algorithm. We carefully analyze the exchanged parameters in section 4.2, and show that they can be used to reconstruct the client upper triangular input matrices. Thereby, we reconfirm that federated learning alone may not be privacy-aware and further privacy-enhancing techniques are required. In section 5 federated linear regression is presented. Section 7 introduces related work, followed by empirical results in section 6. Lastly, the results are briefly discussed in section 8. Section 9 concludes the work.

2 PRELIMINARIES

2.1 Data model and architecture

In this manuscript we assume matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ to be partitioned into a set of $s \in [S]$ partial data sets such that $\mathbf{A}_s \in \mathbb{R}^{n_s \times m}$. $[S]$ denotes the set of clients joining the learning system. This partitioning is referred to as horizontal. We assume all participants have a share of the data, and the ordering of the rows is known and fixed. We describe our algorithm using a star-like architecture. We expect the parameters to be masked using additive secure aggregation (cf. section 2.3), therefore we assume that peer-to-peer communication is possible via secure channels, regardless of the underlying architecture. This implies that our algorithms could be run on a fully decentralized architecture. The main reason for the choice of an aggregator-based architecture is the reduction in overall communication, because when using SMPC the clients do not have to transmit the intermediate parameters to all their peers, only to the aggregator.

2.2 Notation

Vectors and matrices are denoted in boldface, scalars in normal font. Matrices are noted in upper case letters and consist of column vectors which are noted in lower case letters. For instance, the matrix $\mathbf{A}^{n \times m}$ consists of m column vectors $\mathbf{a}_i$ where i is the index of the column. Sometimes we refer to columns and rows of a matrix as $\mathbf{A}_{\bullet,i}$ and $\mathbf{A}_{i,\bullet}$ respectively. Table 1 contains an overview over the most frequently used variables in this work.

Table 1: Notation table.

Syntax	Semantics
$[N] \subset \mathbb{N}$	index set $[N] = \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$
$S \in \mathbb{N}$	number of sites
$m \in \mathbb{N}$	number of features
$n \in \mathbb{N}$	total number of samples
$n^s \in \mathbb{N}$	number of samples at site $s \in [S]$
$\mathbf{A} \in \mathbb{R}^{m \times n}$	complete data matrix
$\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$	subset of data available at site $s \in [S]$
$\mathbf{U} \in \mathbb{R}^{n \times k}$	an orthogonal matrix with $\text{span}(\mathbf{U}) = \text{span}(\mathbf{V})$
$\mathbf{U}^s \in \mathbb{R}^{n \times k}$	the sub matrix of $\mathbf{U}$ available at sites $s \in [s]$
$\mathbf{Q} \in \mathbb{R}^{n \times k}$	an orthonormal matrix with $\text{span}(\mathbf{Q}) = \text{span}(\mathbf{V})$
$\mathbf{Q}^s \in \mathbb{R}^{n \times k}$	the sub matrix of $\mathbf{Q}$ available at sites $s \in [s]$
$\mathbf{R} \in \mathbb{R}^{k \times k}$	an upper triangular matrix

2.3 Secure aggregation

The secure aggregation scheme used in this work relies on the additive aggregation protocol used by [6]. It assumes honest-but-curious participants, i. e. all clients perform the computations following the protocol but try to infer as much information as possible from the exchanged parameters [19]. All s clients create $i = S$ random shares $x_{s,i}$ of their secret value x_s such that $\sum_i^S x_{s,i} \mod p = x_s$, where p is a large prime known to all participants. One can think of the "s" in $x_{s,i}$ as the source of the share and "i" the destination. All clients send the respective shares $x_{s,i}$ to the respective recipients i where the sum of the shares is computed as $\sum_s^S x_{s,i} = x_i$. None of the shares disclose any information on the original values. Lastly, the clients announce their aggregated secret share x_i, such that the global sum $x = \sum_i^S x_i \mod p$ of all private shares can be formed. This scheme is suitable for a cross-silo federated learning systems with reliable clients (i. e. they do not randomly drop out) and relatively few participants. Other secure aggregation schemes,

such as Shamir's protocol, which are more fault tolerant to client dropout [6] could be used instead without conceptual change of the algorithm.

2.4 Centralized QR decomposition

The QR decomposition is the factorization of a square matrix into a square orthonormal matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$.

$$\mathbf{A} = \mathbf{QR} \tag{1}$$

It exists also for non-square matrices (reduced QR decomposition) which is significantly more memory efficient if $n > m$. Three popular schemes exist for the computation of the decomposition, the Householder, Givens and Gram-Schmidt algorithm. In centralized systems, the Householder algorithm and Givens rotation are more popular, because they do not suffer numerical instability as the canonical version of Gram-Schmidt orthonormalization. Generally, Householder reflection is more efficient, and preferred unless the matrices are sparse or parallel compute architecture can be used [16]. See [9] for more details on the algorithms.

2.4.1 The Householder algorithm. The Householder reflection proceeds column-wise, setting all the elements below the diagonal to 0 using a Householder reflector. Therefore, it requires $m-1$ Householder reflections to form an upper triangular matrix $\mathbf{R}$ starting from a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$. A Householder reflector is defined as

$$\mathbf{Q}_u = I - \frac{2\mathbf{u}\mathbf{u}^\top}{\mathbf{u}\top\mathbf{u}}, \mathbf{u} \neq 0 \tag{2}$$

For each column vector $\mathbf{a}_i$ in matrix $\mathbf{A}$ the Householder reflection is computed using the following steps. First, $\mathbf{a}$ is normalized as $\mathbf{a}_i = \frac{\mathbf{a}_i}{||\mathbf{a}_i||_\infty}$ to avoid numerical overflow. Then the vector $\mathbf{u}$, i. e. the vector required for the construction of the Householder reflector, is computed by $\mathbf{u}_i = \mathbf{a}_i \pm ||\mathbf{a}_i||_2 \cdot \mathbf{e}$, where $\mathbf{e} = \begin{bmatrix}1 & 0 & \cdot & 0\end{bmatrix}^\top \in \mathbb{R}^{m\times 1}$ denotes a vector of length m containing a 1 in the first position and 0 otherwise. For ease of notation, the scaling factor $\frac{2}{\mathbf{u}^\top\mathbf{u}}$ is denoted β. The Householder reflection is computed implicitly to increase the computational performance. The resulting matrix $\mathbf{Q}_u\mathbf{A} = \mathbf{A} - \beta\mathbf{u}\mathbf{u}^T\mathbf{A}$ contains 0 in the column corresponding to vector $\mathbf{a}_i$. Algorithm 1 summarizes the a single Householder reflection. In the Householder QR algorithm this operation is performed for all vectors $\mathbf{a}_i$ of $\mathbf{A}$, transforming in each step only the sub matrix, which is not yet upper triangular by choosing the remaining reflection matrix as the identity matrix I. As the Householder reflection is the relevant component for the privacy considerations in section 3.1, we defer the full description of the Householder algorithm to the appendix (algorithm 7).

Algorithm 1: Householder reflection

Input: Data matrix $\mathbf{A}_s \in \mathbb{R}^{m\times m}$

$\bar{\mathbf{a}} = \frac{\mathbf{A}_{\bullet,i}}{||\mathbf{A}_{\bullet,i}||_\infty}$;

$\mathbf{u} = \bar{\mathbf{a}} \pm ||\bar{\mathbf{a}}||_2 \cdot \mathbf{e}$;

$\mathbf{Q}_i\mathbf{A} = \mathbf{A} - \beta\mathbf{u}\mathbf{u}^T\mathbf{A}$;

return $\mathbf{A}, \mathbf{u}$;

2.4.2 Givens rotation. Givens QR algorithm sequentially sets subdiagonal elements of the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ to 0 by multiplying the matrix with the corresponding "Givens matrix" [9]. After $\frac{n\cdot(m-1)}{2}$ operations all elements below the diagonal are 0 resulting an an upper triangular matrix $\mathbf{R}$. Through careful choice of the parameters in the Givens matrices, their product results in an orthogonal matrix $\mathbf{Q}$ which is the desired result.

A Givens matrix has the following form, where i and j are the indices of c and s.

$$J(i,j,c,s) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ & \ddots & & & & & \vdots \\ 0 & & c & \cdots & s & & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & & -s & \cdots & c & & 0 \\ \vdots & & & & & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$J(i,j,c,s)$ is orthogonal if $c^2 + s^s = 1$.

Let $\mathbf{A}$ be the matrix of interest and i and j with $i < j$ indices of the element to be set to 0. Then one can set

$$c = \frac{x_{i,i}}{\sqrt{x_{i,i}^2 + x_{j,i}^s}} \tag{4}$$

and

$$s = \frac{x_{i,j}}{\sqrt{x_{i,i}^2 + x_{i,j}^s}} \tag{5}$$

and compute the respective Givens matrix according to eq. (3).

Then $\mathbf{A}' = J(i,j,c,s)\mathbf{A}$ contains a 0 at position (i,j).The product of a Givens matrix with a general matrix can be computed efficiently, by updating only rows i and j of the matrix as

$$\mathbf{A}_{i,\bullet} = [ca_{i,1} + sa_{j,1}, ca_{i,2} + sa_{j,2}, \cdots, ca_{i,m} + sa_{j,m}] \tag{6}$$

and

$$\mathbf{A}_{j,\bullet} = [ca_{j,1} + sa_{i,1}, ca_{j,2} + sa_{i,2}, \cdots, ca_{j,m} + sa_{i,m}] \tag{7}$$

The full QR decomposition in a centralized setting is summarized in algorithm 2.

Algorithm 2: QR factorization using Givens rotation

Input: Data matrix $\mathbf{A}_s \in \mathbb{R}^{n\times m}$

foreach $i \in [1, ..., m-1]$ **do**

  foreach $j \in [i+1, ..., m]$ **do**

    $[s, c] \leftarrow$ `compute-givens-parameter()`;

    $\mathbf{A} = J(i,j,c,s)\mathbf{A}$;

    $\mathbf{Q} = J(i,j,c,s)\mathbf{Q}$;

$\mathbf{R} = A$;

$\mathbf{Q} = \mathbf{Q}^\top$ **return** $\mathbf{Q}, \mathbf{R}$

2.4.3 Gram-Schmidt orthonormalization. The Gram-Schmidt algorithm produces a orthonormal matrix $\mathbf{Q} = [\mathbf{q}_1 \dots \mathbf{q}_k]$ and an upper triangular matrix $\mathbf{R} = [\mathbf{r}_1 \dots \mathbf{r}_k]$ [3]. With a matrix $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_k] \in \mathbb{R}^{n\times m}$ of m linearly independent column vectors, the matrix $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_k] \in \mathbb{R}^{n\times m}$ of orthogonal column vectors is

computed, such that it has the same span as $\mathbf{A}$. Let $r_{i,j} = \mathbf{u}_j^\top \mathbf{a}_i / n_j$ and $n_j = \mathbf{u}_j^\top \mathbf{u}_j$ then

$$\mathbf{u}_i = \begin{cases} \mathbf{a}_i & \text{if } i = 1 \\ \mathbf{a}_i - \sum_{j=1}^{i-1} r_{i,j} \cdot \mathbf{u}_j & \text{if } i \in [k] \setminus \{1\} \end{cases}, \tag{8}$$

$$\mathbf{q}_i = \frac{\mathbf{u}_j}{||\mathbf{u}_j||} \tag{9}$$

$$\mathbf{r}_{j,i} = \begin{cases} \mathbf{q}_j \cdot \mathbf{v}_i & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases} \tag{10}$$

2.5 Centralized Singular Value Decomposition

Singular value decomposition (SVD) is a matrix decomposition frequently used in data mining applications. A matrix $\mathbf{A}$ is decomposed into two orthonormal matrices of singular vectors $\mathbf{U}$ and $\mathbf{V}$ and a diagonal matrix Σ containing the singular values in non-increasing order $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$ [9]. In the federated domain, SVD has been studied extensively, and multiple algorithms exist (e. g. [2, 4, 10]). Given the vertically distributed matrix $\mathbf{A}^s \in \mathbb{R}^{m \times n^s}$ with dimension $m \times n^s$ at sites s the federated singular value decomposition is defined as

$$\mathbf{A}^s = \mathbf{U}\Sigma\mathbf{V}^{s\top} \tag{11}$$

where $\mathbf{U}$ is the full left singular vector and $\mathbf{V}^s$ are the partial right singular vectors. The right singular vectors should not be shared due to potential privacy breaches [10].

2.6 Solution of systems of linear equations

In centralized computation, QR factorization can be used to compute the solution of systems of linear equations. Given a system $\mathbf{Ax} = \mathbf{b}$, one can compute $\mathbf{A} = \mathbf{QR}$. By setting $\mathbf{QRx} = \mathbf{b} \Leftrightarrow \mathbf{Rx} = \mathbf{Q}^{-1}\mathbf{b}$ the system can be solved efficiently because due to the orthonormality of $\mathbf{Q}$, $\mathbf{Q}^{-1} = \mathbf{Q}^\top$ and $\mathbf{y} = \mathbf{Q}^{-1}\mathbf{b}$ can be computed. This leaves to solve a system of the form $\mathbf{Rx} = \mathbf{y}$, which can be solved efficiently as $\mathbf{R}$ is an upper triangular matrix [3]. This can be used for instance for linear regression [14].

3 FEDERATED QR DECOMPOSITION

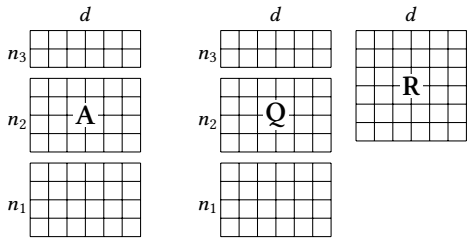


Figure 1: Schematic QR decomposition with 3 participants. A and Q remain private. R is known to all participants.

In this section, we describe and analyse approaches to federate QR factorization. To our knowledge, there exist no deceptions of federated versions of the Householder reflection or Givens rotation-based algorithms. Therefore, we first provide descriptions of the federated algorithms and demonstrate that they are not suitable for the chosen federated setting. Lastly, we describe a novel, extended Gram-Schmidt algorithm which also returns the upper triangular matrix $\mathbf{R}$. Recall that we assume the data $\mathbf{A} \in \mathbf{R}^{n \times m}$ to be partitioned row-wise into chunks $\mathbf{A}^s \in \mathbf{R}^{n_s \times m}$. The goal of federated QR decomposition is to compute $\mathbf{Q}^s$ and $\mathbf{R}$ such that $\mathbf{A}^s$ and $\mathbf{Q}^s$ stay private, meaning the raw data does not leave site s and $\mathbf{Q}$ can only be computed at s. $\mathbf{R}$ is common to all sites.

3.1 Federated Householder algorithm

We describe a straightforward algorithm for a federated Householder reflector. This subroutine could be used to compute the full QR decomposition in a federated manner. Let t_s be the row index set of $\mathbf{A}_s$ at site s.

Algorithm 3: Federated Householder reflection Client-side computations are marked in gray.

Input: Data matrix $\mathbf{A}_s \in \mathbb{R}^{n \times m}$
// t_s is the index set for the rows of $\mathbf{A}^s$
// Compute the global max element $||\mathbf{A}_{\bullet,i}||_\infty$
1 $m^s \leftarrow$ send-to-aggregator($||\mathbf{A}_{t_s,i}||_\infty$);
2 $||\mathbf{A}_{\bullet,i}||_\infty = \max_{s \in [S]} m^s$;
3 // Compute the local portion of $\mathbf{u}$
4 $\bar{\mathbf{a}}^s \leftarrow \frac{\mathbf{A}_{t_s,i}}{||\mathbf{A}_{\bullet,i}||_\infty}$;
5 $\mathbf{u}^s \leftarrow \bar{\mathbf{a}^s} \pm ||\bar{\mathbf{a}^s}||_2 \cdot \mathbf{e}$;
6 // Oracle step: $\mathbf{u}^s$ are stacked to compute $\mathbf{u}\mathbf{u}^\top$
7 $\mathbf{u} \leftarrow$ stack-vertically($[\mathbf{u}_1, \cdots \mathbf{u}^s]$) ;
8 // Update A
9 $\mathbf{A}^s = \mathbf{Q}_i^s \mathbf{A}^s = \mathbf{A}^s - \beta \mathbf{u}\mathbf{u}^T \mathbf{A}^s$;
10 **return** $\mathbf{A}^s$, $\mathbf{u}$;
11

In algorithm 3 we describe a federated householder reflector. The algorithm proceeds column wise. Initially, the global infinity norm $||\mathbf{A}_{\bullet,i}||_\infty$ is computed as the max over all local infinity norms (lines 1 to 2). Then, the clients locally compute $\mathbf{u}$ (lines 4 to 5). In order to compute the Householder reflector, the clients send their partial vectors $\mathbf{u}^s$ to the aggregator (line 7). We call this step an "oracle step" to indicate that under the chosen secure computation paradigm, the aggregation itself cannot be performed privately. Finally, at the clients, the reflection is performed (line 9).

Ad-hoc, this naive federated implementation of the procedure would take three communication rounds per column vector, one for the computation of the maximal element, one for the computation of the norm and one for the computation of the reflector.

In the federated setting, the computation of the Householder reflector itself is immediately problematic regarding the confidentiality of the data. Recall that algorithm relies on the computation of the outer product of $\mathbf{u}$ which is a direct transformation of the original column vectors of $\mathbf{A}$. In step 6, we call this operation an oracle step because it cannot be performed using the SMPC scheme we choose. Furthermore, even if secure multiplication is used, this "summary statistics" constitutes a privacy breach because the diagonal of $\mathbf{u}\mathbf{u}^\top$ contains the squared entries of $\mathbf{u}$. If $\mathbf{u}$, and $||\mathbf{a}||_\infty$ are

known, then the original vector $\mathbf{A}_{\bullet,i}$ can be reconstructed. Therefore, it is not straightforward to privately compute the Householder transform using hybrid federated learning with secure aggregation. Knowledge of the procedures allows the reverse engineering of the data. This can potentially be prevented by performing the entire computation under homomorphic encryption, or SMPC which allows the evaluation of arbitrarily complex circuits. When using SMPC, it would not be sufficient to compute the outer product securely, the intermediate parameters cannot become known to any of the computing parties. Based on these violation of our privacy demands, we exclude federated Householder reflection from any further considerations.

3.2 Federated Givens rotation

In this section we describe a direct translation of a Givens rotation to a federated setting. Again, we only describe the relevant subroutine which would allow the implementation of the complete QR decomposition, albeit inefficiently. Realistically, one would choose a parallelized version of the operator.

Algorithm 4: QR factorization using Givens rotation
Client-side computations are marked in gray.

Input: Data matrix $\mathbf{A}_s \in \mathbb{R}^{n \times m}$

/* Perform local precomputations, setting all possible elements to 0, send all non-zero indices to the aggregator */

1 **foreach** $i \in [1, ..., m-1]$ **do**
2 **foreach** $j \in [i+1, ..., m]$ **do**
   // Compute c and s, using values from two clients $k_1, k_2 \in [S]$
3 send-to-client(i, j);
4 $x_{i,i} \leftarrow$ send-to-aggregator$(a^{k_1}_{\mathbf{a}_{i,i}})$;
5 $x_{j,i} \leftarrow$ send-to-aggregator$(a^{k_2}_{\mathbf{a}_{j,i}})$;
6 $c = \frac{x_{i,i}}{\sqrt{x^2_{i,i}+x^s_{j,i}}}$;
7 $s = \frac{x_{i,j}}{\sqrt{x^2_{i,i}+x^s_{i,j}}}$;
8 $\mathbf{A}^s = J(i, j, c, s)\mathbf{A}^s$;
9 $\mathbf{Q}^s = J(i, j, c, s)\mathbf{Q}$;
10
11 $\mathbf{R} = \mathbf{A}$;
12 $\mathbf{Q} = \mathbf{Q}^\top$ **return** $\mathbf{Q}, \mathbf{R}$

Algorithm 4 summarizes the federated procedure described in the following. As precomputations, the clients perform Givens rotations to set all elements to 0 which only depend on their data. Then, the clients communicate all remaining non-zero indices below the diagonal to the aggregator. Setting an element to 0 requires only two rows i, and j to be manipulated. The clients associated with these rows are called k_1 and k_2. In the main loop, the aggregator announces the current i and j to the current clients k_1 and k_2 (line 3). Client k_1 and k_2 compute and announce the Givens parameters s and c in collaboration with the aggregator (lines 6 to 7). This is an "oracle step", as this implies the communication of x_i and x_j and is not trivially to compute using secure addition. The aggregator announces c and s to k_1 and k_2 and the clients update $\mathbf{R}$ and $\mathbf{Q}$. The broadcast can be combined with the new index broadcast (line 3) if applicable. Lines 3 to 9 are repeated until all elements below the diagonal are 0.

The naive implementation of this procedure would require in the order of $N = O(\frac{2 \cdot n \cdot (m-1)}{2})$ transmission rounds. Each element, would required an index broadcast and a Givens parameter broadcast. The procedure can be parallelized to zero out $\frac{n}{2}$ elements per round [16], reducing the communication complexity to $O(n)$.

However, there is a critical privacy breach when using Givens rotations. Recall that we assume the data to partitioned into s partitions $A_s \in \mathbb{R}^{n_s \times m}$. Assuming rows i and j are located in silo S_1 and S_2 respectively, the aggregator can compute the values x_i and x_j using c and s (cf. eq. (6), eq. (7)). Even, if c and s are computed using SMPC and P2P communication (so that the aggregator does not gain knowledge of the parameters), x_i and x_j can be reconstructed at the current clients k_1 and k_2. In order to prevent this breach, the whole algorithm would have to be performed under encryption, such that S_1 does not gain access to the intermediate matrices A'_s. These considerations render this algorithm unsuitable for hybrid federated learning with secure parameter aggregation. This leaves the Gram-Schmidt algorithm as the final possible algorithm.

3.3 Federated Gram-Schmidt Algorithm including the Computation of R

Based on the algorithm described in [10], where the authors showed that the orthogonal matrix $\mathbf{Q}$ can be computed solely based on the exchange and aggregation of vector norms and co-norms, we extend the algorithm such that the $\mathbf{R}$ matrix can be computed simultaneously. This can be done without further communication steps in comparison to the previously presented method. The main modifications are that the orthonormal vectors in $\mathbf{Q}$ need to be computed right away in order to compute the inner product of $\mathbf{q}_l \mathbf{a}_{i-1}$ contained in the matrix $\mathbf{R}$ at position $l, i-1$. Note, that the procedure also requires the orthogonal vectors $\mathbf{u}_i$.

Therefore, we develop a detailed description of a federated Gram-Schmidt orthonormalization procedure (see Algorithm 5). First, the global vector norm n_i of $\mathbf{u}_i$ is calculated by computing the local vector norms n^s_i at the clients and aggregating them at the central server (lines 2 to 5). The main loop starts at index $i = 2$ and proceeds in 4 stages. Let $\mathbf{R}$ be the upper triangular matrix completed up to vector $i \in d$. First, $\mathbf{e}^s_{i-1}$, is computed by dividing $\mathbf{u}_{i-1}$ through the global norm (lines 7 to 8). Then, the $i-1$st local column $\mathbf{r}^s_{l,i-1}$ of $\mathbf{R}$ is computed as the inner product of the partially normalized vector $\mathbf{q}^s_{i-1}$ and the partial data column $\mathbf{a}^s$ (lines 9 to 10). Then the local residuals r^s_{ij} for vector i w. r. t. to the previous $i-1$ vectors are computed (lines 11 to 13). In stage 2, the two parameters $\mathbf{r}^s_{l,i-1}$ and r^s_{ij} are sent to the central server and aggregated via element-wise addition (lines 15 to 18) to form the global copy of $\mathbf{R}$ up until $i-1$. The global $\mathbf{r}_{l,i-1}$, and r_{ij} are returned to the clients, where the orthogonal vector $\mathbf{u}^s_i$ is computed (lines 20 to 22). In the last stage, the norm of the current vector $\mathbf{u}_i$, n_i is computed by summing up the local norms of $\mathbf{u}^s_i$ (line 23). The procedure is repeated for all d vectors of $\mathbf{A}$. After exiting the main loop, the last column of $\mathbf{A}$ is computed, and the partial orthonormal matrices $\mathbf{R}$ and $\mathbf{Q}^s$ are

returned (lines 24 to 29). This procedure is equal to the centralized Gram-Schmidt algorithm because the vector inner products can be computed exactly in a federated fashion.

Algorithm 5: Federated Gram-Schmidt. Client-side computations are marked in gray.

Input: Data matrices $\mathbf{A}_s \in \mathbb{R}^{n_s \times d}$ at sites $s \in [S]$.
Output: Partial matrices $\mathbf{Q}_s$ and full matrix $\mathbf{R}$ at sites $s \in [S]$

1 // Compute norm of first orthogonal vector.
2 **for** $s \in [S]$ **do**
3 $\quad \mathbf{u}_1^s \leftarrow \mathbf{a}_1^s$;
4 $\quad n_1^s \leftarrow {\mathbf{u}_1^s}^\top \mathbf{u}_1^s$;
5 $n_1 \leftarrow \sum_{s=1}^S n_1^s$;
// Orthogonalize all subsequent vectors.
6 **for** $i \in [d] \setminus \{1\}$ **do**
7 $\quad$ **for** $s \in [S]$ **do**
$\quad\quad$ // Normalise to unit norm
8 $\quad\quad \mathbf{q}_{i-1}^s \leftarrow \mathbf{u}_{i-1}^s / \sqrt{n_{i-1}}$;
$\quad\quad$ // Compute relevant entries for $\mathbf{R}$
9 $\quad\quad$ **for** $l \in [i]$ **do**
10 $\quad\quad\quad \mathbf{r}_{l,i-1}^s \leftarrow \mathbf{q}_l^s \mathbf{a}_{i-1}^s$;
$\quad\quad$ // Compute client residuals for current vector.
11 $\quad\quad$ **for** $s \in [S]$ **do**
12 $\quad\quad\quad$ **for** $j \in [i-1]$ **do**
13 $\quad\quad\quad\quad r_{ij}^s \leftarrow {\mathbf{u}_j^s}^\top \mathbf{a}_i^s / n_j$;
14 $\quad$ // Aggregate residuals
15 $\quad$ **for** $j \in [i-1]$ **do**
16 $\quad\quad r_{ij} \leftarrow \sum_{s=1}^S r_{ij}^s$;
$\quad\quad$ // Aggregate R
17 $\quad\quad$ **for** $l \in [i]$ **do**
18 $\quad\quad\quad \mathbf{r}_{l,i-1} \leftarrow \sum_{s=1}^S \mathbf{a}_{l,i-1}^s$;
19 $\quad$ // Orthogonalize vector and compute norm.
20 $\quad$ **for** $s \in [S]$ **do**
21 $\quad\quad \mathbf{u}_i^s \leftarrow \mathbf{a}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s$;
22 $\quad\quad n_i^s \leftarrow {\mathbf{u}_i^s}^\top \mathbf{u}_i^s$;
23 $\quad n_i \leftarrow \sum_{s=1}^S n_i^s$;
24 **for** $s \in [S]$ **do**
$\quad$ // Compute last column of R
25 $\quad \mathbf{q}_d^s \leftarrow \mathbf{u}_d^s / \sqrt{n_{i-1}}$;
26 $\quad$ **for** $l \in [k]$ **do**
27 $\quad\quad \mathbf{r}_{ld} \leftarrow \mathbf{q}_d^s \mathbf{a}_l^s$;
28 $\quad \mathbf{Q}^s = [\mathbf{q}_1^s \cdots \mathbf{q}_d^s]$;
29 $\quad$ **Return** $\mathbf{Q}^s, \mathbf{R}$;
30

3.4 Privacy considerations

Recall that according to our privacy definition, privacte federated QR decomposition returns $\mathbf{Q}^s$ and $\mathbf{R}$ such that $\mathbf{A}^s$ and $\mathbf{Q}^s$ stay private, meaning the raw data does not leave site s and $\mathbf{Q}$ can only be computed at s. $\mathbf{R}$ is common to all sites.

Proposition 3.1. *At the end of federated Gram-Schmidt decomposition, the clients do not have access to more knowledge than their data matrices $\mathbf{A}_s$, the orthonormal partial matrices $\mathbf{Q}_s$, and the global matrices $\mathbf{R}$.*

Proof. We consider the case, where we have no knowledge of the type of matrix (for instance, whether it is sparse, or triangular) to be orthonormalized and analyze the knowledge at the aggregator. Let $\mathbf{A}^s = \mathbf{Q}^s\mathbf{R}$. At the end of the algorithm, the following knowledge is available at the aggregator (We only show the global aggregates, assuming that they are aggregated using secure addition):

- $[n_1, \cdots, n_d]$, the norms of $[\mathbf{u}_1, \cdots, \mathbf{u}_d]$
- $\mathbf{R}$ the upper triangular matrix

$$\begin{bmatrix} \mathbf{q}_1 \cdot \mathbf{a}_1 & \mathbf{q}_1 \cdot \mathbf{a}_2 & \cdots & \mathbf{q}_1 \cdot \mathbf{a}_d \\ 0 & \mathbf{q}_2 \cdot \mathbf{a}_2 & \cdots & \mathbf{q}_2 \cdot \mathbf{a}_d \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{q}_d \cdot \mathbf{a}_d \end{bmatrix} \tag{12}$$

- the upper triangular matrix of residuals

$$\begin{bmatrix} \mathbf{u}_1 \cdot \mathbf{a}_2 & \mathbf{u}_1 \cdot \mathbf{a}_3 & \cdots & \mathbf{u}_1 \cdot \mathbf{a}_d \\ 0 & \mathbf{u}_2 \cdot \mathbf{a}_3 & \cdots & \mathbf{u}_2 \cdot \mathbf{a}_d \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{u}_d \cdot \mathbf{a}_d \end{bmatrix} \tag{13}$$

- In particular, we do not have access to the matrices $\mathbf{U}^s$, $\mathbf{Q}^s$ or $\mathbf{A}^s$.

Since $\mathbf{q}_i = \frac{\mathbf{u}_i}{n_i}$, the total information available amounts to the information encoded in the $\mathbf{R}$ matrix. We hence have only access to one factor of the decomposition which does not allow us to find a unique solution to $\mathbf{A} = \mathbf{QR}$. We specified our privacy goal as keeping the input matrices $\mathbf{A}^s$ and the orthogonal matrices $\mathbf{Q}^s$ private, therefore the presented algorithm is private as per our definition. □

It should be noted that $\mathbf{R}$ does disclose information on the data in form of the feature covariance matrix:

$$\mathbf{A}^\top\mathbf{A} = \mathbf{R}^\top\mathbf{Q}^\top\mathbf{QR} = \mathbf{R}^\top\mathbf{R} \tag{14}$$

4 FURTHER PRIVACY INVESTIGATIONS

In this section, we apply federated QR factorization as a subroutine in federated PCA to reveal a privacy breach that can occur, if secure aggregation is not used, or if only 2 parties participate in the computation. The original algorithm uses QR factorization as the aggregation step [2]. This centralized procedure can be replaced by federated QR orthonormalization, presumably preventing the disclosure of the local summary statistics. The algorithm is mainly of academic interest, because more efficient schemes for PCA are available for star-like architectures. However, we will show, that knowledge of the procedure allows an honest-but-curious participant to exactly reconstruct the other participants' input data. Our

attack exploits the fact, that the input matrices are upper triangular and that we have full knowledge of the algorithm.

4.1 Algorithm

The algorithm [2] relies on sending a local $\mathbf{R}$ to the aggregator, where a secondary QR decomposition is performed (algorithm 6). We suggest centering the data globally prior to the computation of the matrix (line 1). This avoids having to account for inter site differences in mean later on. The next step is identical to the original: all the $\mathbf{R}$ matrices are computed at the clients (line 3). The original algorithm recursively merges the $\mathbf{R}$ matrices at a processor to form the updated $\mathbf{R}'$ matrix until only one matrix remains. By computing the QR decomposition of all clients' $\mathbf{R}$ matrices at once using federated Gram-Schmidt decomposition, sending $\mathbf{R}$ can be avoided. The federated QR algorithm returns $\mathbf{R}$ at all the clients, therefore the final SVD can be directly computed at the client. The clients can also compute the partial left eigenvectors as $U^s = A^s V$ (line 6).

Algorithm 6: Federated PCA using QR factorization [2]

Input: Data matrices $\mathbf{A}_s \in \mathbb{R}^{n_s \times m}$, # eigenvectors k.

1 $\mathbf{A}^s \leftarrow$ `federated-centering()` ;
2 **for** $s \in [S]$ **do**
  `// Compute local R at all clients`
3 $\mathbf{Q}_s, \mathbf{R}_s \leftarrow$ `orthonormalize`$(\mathbf{A}_s)$;
4 $[\mathbf{Q}^s], \mathbf{R} \leftarrow$ `federated-gram-schmidt`$([\mathbf{R}^1, \cdots \mathbf{R}^s])$;
5 $\mathbf{U}, \Sigma, \mathbf{V}^\top =$ `SVD`$(\mathbf{R})$;
6 $\mathbf{U}^s \leftarrow \mathbf{A}^s \mathbf{V}$;
7 **Return** $\mathbf{U}^k_s, \mathbf{V}^k$

4.2 Privacy considerations

In the original algorithm, the communication of $\mathbf{R}$ poses a problem: Let $A \top \mathbf{A}$ be the covariance matrix of the data. Using the fact that $\mathbf{Q}$ is an orthonormal matrix, $\mathbf{R}$ can be used to compute the local covariance matrices of the data and hence leaks information (eq. (14)). Therefore, this algorithm is no more private than sending the entire set of local eigenvectors to the next party. The advantage of algorithm 6 over its previous version is that it allows the computation of the global $\mathbf{R}$ without communicating the local $\mathbf{R}$ in clear text. The same can be achieved by using secure addition of the covariance matrices or computing the global $\mathbf{R}$ based on the data instead of $\mathbf{R}$. Nonetheless, we investigate this algorithm, because with close analysis it reveals a privacy breach if secure aggregation is not used or only two participants join. We show, that in this case the federated QR decomposition of upper triangular matrices is no more private than sending the upper triangular matrices themselves. The reason for this is the fact that the initial vector norm of the QR step is not technically an aggregate. We visualize the aggregation step in algorithm 6 in eq. (15), as it is the motivation for our investigation. To avoid ambiguity, we denote the resulting upper triangular matrix $\mathbf{S}$ with elements $s_{i,j}$. For the remainder of this section, we assume that secure aggregation is not used.

Proposition 4.1. *Let $\mathbf{R}^* = \begin{bmatrix}\mathbf{R}^1 & \mathbf{R}^2 & \cdots & \mathbf{R}^s\end{bmatrix}^\top$ be a vertical stack of upper triangular matrices, of which we want to compute the QR decomposition as $\mathbf{R}^* = \mathbf{QS}$. Denote $\mathbf{Q}^s = [\mathbf{u}^s_1, \mathbf{u}^s_2, \cdots \mathbf{u}^s_d]$ the block wise orthogonal matrices at sites s. It is possible to reconstruct all $\begin{bmatrix}\mathbf{R}^1 & \mathbf{R}^2 & \cdots & \mathbf{R}^s\end{bmatrix}$ as well as all $\begin{bmatrix}\mathbf{Q}^1 & \mathbf{Q}^2 & \cdots & \mathbf{Q}^s\end{bmatrix}$ when applying the federated QR algorithm on $\mathbf{R}^*$, given one knows that $\mathbf{R}^s$ are upper triangular.*

$$\mathbf{R}^* = \begin{bmatrix}\mathbf{R}^1\\ \mathbf{R}^2\\ \vdots\\ \mathbf{R}^s\end{bmatrix} = \begin{bmatrix}\begin{pmatrix} r^1_{11} & r^1_{12} & \cdots & r^1_{1d}\\ 0 & r^1_{22} & \cdots & r^1_{2d}\\ \vdots & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & r^1_{dd}\end{pmatrix}\\ \begin{pmatrix} r^2_{11} & r^2_{12} & \cdots & r^2_{1d}\\ 0 & r^2_{22} & \cdots & r^2_{2d}\\ \vdots & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & r^2_{dd}\end{pmatrix}\\ \vdots\\ \begin{pmatrix} r^s_{11} & r^s_{12} & \cdots & r^s_{1d}\\ 0 & r^s_{22} & \cdots & r^s_{2d}\\ \vdots & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & r^s_{dd}\end{pmatrix}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}^1\\ \mathbf{Q}^2\\ \vdots\\ \mathbf{Q}^s\end{bmatrix} \mathbf{S} \tag{15}$$

Proof. Let $\mathbf{R}^* = \begin{bmatrix}\mathbf{R}^1 & \mathbf{R}^2 & \cdots & \mathbf{R}^s\end{bmatrix}^\top$ be the matrix to be decomposed into $\mathbf{Q}$ and $\mathbf{S}$. Denote $\mathbf{R}^s$ and $\mathbf{Q}^s$ the partial matrices only available at site $s \in [S]$. Denote $[\mathbf{u}^s_1 \cdots \mathbf{u}^s_d]$ the partial orthogonal vectors at sites s. We show by induction on i that $\mathbf{R}^s$ and $\mathbf{Q}^s$ can be reconstructed at the aggregator based on the intermediate summary statistics exchanged during the execution of algorithm 5. Let $i = 1$. In the first step of the algorithm (lines 3 to 5, algorithm 5), when computing $n_1 = \sum_{s=1}^S \mathbf{r}^{s\top}_1 \mathbf{r}^s_1$ the clients disclose $(\mathbf{r}^s_{1,1})^2$ to the aggregator which can compute

$$\mathbf{u}_1 = [\sqrt{\mathbf{r}^1_{1,1}}, 0, \cdots, 0, \sqrt{\mathbf{r}^2_{1,1}}, \cdots, 0, \sqrt{\mathbf{r}^s_{1,1}}, 0, \cdots, 0]^\top. \tag{16}$$

Let now $i = 2$ and $j = 1$, the residuals $p^s_{2,1} \leftarrow \frac{\mathbf{u}^{s\top}_1 \mathbf{r}^s_2}{n_1}$ are computed and aggregated as $p_{2,1} = \sum_{s=1}^S p^s_{2,1}$ (line 13 and line 16). n_1 and $\mathbf{u}_1$ are known. We can compute $r^s_{1,1} = \mathbf{q}^{s\top}_1 s^s_{1,1}$ because $\mathbf{q}^s_1$ is orthonormal and only contains a single non-zero entry (line 9, (s corresponds to r in the algorithm description)). For the same reason, we can also compute $r^s_{1,2} = \frac{p^s_{2,1} \cdot n_1}{q_{1,1}}$ (line 13).

Finally, we compute $n_2 \leftarrow \sum_{s=1}^S n^s_2$, with $n^s_2 = \mathbf{u}^{s\top}_2 \mathbf{u}^s_2$ where $\mathbf{u}^s_2 \leftarrow \mathbf{r}^s_2 - \sum_{j=1}^{i-1} p_{21} \cdot \mathbf{u}^s_1$. This can be simplified to $\mathbf{u}^s_2 = \begin{pmatrix} r^s_{12} - p_{21} \cdot u^s_{11} \\ r^s_{22}\end{pmatrix}$, because only $u^s_{1,1}$ is non-zero. $r^s_{1,2}$, $p_{2,1}$ and $u^s_{1,1}$ are known, so $r^s_{2,2} = \sqrt{n^s_2 - (r^s_{1,2} - p_{2,1} \cdot u^s_{1,i})^2}$ can be computed, which in turn means $\mathbf{u}^s_2$ is known completely. At this point, $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{r}^s_1$ and $\mathbf{r}^s_2$ are known to the aggregator.

For the inductive step, we assume to have computed $\mathbf{Q}$ and $[\mathbf{R}^1 \cdots \mathbf{R}^s]^\top$ up to column $i-1$, we can compute column i. We set $j = i - 1$.

The residuals $p^s_{ij} \leftarrow \mathbf{u}^{s\top}_j \mathbf{r}^s_i / n_j$ are computed, and aggregated as $p_{ij} = \sum_{s=1}^S r^s_{ij}$. n_j, $\mathbf{r}_j$ and $\mathbf{u}_j$ are known for $j \in [i-1]$. We can

compute r^s_{ij} for $j \in [i-1]$ via successive variable substitution due to the fact that the $\mathbf{R}^s_i$ are upper triangular.

$$\begin{cases} r_{1,i} = \frac{p_{i,1} \cdot n_1}{u_{1,1}} \\ r_{2,i} = \frac{p_{i,2} \cdot n_2 - u_{1,2} \cdot r_{1,i}}{u_{2,2}} \\ \vdots \\ r_{j-1,i} = \frac{p_{i,j} \cdot n_j - \sum_{n=0}^{i} u_{n-1} \cdot p_{i,n-1}}{u_{n-1,n-1}} \end{cases} \tag{17}$$

Finally, we compute $n_i \leftarrow \sum_{s=1}^{S} n_i^s$, with $n_i^s = \mathbf{u}_i^{s\top}\mathbf{u}_i^s$ where $\mathbf{u}_i^s \leftarrow \mathbf{r}_i^s - \sum_{j=1}^{i-1} p_{ij} \cdot \mathbf{u}_j^s$ which can be rewritten as

$$\mathbf{u}_i^s = \begin{pmatrix} r^s_{1,i} - \sum_{j=1}^{i-1} p_{i,j} \cdot u^s_{1,j} \\ r^s_{2,i} - \sum_{j=1}^{i-1} p_{i,j} \cdot u^s_{2,j} \\ \vdots \\ r^s_{j,i} - \sum_{j=1}^{i-1} p_{i,j} \cdot u^s_{j,j} \\ r^s_{i,i} \end{pmatrix} \tag{18}$$

where $\mathbf{r}_{i,i}$ is the only unknown. $\mathbf{r}_{ji} = \sqrt{n_i^s - \sum_{j=1}^{i-1}(\mathbf{r}^s_{j,i} - p_{i,j} \cdot \mathbf{u}^s_{j,i})^2}$, which in turn means $\mathbf{u}^s_j$ is complete, because n_i, $p_{i,j}$ and $\mathbf{u}^s_j$ as well as $\mathbf{r}_{j,i}$ are known. □

When using general matrices, even with only two participants, and if SMPC is used, this attack is not possible, because the first vector norm summarizes more than one element. However, the previous application highlights that tracking the parameters during federated iterations could reveal more information on the input data than the participants intend, especially when the methods are fully traceable and do not involve randomized steps. In the case of sparse matrices, a partial column $\mathbf{a}^s_i$ which contains no entries, can be detected at the aggregator as the inner product in $\mathbf{R}$ would be 0. The problem, with the methods presented here, is that the knowledge of algorithmic procedure and the absence of random elements in the algorithm allow us to backtrack more information than intended, given an 'attack angle'.

5 SYSTEMS OF LINEAR EQUATIONS

To showcase the realistic use of our algorithm, we consider the application of federated orthonormalization for the solution of systems of linear equations. A popular use of QR decomposition is linear regression. For example, R's `lm()` function uses the QR algorithm by default [15]. Here, we demonstrate how it is possible to solve a system of linear equations of the form $\mathbf{Ax} = \mathbf{b}$ with only one further round of communication, based on the QR decomposition. This technique can be used to replace the solver implemented for instance in [14]. It does not require matrix inversion and is therefore more suitable for large scale matrices. Let $\mathbf{A}$ and $\mathbf{b}$ be partitioned into $\mathbf{A}^s$ and $\mathbf{b}^s$ respectively, and $\mathbf{x}$ the solution common to all sites. After the QR decomposition of the matrix $\mathbf{A}$, $\mathbf{Q}^s$ is known at the sites, and $\mathbf{R}$ is known at all sites and the aggregator. In order to compute $\mathbf{x}$, the clients have to send their vector inner product of $\mathbf{y}^s = \mathbf{Q}^\top\mathbf{b}^s$ to the aggregator which securely computes the global vector $\mathbf{y} = \sum_s^S$. The aggregator can directly compute $\mathbf{x}$ by successive variable substitution and share the result with the clients (see section 2.6). With one additional step, one can also compute p-values and r^2 statistics. The clients compute the sum of the squared residuals as $rss_s = \sum(\mathbf{A}_s\mathbf{x} - \mathbf{b})^2$ and the sum of the squared fitted values $mss_s = \sum \mathbf{A}_s\mathbf{b}$ and send them to the aggregator, which computes the global sums $rss = \sum_s^S rss_s$, and $mss = \sum_s^S mss_s$. For the p-value, the variance is computes as $\sigma = \frac{rss}{n-m-1}(\mathbf{R}^\top\mathbf{R})^{-1}$, and standard error as $SE = \sqrt{\sigma}$. Here, we exploit the fact that the covariance matrix can be expressed using $\mathbf{R}$ (eq. (14)). The T-statistic used to determine the p-value can be computed as $T = \frac{\mathbf{x}}{SE}$. For more details see [14] who provide a detailed description of the p-value calculation. r^2 can be computed as follows: $r^2 = \frac{mss}{mss+rss}$.

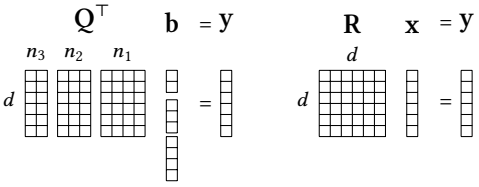


Figure 2: Schematic solution of a system of linear equations based on federated QR factorization of matrix A.

6 EXPERIMENTS & IMPLEMENTATION

We implement the QR decomposition scheme and a prototype for linear regression in python to show that they provide accurate results in practice. In this experimental study we use three example data sets from `sklearn` and Kaggle: the Pima Indians diabetes [8], WHO life expectancy [12] and fish market [7] data sets. We split the data sets horizontally in 5 chunks. We compute the baseline reduced QR decomposition using `scipy.linalg.qr`. As an error measure, we use the Frobenius norm between the centralized and federated $\mathbf{Q}$ and $\mathbf{R}$ matrices ($||\mathbf{Q}_c - \mathbf{Q}_f||_F$,$||\mathbf{R}_c - \mathbf{R}_f||_F$). For the linear regression, we use the `lm` function in `R` as it uses QR decomposition as its standard solver. As additional error measures we compute the sum of the absolute differences between the coefficients ($\sum_{s\in[S]} \mathbf{x}_c - \mathbf{x}_f$), r^2-values ($r_c^2 - r_f^2$), and p-values ($\sum_{s\in[S]}(p_c - p_f)$). The results of these experiments are summarized in table 2. The matrices, coefficients and r^2 values are identical, and there only minor variations in the $p-value$. The simulation code and example data are available online https://anonymous.4open.science/r/federated-qr-7D3F/.

Table 2: Results of the experiments

Dataset	Diabetes	WHO	Fish market
$\|\|\mathbf{Q}_c - \mathbf{Q}_f\|\|_F$	$3.0e^{-14}$	$1.1e^{-14}$	$3.8e^{-13}$
$\|\|\mathbf{R}_c - \mathbf{R}_f\|\|_F$	$1.7e^{-14}$	$7.5e^{-7}$	$9.3e^{-12}$
$\mathbf{x}_c - \mathbf{x}_f$	$2.23e^{-11}$	$4.5e^{-12}$	$3.04e^{-11}$
$\sum_d(p_c - p_f)$	0.003	0.019	0.029
$r_c^2 - r_f^2$	$1.4e^{-17}$	0	$2.5e^{-15}$

7 RELATED WORK

Federated QR algorithms have been suggested mainly in the field of peer-to-peer networks relying on the PushSum algorithm and gossiping [17, 18, 20]. While these schemes can be implemented in

a modern federated learning system, the assumptions governing FL make these algorithms unsuited. Notably, in cross-device FL, the client-to-client communication is assumed to be a bottleneck [11] and client-aggregator communication is preferred. Secondly, cross-silo FL assumes more data and higher compute power at the nodes, so local computational constraints do not impact the computations as severely. In medical systems, practitioners might want to avoid approximation errors at the cost of higher compute time [21].

8 DISCUSSION AND FUTURE DIRECTIONS

Based on the previous analyses, we have suggested a novel extended QR decomposition algorithm with clear privacy considerations, extensively analyzing all the options for the first time. Further, this work is backed by a detailed analysis of the other popular QR algorithms and exposing their weaknesses in particular with respect to privacy. As explained in section 3.1 and section 3.2, Householder reflection and Givens rotation have immediate drawbacks that make them unsuitable to hybrid federated learning where the parameters are securely aggregated, because it is possible to extract the original data from the parameters. This makes the presented federated Gram-Schmidt QR algorithm the only algorithm which does not trivially expose the original data under the assumed federated setting. We argued that the parameters revealed during the orthonormalization procedure contain no more information than the upper triangular matrix $\mathbf{R}$, and therefore fulfills our privacy specification of federated QR decomposition.

In this article, we assume a hybrid federated learning setup, where the global parameters become known in clear text. This means, the results may only partially translate to systems which rely on encrypting the entire learning process under homomorphic encryption or computing the whole algorithm using secure multiparty computation. These techniques are still expensive in practice [1, 5] but might be required to provide secure algorithms for Householder factorization and Givens rotation. If privacy is not a concern, detailed investigations of potential gains in transmission rounds would be required to find the most efficient QR scheme, most likely Givens algorithm according to our preliminary analysis.

The investigation of information leakage associated with the parameters exchanged during the federated QR orthonormalization spins a cautionary tale. We showed that it is possible to reconstruct the input matrices, if they are upper triangular, solely from the exchanged parameters, because the first aggregate is technically not an aggregate and triggers a revealing cascade. This means that with fewer than three parties, even the clients could reconstruct the other participants' matrices. We showed that for upper triangular matrices privacy breaches are possible. Therefore, further investigations on other special types of matrices will be required.

Another interesting application of federated QR decomposition is a recently suggested masking based PCA, which relies on orthonormal masks to hide the raw data from an aggregator [4]. In this scheme, QR factorization can be used to create the orthonormal masking matrices instead of trusting a third party server to generate these matrices.

9 CONCLUSION

This work presented federated implementations of three popular QR algorithms and thoroughly investigated them with respect to their privacy, if they are deployed in a hybrid federated system with secure parameter aggregation. We come to the conclusion, that only our suggested Gram-Schmidt QR decomposition is suitable, due to it's reliance on inner vector products. The use of the outer vector product in Householder factorization, introduces a trivial confidentiality breach. Likewise, a trivial privacy leak in Givens rotation makes this algorithm unsuitable for the chosen federated learning paradigm. We illustrate the versatility and power of the novel federated QR decomposition algorithm by demonstrating how it can be used to compute federated linear regression.

ACKNOWLEDGMENTS

The FeatureCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078. This pub- lication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] M Al-Rubaie, P.-Y. Wu, J M Chang, and S.-Y. Kung. 2017. Privacy-preserving PCA on horizontally-partitioned data. *EEE DSC 2017* (2017), 280–287.
[2] Zheng-Jian Bai, Raymond H. Chan, and Franklin T. Luk. 2005. Principal Component Analysis for Distributed Data Sets with Updating. In *Advanced Parallel Processing Technologies*, Jiannong Cao, Wolfgang Nejdl, and Ming Xu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 471–483.
[3] Robert A. Beezer. 2016. A second course in linear algebra. http://linear.ups.edu/scla/html/gfdl.html. [Online book; accessed 2022-01-31].
[4] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2019. Secure Federated Matrix Factorization. *CoRR* abs/1906.05108 (2019). arXiv:1906.05108
[5] Hyunghoon Cho, David J. Wu, and Bonnie Berger. 2018. Secure genome-wide association analysis using multiparty computation. *Nat Biotechnol* 36, 6 (2018), 547–551.
[6] Ronald Cramer, Ivan Bjerre Damg**r** ard, and Jesper Buus Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.
[7] Fish Market data set. 2022. https://www.kaggle.com/aungpyaeap/fish-market. Accessed: 2022-01-31.
[8] Pima Indians Diabetes Database. 2022. https://www.kaggle.com/uciml/pima-indians-diabetes-database. Accessed: 2022-01-31.
[9] William Ford. 2014. *Numerical Linear Algebra with Applications : Using MATLAB*. Elsevier Science & Technology.
[10] Anne Hartebrodt, Reza Nasirigerdeh, David Blumenthal, and Richard Rottger. 2021. Federated Principal Component Analysis for Genome-Wide Association Studies. *IEEE ICDM 2021*, 1090–1095.
[11] Peter Kairouz *et al.* 2021. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning* 14, 1-2 (2021), 1–210. https://doi.org/10.1561/2200000083 arXiv:1912.04977
[12] WHO life expectancy data set. 2022. https://www.kaggle.com/kumarajarshi/life-expectancy-who. Accessed: 2022-01-31.
[13] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. 54 (2017), 10.
[14] Reza Nasirigerdeh, Reihaneh Torkzadehmahani, Julian Matschinske, Tobias Frisch, Markus List, Julian Späth, Stefan Weiss, Uwe Völker, Esa Pitkänen, Dominik Heider, Nina Kerstin Wenke, Georgios Kaissis, Daniel Rueckert, Tim Kacprowski, and Jan Baumbach. 2022. sPLINK: a hybrid federated tool as a robust alternative to meta-analysis in genome-wide association studies. *Genome Biol* 23 (2022).
[15] R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
[16] Ahmed H. Sameh. 1971. On Jacobi and Jacobi-Like Algorithms for a Parallel Computer. *Math. Comp.* 25, 115 (1971), 579.
[17] Ondrej Sluciak, Hana Straková, Markus Rupp, and Wilfried Gansterer. 2016. Distributed Gram-Schmidt orthogonalization with simultaneous elements refinement. *Eurasip Journal on Advances in Signal Processing* 2016, 1 (2016), 1–13.

[18] Ondrej Sluciak, Hana Strakova, Markus Rupp, and Wilfried N. Gansterer. 2012. Distributed Gram-Schmidt orthogonalization based on dynamic consensus. *ACSSC* (2012), 1207–1211.
[19] Peter Snyder. 2014. Yao ' s Garbled Circuits : Recent Directions and Implementations.
[20] Hana Straková, Wilfried N. Gansterer, and Thomas Zemen. 2012. Distributed QR Factorization Based on Randomized Algorithms. In *Parallel Processing and Applied Mathematics*, Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Waśniewski (Eds.). Springer Berlin Heidelberg, 235–244.
[21] Joo Hun Yoo, Hyejun Jeong, Jaehyeok Lee, and Tai-Myoung Chung. 2021. Federated Learning: Issues in Medical Application. (2021), 3–22. arXiv:2109.00202

Chapter 6

Federated K-Means – evaluation of initialization, clustering strategies, and k-selection

6.1 Summary

Federated Learning is a novel data analysis paradigm, dealing with data that is physically distributed over several data holders. Instead of computing a global model at a central server, each participant sends only parameters to an aggregator which computes the global models. A number of K-Means algorithms have been suggested for federated clustering. They generally perform well if k is known. However, there is a lack of algorithms which automatically determine k. This is especially important because with the data distributed over several sites, k cannot be easily determined using for example visual inspection. Another problem in federated machine learning is the fact, that the data is not necessarily independently and identically distributed (iid). In this article, we identify existing algorithms for federated K-Means, evaluate them with respect to their suitability for non-iid data. Lastly, we suggest a strategy to infer k from the data in a federated fashion. Instead of performing many untargeted clusterings in parallel, we suggest to identify locally optimal clusterings and send the centroids, as well as summary statistics for each cluster to the aggregator which determines a good global k and appropriate centroids by sampling artificial data. Once the candidate centroids are found, the final clustering is performed using the real data.

6.2 Introduction

Federated learning (FL) has recently gained traction in the machine learning (ML) community. ML requires large amounts of data in order to train sufficiently generalizable models. However, in the medical setting, data is distributed over several data holders, such as doctors and hospitals. Those parties are usually not willing or allowed to share the patient-level data due to privacy regulations such as the GDPR. FL has been proposed as a method to overcome the challenges of limited data availability in the medical sector, while adhering to legal standards and protecting the privacy of the patients. The idea of FL is to compute local models which can be safely disclosed to a central aggregator as they do not contain patient level information (Matschinske et al. 2021; Yoo et al. 2021). The central aggregator computes a global model from the local model it obtains from the parties, and redistributes it to the participants.

Different versions of federated learning have been proposed in the literature. Cross-silo federated learning assumes the presence of relatively few data silos, for instance hospitals or banks, which contain larger data sets and have access to powerful computing infrastructure. The orthogonal case is called cross-device federated learning where many small (edge) devices with limited compute power contribute data belonging to one individual (Kairouz et al. 2021). In this article, we assume the setting of cross-silo FL.

As federated learning is a relatively new concept, many traditional algorithms in ML need to be adapted to suit the federated setting (Kairouz et al. 2021). In this article, we are investigating federated K-Means clustering. Several federated K-Means versions have been suggested recently. Many of these articles provide good adaptations of centralized K-Means to federated learning. However, many neglect a crucial step in K-Means: the initialization and selection of an appropriate k. When applied to real world data, K-Means is usually run multiple times using randomly selected initial centroids to avoid a solution that corresponds to a local minimum (Fränti and Sieranoja 2019). To promote the application of federated K-Means in practice, a development of a suitable initialization strategy and methods for choosing an appropriate k are required.

Another challenge in federated learning is the data distribution over the participating sites. Generally, it cannot be assumed that the data is drawn from the same underlying distribution (Kairouz et al. 2021). A particular problem for clustering is quantity skew, where different number of points are assigned to the same cluster at different data sites. While some articles

evaluate this to some extend, not all do it in an exhaustive manner. This will be elaborated in section 6.3.5. For the adoption of federated learning in practice it is necessary to know the expected performance of the tools even with unfavorably distributed data. Therefore, in this article, we devise a scheme to thoroughly investigate the performance of clustering algorithms when used on non-iid data.

A bottleneck in federated learning is the transmission of parameters between the computing parties, as every communication step creates a run time overhead (Kairouz et al. 2021). In many federated K-Means adaptations the amount of transmitted data essentially amounts to the cluster centroids, so the volume is small. This would allow K-Means to be run with multiple initializations for multiple k in parallel, to find a consensus solution and the best k. However, the random initialization poses a problem as random centroids can be initialized in sparse regions of the data space. In that case, only few points would be assigned to certain centroids, which could leak information. Therefore, it is beneficial to let the clients initialize the clustering locally, so that they can suggest initial centroids from data dense regions. This also avoids running K-Means with a potentially poor clustering result.

The goal of the clustering is to gain a general understanding of the data without disclosing too much information about the individual. Therefore, we deem allowing the other participants to know, where dense regions are, a reasonable privacy trade-off in collaborative clustering. In a scenario, where patients are clustered into subgroups for diagnostic application, a doctor would like to know what groups there are, i. e. all possible centroids. The doctor might also want to know, how likely the diagnosis is, i. e. how many patients belong to the cluster, and how clear the cluster is separated from other clusters. This requires a certain amount of information to be disclosed to other participants. However, contrary to centralized clustering in the federated clustering scheme suggested in this article, the individual data points are not disclosed, only the global characteristics.

6.2.1 Summary of the contributions

- We identify and evaluate different approaches to compute a federated K-Means clustering of the data.
- We suggest an evaluation strategy for federated clustering in general, with the aim of promoting a more thorough evaluation of clustering

Table 6.1 – Notation table.

Syntax	Semantics
$[N] \subset \mathbb{N}$	index set $[N] = \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$
$S \in \mathbb{N}$	number of sites
$m \in \mathbb{N}$	number of features (i. e. SNPs)
$n \in \mathbb{N}$	total number of samples
$n^s \in \mathbb{N}$	number of samples at site $s \in [S]$
$k \in \mathbb{N}$	number of eigenvectors
$\mathbf{A} \in \mathbb{R}^{n \times m}$	complete data matrix
$\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$	subset of data available at site $s \in [S]$
c	the set of centroids
c_i	the set of centroids at iteration i
`iid`	independently and identically distributed

algorithms using non-iid data.

- We suggest a method to automatically infer k from the data based on the gap statistic and random sampling and evaluate our method using the suggested evaluation procedure.

The remainder of this article is organized as follows. In section 6.3 we will introduce the problem formally. Section 6.4 will cover related work. Section 6.5 details the suggested scheme for the thorough empirical evaluation of the algorithms. In section 6.6 we will provide a description of the algorithms enabling practical application of K-Means in the federated setting. Section 6.7 shows the empirical benchmark both of the previously suggested algorithms as well as the new scheme for the determination of a suitable k.

6.3 Preliminaries

6.3.1 Data distribution model and federated architecture

In federated learning, the data can be distributed over the different parties in different ways. Horizontal data partitioning refers to the case where all clients have a common set of attributes, but distinct set of samples. Vertical data partitioning is the complementary case, where all parties have the same set of samples, but different attributes. In the remainder of this article, we will treat the case of horizontal data partitioning. Let the data $A \in \mathbb{R}^{n \times m}$

be distributed over S sites with $\mathbf{A}^s \in \mathbb{R}^{n^s \times m}$ the local data matrices at sites $s \in [S]$.

In this article, we focus on systems suitable for cross-silo federated learning. We describe our algorithms in terms of a star-like architecture with a central aggregator. Peer-to-peer (P2P) architectures do not rely on a central parameter server, only on communication between the clients. The aggregation operations used in K-Means rely on simple aggregations, such as summing and averaging of small parameters. They can be computed in any architecture, at the expense of higher communication overhead. For secure multiparty aggregation, peer-to-peer communication is necessary, to mask the individual contributions (Matschinske et al. 2021).

6.3.2 Security model & Secure addition

We assume honest-but-curious participants who try to infer as much information as they can from the shared parameters, but do not deviate from the protocol, such as arbitrarily modifying parameters. Under this security model, a simple and efficient scheme has been suggested to securely add values from different parties. In a system with N participants, the parties shard their values into N random values which add up to their secret s_n mod p with p a large prime. Each participant sends each of the $N-1$ shares to the other $N-1$ parties. Every partner computes the sum of the random shards, and communicates those to all participants (or only the aggregator, to save bandwidth) so that the global sum of all secrets can be computed (Cramer et al. 2015).

6.3.3 Centralised K-Means

K-Means is a popular algorithm in unsupervised data analysis, therefore it has been extensively studied. It has initially been suggested by MacQueen (1967) and Lloyd (1982). The goal of K-Means is to minimize the objective function, ususally the mean squared error (MSE) of the points and their respective centroids. Very briefly, the algorithm proceeds as follows. (1) A set $\mathcal{C}_0$ of k initial centroids is chosen. (2) Then, every data point is attributed to its nearest centroid. (3) Finally, the centroids are updated by computing the average over all the points belonging to one centroid. Steps (2) and (3) are repeated until the centroids converge to a stable solution. Several versions of this algorithm have been proposed which differ in implementation details, such as when the centroids are updated.

6.3.4 Challenges in centralised K-Means, local mitigation, and their translation into the federated case

Selection of k

K-Means is a popular algorithm due too its conceptual simplicity and interpretability. There are several challenges in the application of K-Means to real world data. Firstly, the choice of a suitable k, which corresponds to the number of expected clusters. k is usually not know **a-priori**. Several methods have been proposed to find an optimal k, such as the elbow method (Hastie 2017) and the gap statistic (Tibshirani et al. 2001). In K-Means clustering, an error of 0 can be achieved when choosing k as the number of points. Naturally, this is not the k, the user will be interested in. In practice, the user wants to choose k such that the error becomes small, without unduly partitioning natural clusters. The elbow method plots k against the clustering error. The 'elbow' in the curve, meaning the point where the curve becomes flat, signifies the point where the gain due to the increase of k becomes small. A more theoretically founded method is the **gap statistic** introduced by Tibshirani et al. (2001). For the method, data is sampled uniformly from a space approximately spanning the data, meaning random data without cluster structure is generated. This data can be considered the null model. Then, with varying k the artificial random data and the real data are clustered and the errors of the clusterings are compared. The k maximizing the difference in error between the data under the null model and the real data is chosen.

Choice of initial centroids

Secondly, the choice of initial centroids plays a crucial role in the performance of the algorithm. With unsuitable initialization, K-Means is know converge to a local minimum and produce bad clusterings. Fränti and Sieranoja (2019) evaluate the initialization of K-Means extensively. They identify several groups of initialization strategies: random initialization, furthest-first initialization (with K-Means++ a special case of it), sorting-based, projection based, and density based initialization, as well as iterative splitting of the ensemble data. They conclude that taking the consensus of several executions of K-Means with randomized furthest-first initialization performs well in practice. Furthest-first initialization iteratively chooses the point to be included in the set of initial centroids which has the largest distance to the closest centroid among the already chosen ones. In practice,

K-Means is often executed multiple times in order to avoid returning a local optimum as the solution. This could be parallelized easily by computing multiple clusterings at once, such that the number of iterations is equal to to worst run.

High dimensional data

Another problem of distance based methods in general is, that the distance metrics tend to "degenerate" in high dimensions, meaning all data points appear to be at a similar distance from each other. A popular strategy to mitigate this problem is the application of Principal Component Analysis (PCA) prior to the application of K-Means and to cluster the projections of the data into a reduced eigenspace instead of the high dimensional original data. Such an approach has been suggested by M. F. Balcan et al. (2013). Furthermore PCA is a useful tool for the detection of an appropriate k and can be used in an advanced version of the gap statistic. Algorithms for the efficient retrieval of the federated principal subspace have been proposed previously (M.-F. Balcan et al. 2014; Hartebrodt et al. 2021).

6.3.5 Evaluation of K-Means in the literature

Many authors include evaluation of their algorithm on non-iid data, as many of the algorithms are designed with non-iid data in mind. However, often the non-iid scenarios evaluated are ad-hoc data distributions, which focus on a particular type of data distribution. Brandão et al. (2021) use data sets available online and distribute them based on arbitrarily set values. Naldi and Campello (2014) evaluate their approach on online available data by distributing the clusters such that a randomly selected cluster $\mathcal{C}_i$ contributes a high fraction of points to a site j and few points to all other sites. Depending on the number of sites, the remaining sites might receive a still significant fraction of points of a cluster, or "skewed noise". Dennis and Smith (2019) evaluate their algorithm on iid data as well as structured partitions where each site receives data of only a subset of clusters. Liu et al. (2020) and Rao et al. (2016) evaluate on iid data. Datta et al. (2009) use a Zipf distribution over the network. A few works focusing on privacy rather than proposing a new algorithm omit practical evaluation in terms of accuracy or practical run time.

6.4 Related work

6.4.1 Federated k-means

Federated K-means has the same goal as centralized k-means, to compute a partitioning of the data, with the difference that the data is available at multiple sites which cannot transmit their shares to a central server. Despite the relative novelty of federated learning, a number of k-means algorithms approaching the problem in different ways have been proposed. Here, we sub divide into "one shot" approaches which require a single communication round between the clients and the aggregating server; and iterative approaches which allow more communication rounds.

Exact k-means

K-means can be computed exactly with respect to the centralized solution based on summary statistics shared between the central server and the clients (Brandão et al. 2021; Mohassel 2020; Ghosal and Chatterjee 2018). At each iteration there is a local phase where the clients compute the local sum of distances from each point to the nearest global centroid broadcast by the aggregator $\mathbf{c}_s^s = (\sum_p dist(p, c_i) | c_i \in \mathcal{C}_i)$. The $\mathbf{c}_s^s$ is broadcast to the aggregator together with the vector of number of elements belonging to the sums $\mathbf{c}_s^n$. At the aggregator the global centroids are computed as $\mathbf{c}_i = \sum_{s \in S} \mathbf{c}_s^s / \sum_{s \in S} \mathbf{c}_s^n$. These global centroids are then broadcast to the clients again, where they can be updated. The process is then repeated until convergence. This can for example be defined as the stability of the centroids with regards to the previous iteration.

The set of initial centroids can be determined using several methods proposed in the literature. Mohassel (2020) suggests a local round of k-means at each client to create the initial centroids. The initialization proposed by Brandão et al. (2021) leads to a collapse in k in practice. The idea is to generated random points close to the center of the data. Each client then computes local centroids and uses the fake initial centroids to align their centroids, which fails if two initial centroids are close to the same fake cluster and are assembled to one. Ghosal and Chatterjee (2018) use random initialization which has been shown to perform badly in practice (Fränti and Sieranoja 2019)

Table 6.2 – Federated k-means algorithms

Type	initialization	Global centroid	Article
Single pass	Local k-means	Furthest first	Dennis and Smith 2019
	Local k-means	K-means	This article
Fully federated	Random		Ghosal and Chatterjee 2018 Zhang et al. 2022
	Local k-means		Brandão et al. 2021 Mohassel 2020
Multipass	Random		Naldi and Campello 2014

One shot approaches

One shot approaches may not reach the exact same clustering as federated exact k-means, because they follow a different protocol. There is a local phase at the clients, where an optimal clustering of the data is computed, and the centroids are shared with the aggregator. The aggregator computes global centroids given the local centroids and broadcasts the result to the clients. Dennis and Smith (2019) propose the following approach to aggregate the centroids: The aggregator selects a random site as the leading party and includes all its centroids in the set of global centroids. Then, from all remaining candidates, the furthest centroid to the existing set are added until there are k centroids in the set of global centroids. According to the authors, this algorithm is able to deal with heterogeneous data, which in this context means that clusters do not necessarily need to be available at all sites. After the initial phase the clustering is computed immediately, by assigning the local points to the global centroids

6.4.2 Other methods

Khedr and Bhatnagar (2014) devise a way to deal with arbitrary data partitioning. Their main contribution is the generation of a joint feature space. As this is not our primary concern here, we omit this algorithm from the benchmark.

Naldi and Campello (2014) propose an evolutionary k-means algorithm, based on a previous non-federated evolutionary k-means algorithm. As this version is so strongly modified from the base k-means version and uses the silhouette score instead of the MSE as a secondary evaluation criterion, we exclude it from the comparison.

Many methods have been suggested for peer-to-peer (P2P) networks operating under a fundamentally different computation and communication model (Datta et al. 2009; Liu et al. 2020; Soliman et al. 2020; Qin et al. 2017). For instance the number of devices is assumed to be high, but each device has little computation power. Furthermore, devices are assumed to communicate with a few of their immediate neighbors (Datta et al. 2009) and therefore the aggregation process differs. As a consequence P2P algorithms are highly specialized to the architecture and not immediately applicable to cross-silo federated learning.

6.4.3 Private K-means clustering

The federated learning paradigm means that the data cannot leave the owner's data center. However this does not necessarily prevent attackers from inferring knowledge about the data from aggregated statistics. Therefore, several privacy preserving k-means algorithms, both federated and centralised have been proposed. Among the federated versions there are several relying on secure multiparty computation (Omri et al. 2019; Ramírez and Auñón 2020; Jagannathan and Wright 2005; Rao et al. 2016; Mohassel 2020). Jagannathan and Wright (2005)'s protocol is designed for two parties. Ramírez and Auñón (2020) and Rao et al. (2016) have a relatively high run time, while Omri et al. (2019) do not provide an empirical evaluation. Mohassel (2020) require 18 minutes using two compute clusters to cluster 100,000 points with $k = 2$. Kaplan and Stemmer (2018), M. F. Balcan et al. (2013), and Feldman et al. (2017) suggest the use of differential privacy by computing a private core set. The common denominator of the SMPC based approaches is to replace the insecure mean computation in of the 'exact' federated k-means algorithms by secure aggregation. Feldman et al. (2017) develop a clustering scheme with reasonable privacy guarantees, but the scheme is limited to small k and high numbers of samples. Hou et al. (2021) use secure outsourcing of the data to compute K-Means on the server side. The centers are initialized randomly. Their article also contains an overview of related methods.

6.5 Systematization of evaluation of federated clustering

Many articles perform an empirical evaluation, and some do so on non-iid data, but not all take multiple types of non-iidness into account. Therefore, we attempt to devise a systematic way of evaluating the clustering

algorithms. In the following section, we describe several scenarios for the distribution of data over multiple sites. We will use the term 'cluster' to refer to all data points labeled as belonging to the same cluster. We devise two major generic sources of bias, firstly, unequal distribution of the clusters, i.e. all samples belonging to a cluster, over the sites, secondly, unequal distribution of the number of samples per cluster over the sites, i.e. some sites having a larger proportion of samples of one cluster than others. The first one (cluster centric bias) can be seen as an extreme version of the latter (sample centric bias), but the full spectrum of scenarios is impossible to investigate. Therefore, we make this attempt at disentangling the two.

As we chose K-Means as a demonstration case, here we use terminology used to describe K-Means for the number of clusters. Naturally, we suggest to use this clustering evaluation for different clustering algorithms as well.

The data $\mathbf{D}$ with n samples and m features is distributed among S sites. All data points have the same m features. The global data ('oracle data') can be divided into k_{global} clusters. We assume to have an oracle clustering, meaning we know the number of global clusters, as well the labels, and for each site we know the number of local clusters including their labels. This oracle may be the clustering induced by a centralised version of K-Means or an independent gold standard. In a real test case, the number of k_{global} and k_{local} is naturally not known and should be determined independently.

To illustrate this we imagine fictional data with 10 clusters of 100 points each to be distributed over 5 sites. Also refer to figure 6.1 to gain a visual intuition of the data partitioning.

`iid` **data distribution** : All s sites receive an equally sized `iid` sample of data points belonging to all k_{global} clusters. I.e. our fictional data would be distributed such that all 5 sites receive 20 points of each fo the 10 clusters, sampled `iid` with respect to the cluster.

cluster-centric non `iid` **data distribution** : A number of clusters $k_{iid} < k_{global}$ is distributed equally over all sites. The remainder of the data is distributed over $S' < S$ sites, such that $S - S'$ sites miss some clusters. For each cluster, the points are distributed equally over all sites having the respective cluster. I.e. In our example $S = 5$ and let S' be 3. Let further $k_{iid} = 5$. These 5 clusters would be distributed iid over all $S = 5$ sites. Every one of the 5 remaining clusters would be distributed equally over $S - S' = 2$ randomly selected sites. (The non iid clusters do not all go to

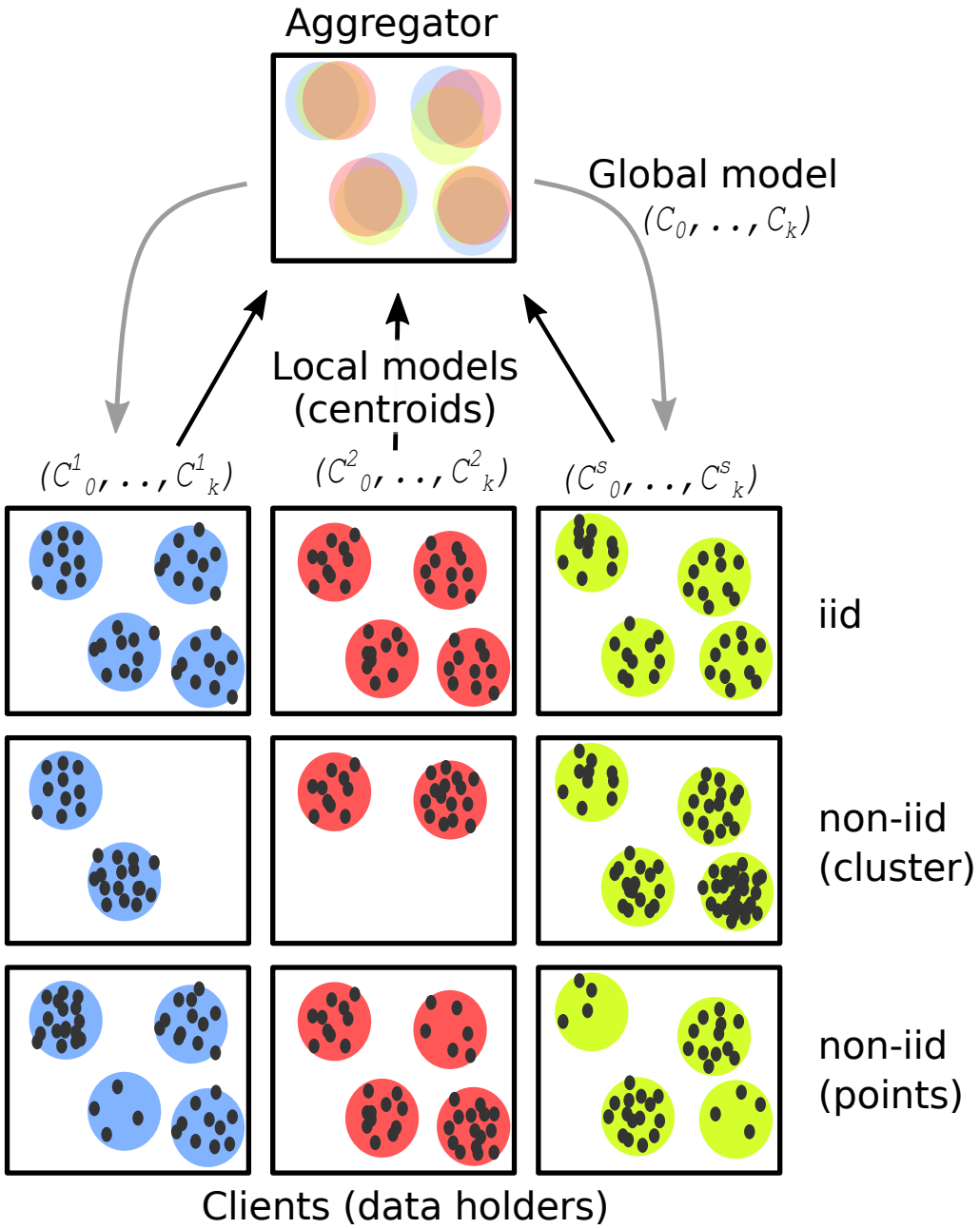


Figure 6.1 – Overview over the possible data partionings in federated clustering. Data can be distributed `iid` with respect to the labels, non-iid with repect to the clusters and non-iid with respect to the number of points.

the same two sites). One would say "5 clusters are not available at 3 sites". The corner cases might be noteworthy (see also figure for illustration):

- If $k_{iid} == 0$ and $S' = S - 1$ every cluster will only be available at a single site, the clusters will be completely disjoint.
- if $k_{iid} == k'$ for k' a small integer and S' is small, this will result in some sites having extra clusters.
- if $k_{iid} == k'$ for k' a large integer and S' is large, some sites will have missing clusters.

point-centric non-iid : All sites receive a random unequal share of points for each cluster, but the fraction is non-zero. This version can be seen as a continuous version of the previous scenario, where for each cluster a random probability vector $p \in \mathbf{R} s.t. \sum_p = 1$ is generated. For the discretized example described in the previous paragraph with $S' = 3$ such a random share could for instance be $[0, 0.5, 0, 0.5, 0]$. In the current case the probabilities are non-zero for all participants and clusters. The points of the cluster are then distributed according to the generated probabilities. The full space of all possible probability vectors cannot, and does not need be, explored as there are too many possibilities. Therefore, we suggest to use 'scenarios' to reduce the search space to a manageable size and maintain interpretability, which would be lost using completely arbitrary data splits. Here, we proposed two example scenarios, but suggest the use of additional ones if applicable.

- 'Stairs' (Algorithm 1): The distributions are generated based on a base probability p_b and the remaining probability p_r which is initially set to 1 and which is updated as $p_r = p_b * p_r$. So, the first site receives p_b, every subsequent site a fraction in function of this initial probability. This leads to a 'stair' like pattern, which might need to be randomized, unless one site should systematically receive more points of every cluster. Depending on the number of sites, the base probability should be set such that there are not too few points in the smallest cluster.
- 'High-low' (Algorithm 2): In this scenario, for every cluster, there are only two modes, 'large' and 'small'. Large sites receive p_b/s_l percent of the point of a cluster where s_l is the number of large sites and p_b a base probability. Small sites receive $(1 - p_b)/(1 - s_l)$ percent of the

points for the respective cluster. Again, b_p should be chosen carefully to avoid overly small sites, unless this scenario is intended.

- Other specific scenarios, such as points, that appear as outlier at one location, but are part of a larger cluster at another location could be modeled seamlessly using this approach.

To summarize, we suggest the following partitions of the data over the different sites:

- `iid` data distribution with respect to the clusters and the number of points per clusters.
- non-`iid` data w. r. t. to the number of points per cluster which can be as serious as not having any points per site.

Algorithm 1: Stairs

Input: Base probability p_b, number of sites S
Output: Probability vector p_d

$p_c = p_b$
$p_r = 1$
for *s in 1..S-1* **do**
    $p_d[s] = p_b * p_r$
    $p_r = 1 - p_d[s]$
$p_d[s] = p_r$
return p_d

Algorithm 2: High-low

Input: Base probability p_b, number of sites S, number of large large sites S_l
Output: Probability vector p_d

$p_d[1..S_l] = p_b/S_l$
$p_d[S_l + 1..S] = (1 - p_b)/(S - S_l)$
return p_d

6.5.1 Other possible sources of error

Federated learning is likely to suffer from other batch effects which are difficult to predict and may lead to complete failure. While label shift will

not be a problem, due to the unsupervised nature of the algorithms, a shift or rotation of the data within the coordinate system, need to be addressed prior to the application of federated clustering. In practice, these problems will be difficult to detect and likely require more traditional data mining techniques such as visual data inspection.

6.6 Practical federated clustering using federated k-means

The second goal of this work is to suggest a practical k-means clustering scheme, which is able to determine a good choice of k, arrives at at good clustering score and is communication efficient. Based on prior work, we identify components which can be used in federated k-means. These methods assume k to be known. We assemble these components into several meta k-means versions to evaluate their performance. Based on the combination(s) which achieve good performance, we suggest a strategy which also allows to infer k during the execution of the algorithm

6.6.1 Algorithm components

We investigate the following components and a few of their combinations in the search for a practical k-means strategy for federated learning. We also introduce a few additional ideas from the centralized K-Means literature in an attempt to provide a more comprehensive evaluation. We introduce the notation and general ideas of the components and then proceed to explain the new strategies in more detail below.

- Initialization strategies:
  - LCLU Perform a locally optimal k-means clustering and broadcast the centroids (Dennis and Smith 2019; Mohassel 2020). In addition to the centroids, the empirical covariance matrix of each cluster can be computed and broadcast. The clustering stays the same but more parameters are transmitted.
  - FFI - Furthest first initialization: adaptation of the local furthest first procedure. One of the sites selects an initial centroid, all subsequent sites select a centroid furthest from the current set of candidate centroids. In order to not disclose data points, a local average over several data points is formed.

- Aggregation strategies:
  - FFA - Furthest first aggregation: Procedure to compute a set of global centroids from a multiset of local centroids. Similar to furthest first initialization, an initial centroid is chosen from all available local centroids and further candidates are added to the set until the desired k is reached (Dennis and Smith 2019).
  - CCLU - Clustering of the local centroids at the aggregator to form global centroids.
  - SAMCLU - Generation of artificial data based on the centroids and the covariance information, followed by the clustering of the artificial data with the optimal k inferred using the gap statistics based on the artificial data (See section 6.6.2 for further explanation). The centroids computed based on the artificial data are transmitted to the clients, which either proceed with further iterations or label the data directly.
- Cluster refinement strategies:
  - LU - Local refinement of the global centroids which amounts to a single round of k-means at the local site using the global clusters as initial centroids, and the computation of new centroids locally. These are then sent to the aggregator which clusters the centroids and returns the aggregated centroids back to the clients.
  - FKA - Fully federated k-means, a direct adaptation of centralized k-means. A global set of centroids is broadcast, then the clients compute the sum of the distances of the points belonging to each centroid, and the aggregator computes the new global centroids, based on the sums and the corresponding numbers of points (Mohassel 2020).

6.6.2 Determination of k

In this section, we describe an extension of previous approaches to determine a good choice of k simultaneously with a good clustering. The gap statistics works well in the centralized case, but faces an important issue in federated k-means. As the data cannot be moved to the central server, even with a global null model, the local optimal clusterings may lead to different optimal k at the sites. Averaging, or maximizing the number of clusters are unwise choices, because different sites can have identical numbers of clusters,

but they might be subsets of a larger set of clusters. It would possible to compute the global gap statistics by running k-means in a parallel manner, so choosing a range of k, computing the clustering, and selecting the best k after the computation. This however, creates communication overhead, and potentially allows to triangulate data points, because many summary statistics are available. Therefore, we test two strategies to compute a good number of k.

First the locally optimal clustering is determined using the gap statistics. This can be done without any communication overhead. After determination of a good local k, the locally optimal cluster centroids are sent to the aggregator. At the aggregator, the local centroids are clustered for a range of k and the clustering and the k with the best score is selected and the corresponding cluster centers are returned to the clients. We allow clusters of centroids containing only one centroid to account for the possibility of client-specific clusters. This approach is called CCLU.

The number of points available in the centroid clustering is potentially small, with few participants. Therefore, we suggest a second strategy to determine a good k which discloses more information on the local clusterings. In addition to the centroids, the covariance matrix of the points belonging to the centroid is sent to the aggregator. The aggregator creates artificial data from this information by sampling from a multivariate Gaussian distribution. This allows to account for the number of cluster members by choosing the number of samples appropriately. This approach is called SAMCLU.

There are two possible strategies to select the optimal k according to the gap statistic. The gap statistic quantifies how much better the clustering becomes by adding an additional k. For a range of k there are $k-1$ values. The original version by Tibshirani et al. uses the k corresponding to the first maximum. We denote this selection strategy T. An alternative is to choose the k corresponding to the global maximum (GM). As a third strategy, we investigate the silhouette score as a measure to determine the locally optimal clustering and the global k (S). The silhouette score estimates how well the clusters are separated.

6.7 Practical evaluation of federated K-Means clustering

6.7.1 Synthetic data and data partitioning

We generate Gaussian clusters using different parameter combinations, and partition the generated data using the setups describes earlier. As we focus on K-Means, an algorithm which has been shown to under perform on other data, here we focus on Gaussian data, but the distribution used for data generation can naturally be replaced by more task appropriate generators. Table 6.2 summarizes the parameters used. First, we generate the cluster centers. For every set of cluster centers, we generate data sets with different variances, such that we have data sets with matching centroids, but varying variances. Each cluster consists of 500 points. We furthermore vary the number of features, and create datasets with 2, 5, and 10 features. For each of the data sets we create versions with and without outliers. Outliers are data points that have a higher variance, for example by a factor of 2, than data points belonging to the cluster.

Once, we have generated the data, based on the considerations in section 6.5, we generate a cluster-centric non-iid distribution with an increasing number of clusters which are only available at very few sites. Tests with this data partitioning strategy are called "grid". We then use the two point centric generators to generate 'softer' non-iid distributions where the clusters are available at all sites but with an increasingly skewed distribution (stairs, and high-low). As a baseline we generate an iid-distributed dataset which assigns an equal number of points to every cluster across all sites.

6.7.2 Test setup

We run two sets of tests. One evaluating the different K-Means algorithms from the literature, with the aim to find the most accurate and most communication efficient algorithm when k is known. See table 6.4 for a summary of all the test configurations. Based on the initial tests, we run a second set of tests where k is unknown and needs to be inferred during the federated clustering process using the clustering strategies that performed best in the initial test in terms of accuracy and communication efficiency together with the federated gap statistic. Note that the one-shot approaches come "for free" and thereby are evaluated implicitly.

Table 6.3 – List of parameters for data generation and partitioning

	Name	Values
Data generation	Number of clusters	5
	Number of Points	500
	Variance	0.5 1.5 2.5
	Number of Features	2 5 10
	Percentage of outliers	0% 1%
Data partitioning	Scheme	iid disjoint stairs high-low
	Sites	5

6.8 Test metrics

We use the the average F1-Score to determine the overall quality of the clustering. It is calculated as the harmonic average over precision and recall. TN denotes the number of true negative predictions; TP the number of true positives; FN the number of false negatives and FP the number of false positive predictions. The F1 score is computed as follows.

$$P = \frac{TP}{TP + FP} \tag{6.1}$$

$$R = \frac{TP}{TP + FN} \tag{6.2}$$

$$F1 = \sum \frac{2 \cdot P \cdot R}{P + R} * \frac{1}{\#Samples} \tag{6.3}$$

For the generated data, we have the true labels. The test metric we report is the F1 Score obtained from the federated clustering normalized by the

Table 6.4 – Algorithm configurations

Abbreviation	Description
LCLU - CCLU	One shot approximate clustering
LCLU - CCLU - LU	Clustered initialisation with iterative refinement
LCLU - FFA	One shot approximate clustering with furthest first aggeregation
CLUL - FFA - LU	Clustered initialisation with iterative refinement.
FFA - FKA	Furthest first initialisation with fully federated K-Means
CLU - FKA	Clustered initialisation with fully federated K-Means.
LCLU - CCLU - (T/GM/S)	One shot approximate clustering with automatic k selection.
LCLU - CCLU - LU - (T/GM/S)	Clustered initialisation with iterative refinement with automatic k selection.
CLU - FKA - (T/GM/S)	Clustered initialisation with fully federated K-Means with automatic k selection.

F1 score obtained from the centralized clustering evaluated based on the gold standard solution. Therefore, the reported scores are a measure for how faithful the federated clustering is to the centralized clustering. We also report the number of iterations until convergence of the results.

6.8.1 Test results

Figure 6.2 shows the evaluation results for test run with the grid evaluation strategy regarding the number of iterations. The figure is structured as follows: There are three rows of panels, which show the results for the different number of features. Each row is divided into three colums for the different variances. Within each panel, the three columns show the results for three evaluation strategies ('stairs', high-low' and 'grid'). The rows in each panel contain the values for the different algorithm versions described earlier. The maximal number of iterations allowed was 100. The data did not contain outliers.

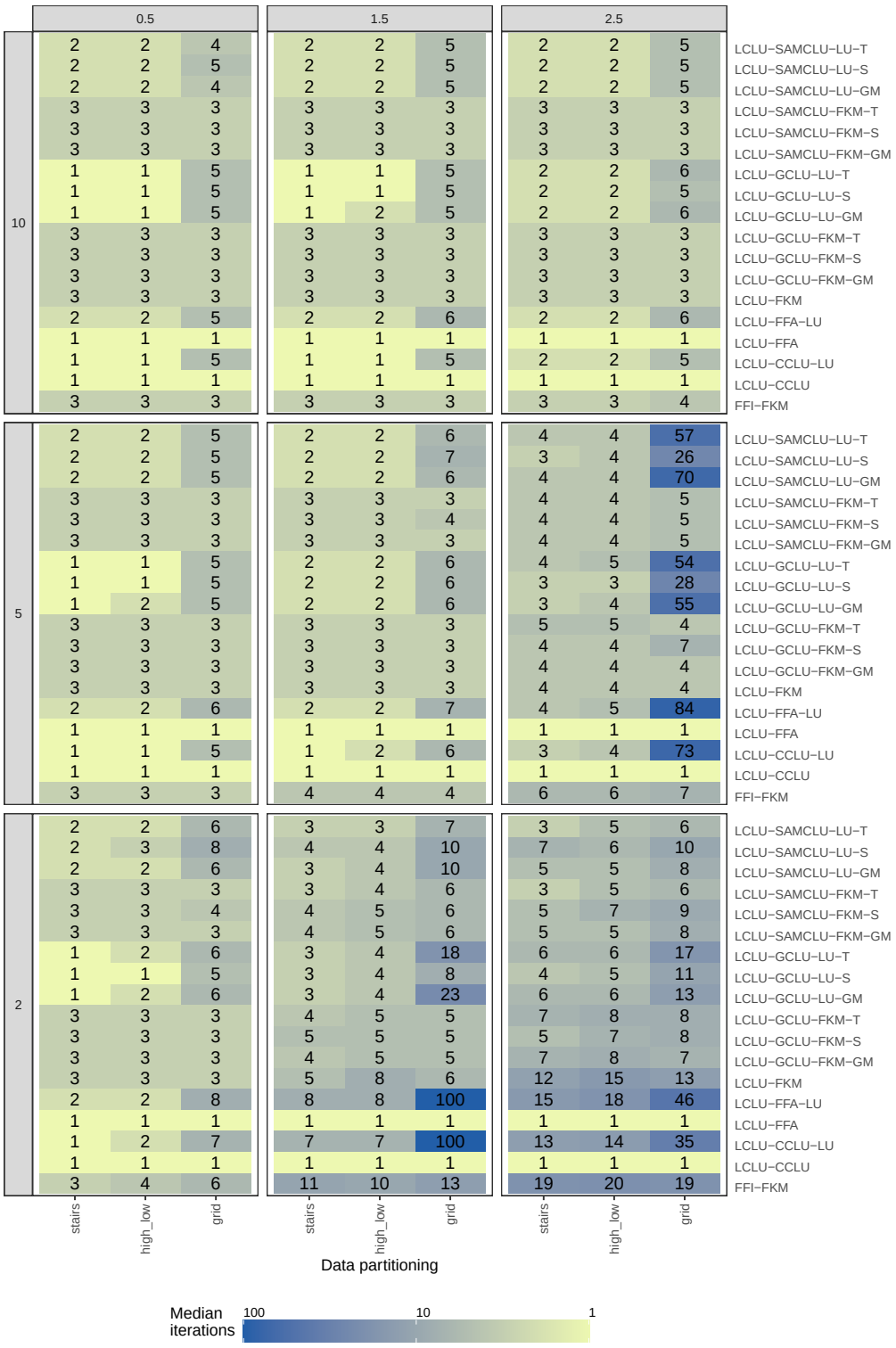


Figure 6.2 – Number of iterations until convergence for different numbers of features, different variances (panel columns), test configurations and data splits.

We observe that generally, a low number of iterations is sufficient for the algorithms to converge. Data which is distributed using the grid strategy required a slightly higher number, but still generally below 10 rounds. With a low number of features (2, and 5) the grid strategy lead to worse convergence, especially for the 'LU' refinement strategy. This is due to few points at the edge of the cluster constantly switching labels. For the 2 dimensional data set the convergence was worse compared to the higher dimensional ones. We hypothesize, that in the artificially generated data, the clusters are generally well separated, even for higher dimensions, so the algorithm benefits from additional planes separating the clusters. To summarize, most strategies perform adequately according to the number of iterations, however the LU refinement strategy is prone to bad convergence and should be excluded from further investigation. This is further backed by supplemental fig. C.4.

Figure 6.3 (on the next page) summarizes the results for the accuracy evaluation based on the grid strategy for the data sets without outliers. The figure is organized as follows. The panel columns contain experiments for data sets with different variances. We observe, the following trend: with growing cluster variance, the clustering results deviate more strongly from the gold standard clustering. In the bottom row of each cluster we show the expected result from the centralized clustering. We did not expect the federated clustering to perform generally better than the centralized clustering algorithm, therefore these results are not surprising.

Now we describe, how to understand each panel. Each panel row represents the number of clusters missing. The columns within each panel stand for the number of sites where a cluster is missing. For the top left most cell, one would read "Four clusters are missing at 5 sites". This corresponds to a completely disjoint clustering, with 1 cluster per site. Within each panel, the rows show the performance of each algorithm. Globally, the strategies with known k perform better than the adaptive clustering strategies. We have two sets of adaptive clustering strategies: the ones, which clusters the centroids (blue), and the strategies which generate artificial data at the aggregator based on the covariance matrix. Based on the figure and further material included in the supplement (figs. C.1 to C.3), both clustering strategies perform similarly. The overall best selection criterion for the best k is the global maximum selection strategy which consistently delivers relatively good results, although not quite reaching the baseline accuracy.

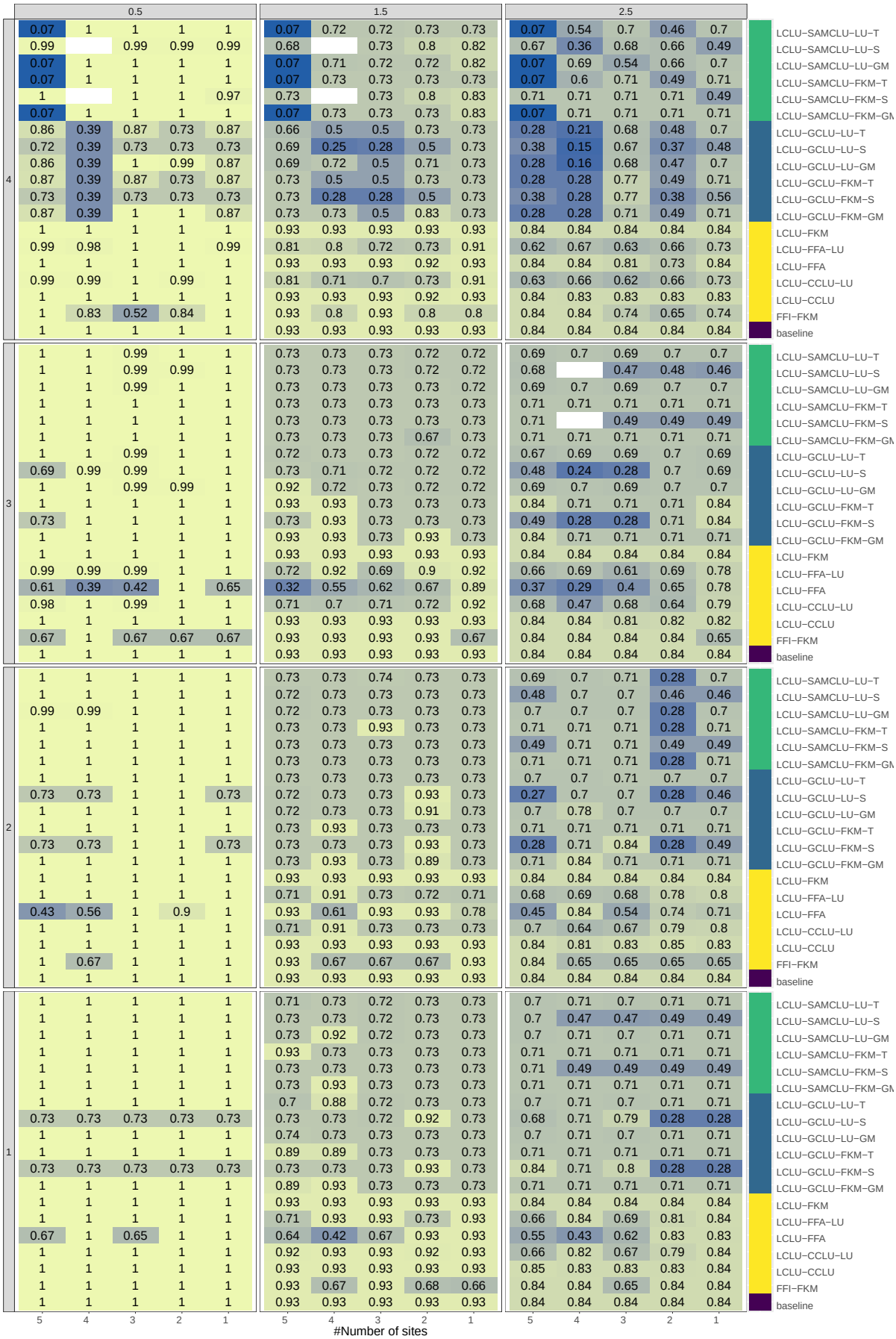

0.5
1.5
2.5
4
3
2
1
LCLU−SAMCLU−LU−T
LCLU−SAMCLU−LU−S
LCLU−SAMCLU−LU−GM
LCLU−SAMCLU−FKM−T
LCLU−SAMCLU−FKM−S
LCLU−SAMCLU−FKM−GM
LCLU−GCLU−LU−T
LCLU−GCLU−LU−S
LCLU−GCLU−LU−GM
LCLU−GCLU−FKM−T
LCLU−GCLU−FKM−S
LCLU−GCLU−FKM−GM
LCLU−FKM
LCLU−FFA−LU
LCLU−FFA
LCLU−CCLU−LU
LCLU−CCLU
FFI−FKM
baseline
5 4 3 2 1 5 4 3 2 1 5 4 3 2 1
#Number of sites

Figure 6.3 – (previous page) Evaluation results using the grid evaluation strategy without outliers.

The FFA aggregagtion strategy did not return good results already on the easy cases with clusters that were well separated and had a low variance. Therefore, this strategy will be removed from further considerations in the future.

A higher number of outliers did lead to a generally worse performance of the algorithms and to the complete failure of some test runs, where the automatic detection of k lead to a high number of very small clusters. On data with completely disjoint clusters, the federated algorithms achieved better performance, because they had to detect only one dense clustering (see fig. C.1).

6.9 Conclusion & Outlook

Our evaluation of the K-Means algorithms and their extension to unknown k closes a practicability gap in federated learning. For general use cases, local clustering followed by global clustering are reasonable choices. Overall the number of iterations (the number of communication rounds) is low. Predictably, bad cluster separation leads to worse clustering results. If k is known, the federated clustering performs well. The algorithms are generally not suited to handle edge cases in data distribution and outliers. This emphasizes the need to evaluate the clustering algorithms to obtain realistic estimates of their performance. Future work will require the evaluation of the algorithms on real data. With unknown k the algorithms achieve worse performance compared to the case where k is known. Based on these results, future improvements of the adaptive federated clustering strategies can be made.

References

Balcan, M. F., Ehrlich, S., and Liang, Y. (2013). "Distributed k-Means and k-Median Clustering on General Topologies". In: pp. 1–9. arXiv: 1306.0604. URL: http://arxiv.org/abs/1306.0604.

Balcan, M.-F. et al. (2014). "Improved Distributed Principal Component Analysis". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, pp. 3113–3121.

Brandão, A., Mendes, R., and Vilela, J. P. (2021). "Efficient Privacy Preserving Distributed K-Means for Non-IID Data". In: pp. 439–451. DOI: 10.1007/978-3-030-74251-5_35.

Cramer, R., Damgård, I. B., and Nielsen, J. B. (2015). *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press. DOI: 10.1017/CBO9781107337756.

Datta, S., Giannella, C. R., and Kargupta, H. (2009). "Approximate distributed K-means clustering over a peer-to-peer network". In: *IEEE Transactions on Knowledge and Data Engineering* 21.10, pp. 1372–1388. ISSN: 10414347. DOI: 10.1109/TKDE.2008.222.

Dennis, D. K. and Smith, V. (2019). "Heterogeneity for the Win: Communication-Efficient Federated Clustering". In: arXiv: arXiv:2103.00697v1.

Feldman, D. et al. (2017). "Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks". In: *Proceedings - 2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2017*, pp. 3–15. DOI: 10.1145/3055031.3055090.

Fränti, P. and Sieranoja, S. (2019). "How much can k-means be improved by using better initialization and repeats?" In: *Pattern Recognition* 93, pp. 95–112. ISSN: 00313203. DOI: 10.1016/j.patcog.2019.04.014.

Ghosal, R. and Chatterjee, S. (2018). *Privacy Preserving Multi-server k-means Computation over Horizontally Partitioned Data*. Vol. 11281 LNCS. Springer International Publishing, pp. 189–208. ISBN: 9783030051709. DOI: 10.1007/978-3-030-05171-6_10. arXiv: 1808.03811. URL: http://dx.doi.org/10.1007/978-3-030-05171-6_10.

Hartebrodt, A. et al. (2021). "Federated Principal Component Analysis for Genome-Wide Association Studies". In: Icdm, pp. 1090–1095. DOI: 10.1109/ICDM51629.2021.00127.

Hastie, T. T. (2017). "The Elements of Statistical Learning Second Edition". In: *Math. Intell.* 27.2, pp. 83–85. ISSN: 03436993. DOI: 111. eprint: arXiv:1011.1669v3.

Hou, R. et al. (2021). "Multi-Party Verifiable Privacy-Preserving Federated k -Means Clustering in Outsourced Environment". In: *Security and Communication Networks* 2021. ISSN: 19390122. DOI: 10.1155/2021/3630312.

Jagannathan, G. and Wright, R. N. (2005). "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 593–599. DOI: 10.1145/1081870.1081942.

Kairouz, P. et al. (2021). "Advances and open problems in federated learning". In: *Foundations and Trends in Machine Learning* 14.1-2, pp. 1–210. ISSN: 19358245. DOI: 10.1561/2200000083. arXiv: 1912.04977.

Kaplan, H. and Stemmer, U. (2018). "Differentially private k-means with constant multiplicative error". In: *Advances in Neural Information Processing Systems*. Vol. 2018-Decem, pp. 5431–5441. arXiv: 1804.08001.

Khedr, A. M. and Bhatnagar, R. K. (2014). "New algorithm for clustering distributed data using K-means". In: *Computing and Informatics* 33.4, pp. 943–964. ISSN: 13359150.

Liu, Y. et al. (2020). "Privacy-preserving fe derate d k -means for proactive caching in next generation cellular networks". In: *Information Sciences* 521, pp. 14–31. DOI: 10.1016/j.ins.2020.02.042. URL: https://doi.org/10.1016/j.ins.2020.02.042.

Lloyd, S. P. (1982). "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. ISSN: 15579654. DOI: 10.1109/TIT.1982.1056489.

MacQueen, J. (1967). "SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS". In: 233.233, pp. 281–297.

Matschinske, J. et al. (2021). "The FeatureCloud AI Store for Federated Learning in Biomedicine and Beyond". In: pp. 1–32.

Mohassel, P. (2020). "Practical Privacy-Preserving K-means Clustering". In: pp. 1–31.

Naldi, M. C. and Campello, R. J. (2014). "Evolutionary k-means for distributed data sets". In: *Neurocomputing* 127, pp. 30–42. ISSN: 09252312. DOI: 10.1016/j.neucom.2013.05.046. URL: http://dx.doi.org/10.1016/j.neucom.2013.05.046.

Omri, O. E. et al. (2019). "Privacy-Preserving k-means Clustering: an Application to Driving Style Recognition". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11928 LNCS, pp. 685–696. ISSN: 16113349. DOI: 10.1007/978-3-030-36938-5_43.

Qin, J. et al. (2017). "Distributed k-Means Algorithm and Fuzzy c-Means Algorithm for Sensor Networks Based on Multiagent Consensus Theory". In: *IEEE Transactions on Cybernetics* 47.3, pp. 772–783. ISSN: 21682267. DOI: 10.1109/TCYB.2016.2526683.

Ramírez, D. H. and Auñón, J. M. (2020). "Privacy Preserving K-Means Clustering: A Secure Multi-Party Computation Approach". In: arXiv: 2009.10453. URL: http://arxiv.org/abs/2009.10453.

Rao, F. Y. et al. (2016). "Privacy-Preserving and Outsourced Multi-user K-Means Clustering". In: *Proceedings - 2015 IEEE Conference on Collaboration and Internet Computing, CIC 2015*, pp. 80–89. DOI: 10.1109/CIC.2015.20.

Soliman, A. et al. (2020). *Decentralized and Adaptive K-Means Clustering for Non-IID Data Using HyperLogLog Counters*. Vol. 12084 LNAI. Springer International Publishing, pp. 343–355. ISBN: 9783030474256. DOI: 10.1007/978-3-030-47426-3_27. URL: http://dx.doi.org/10.1007/978-3-030-47426-3_27.

Tibshirani, R., Walther, G., and Hastie, T. (2001). *Estimating the number of data clusters via the gap statistic*.

Yoo, J. H. et al. (2021). "Federated Learning: Issues in Medical Application". In: pp. 3–22. ISSN: 16113349. DOI: 10.1007/978-3-030-91387-8_1. arXiv: 2109.00202.

Zhang, E. et al. (2022). "Practical multi-party private collaborative k-means clustering". In: *Neurocomputing* 467, pp. 256–265. DOI: 10.1016/j.neucom.2021.09.050. URL: https://doi.org/10.1016/j.neucom.2021.09.050.

Chapter 7

Discussion & Conclusion

The previous chapters were dedicated to the study of specific unsupervised federated machine learning problems. As an emerging research field, many questions in federated learning have no definite answer yet, and not all of them can be addressed at once. The investigated three methods, singular value decomposition, QR factorization and K-means clustering are popular applications in bioinformatics and will be required for the adoption of federated learning in computational biology. The five manuscripts address crucial issues for federated learning: privacy, accuracy, and communication and resource efficiency. This final chapter discusses the manuscripts with respect to these challenges. First, in section 7.1, chapters 2 to 6 will be summarized briefly, highlighting the main contributions. Then the overall conclusions will be drawn in sections 7.2 to 7.4, followed by a brief discussion of future directions in section 7.5. The final section 7.6 will conclude the work.

7.1 Summary

Chapter 1 introduced the necessary terminology and open problems in federated learning in general. Based on these open challenges, the aim of this thesis was the investigation and development of communication and resource efficient, privacy-preserving and accurate algorithms for federated unsupervised learning.

Chapter 2 identified a number of PCA algorithms suitable for horizontally partitioned PCA. They were investigated with respect to their accuracy and communication efficiency. The focus of the manuscript was the accuracy

of the methods when using non-iid data. We found federated approximate PCA to be unsuited to detect batch effects. Otherwise most methods perform adequately. Low communication overhead can be attained at the privacy cost of communicating the entire covariance matrix.

Chapter 3 represents initial work on federated PCA for Genome-Wide Association Studies. In this context, we suggest power iteration. To achieve orthonormality, while keeping the sample associated eigenvectors private, in the last iteration, federated Gram-Schmidt orthonormalization can be used. The results converge to the centralized equivalent after sufficient iterations.

Chapter 4 builds on the work in the third chapter, and extends the presented algorithms, by applying approximate PCA as an initialization strategy. A second improvement was made by using randomized projections reducing the overall volume of transmitted data greatly. Additionally, proofs for the equivalence of the methods to their centralized counterparts are provided.

Chapter 5 extends the Gram-Schmidt procedure developed for the third and fourth chapter to return the full QR decomposition and can hence be used for other applications such as linear regression as well. This manuscript has a stronger emphasis on privacy, investigating the Householder, Givens and Gram-Schmidt algorithms with respect to their privacy when transformed into a federated algorithm. We conclude that Gram-Schmidt decomposition is the most suited method. Even with this algorithm, privacy breaches can occur, if users are not careful or the attacker has auxiliary knowledge.

Chapter 6 investigates different strategies for federated K-Means clustering, and evaluates them under strong non-iid assumptions. The aim is to provide an adaptive clustering strategy which allows to find a good partition without specifying k. This should be achieved while remaining communication efficient and privacy-aware. The preliminary conclusion is that most well performing clustering strategies are also communication efficient. Outliers and strongly skewed data distributions lead to a worse performance of federated K-Means compared to the centralized algorithm. The solutions which include the determination of k do not quite achieve the same performance yet.

7.2 Communication and resource efficiency

Communication efficient PCA methods are available, but, as explained in chapter 2, they come with the drawback of immediately exposing the full covariance matrix, or accuracy loss. Therefore, we made an effort to reduce

the communication overhead of federated power iteration. In our current implementations, presented in chapter 4 the overhead is mainly due to the high number of iterations. The volume of transmitted data plays a minor role. For power iteration, the high number of communication steps is a disadvantage to its practical applicability, but the execution times are still reasonable, given that the data sourcing process is expensive and time consuming in medical studies. As detailed in chapters 3 to 4 for very large data, the computation of the covariance matrix is computationally very demanding and therefore not feasible. In this case, covariance-free methods, such as federated power iteration, are the only viable solution. The communication effort of randomized PCA can be reduced to a constant number of iterations. This can be achieved by sending the covariance matrix of the randomized projection of the data, instead of proceeding with power iteration. We will suggest this improvement in further iterations of the manuscript. This will make federated power iteration suitable for repeated application on high dimensional data.

In the federated Gram-Schmidt algorithm, the number of communication steps scales linearly with the number of columns (see chapter 3). According to our analyses in chapter 5, it outperforms a naïve implementation of the Householder reflection based algorithm. Due to it's parallelizability, Givens rotation may enjoy better communication properties, which remains to be determined. Generally, the algorithms can be implemented efficiently on centralized systems and do not require extensive compute power locally. In federated Gram-Schmidt factorization, the communicated parameter size is small. Therefore, the property which needs to be improved is the number of communication steps. Future work may try to provide approximate versions of the Gram-Schmidt algorithm to remove the dependence on the dimensionality of the data.

As shown in chapter 6, for federated K-means, the communication overhead is generally favorable. There are algorithms which, with limited data transmission are sufficient to achieve very good approximations of the centralized algorithm in the federated setting with a low number of iterations. Outliers and skewed data lead to slightly increase communication requirements, but overall they remain reasonable. Therefore, for the intended use case of cross-silo federated learning in the medical domain, K-Means is suited for routine application.

7.3 Accuracy

As detailed in chapter 2, systematic batch effects are assumed to be present in many biomedical data sets. Generally, the non-iid-ness of data is not a problem for the PCA and Gram-Schmidt orthonormalization algorithms developed in this thesis, as they are provably equivalent to their centralized counterparts, as shown in chapter 4. Approximate PCA should be avoided, given its application for batch effect detection, where the violation of iid-ness is assumed.

With regards to federated K-Means, in chapter 6, we come to the conclusion that most schemes cope reasonably well in different types of non-iid-ness in the literature when used on well separated, clean data. Outliers and edge cases in the data distribution skew, predictably lead to worse performance. Schemes which automatically determine k perform reasonably well, but improvements can be made in the future. A challenge in the evaluation of the algorithms with respect to their suitability for non-iid data is the high number of possible data configurations. We aimed at establishing a good trade-of between computational feasibility and coverage of the cases.

7.4 Privacy

In this thesis, federated learning in combination with secure additive aggregation is used as the federated learning paradigm. This means, the local parameter updates are hidden, but the aggregated parameters become known in clear text (see chapter 1).

In federated SVD, the amount of information encoded in these parameters is quite extensive. If the top left and right singular vectors become known, an approximation of the data matrix is possible. Therefore, in the context of GWAS, we suggested to only compute partial right eigenvectors, which prevents data approximation and further data reconstruction. If either the top left or top right singular vector become known, an approximation of the respective covariance matrix is possible. We established that the sample-by-sample covariance matrix cannot be shared, as it allows the computation of the full eigenvectors. An open question that remains, is how privacy preserving the feature-by-feature covariance matrix is in practice. In combination with the column mean it can be used to generate data representatives. These representatives may be used to create shadow models (Shokri et al. 2017; Pustozerova and Mayer 2021) and thus enable a privacy breach. In chapter 4, we show that the eigenvector updates in power

iteration can be used to recreate the full covariance matrix under some conditions. We therefore provided an algorithm which prevents this leakage for high-dimensional data. If the disclosure of the exact covariance matrix is found to be critical, hybrid federated learning is not a suitable option for federated principal component analysis, and only 'oracle' algorithms which perform the entire algorithm under encryption, or secure multiparty computation schemes are suitable. Even then, the approximate covariance matrix is disclosed once the top k subspace is announced. If the disclosure of an approximate matrix is deemed critical, the eigenvectors cannot be disclosed. In this case, it is also doubtful, that the analyst would allow a utility preserving differentially private release of the covariance matrix (Dwork et al. 2014). Even with DP eigenvectors, computing the projections of the data is not covered by the closure under post-processing. Overall, if a data analyst wants to disclose the result of PCA or SVD to the public, they need to have a very good understanding of the nature of the parameters, the eigenvectors. Even if measure are taken to prevent blatant privacy leakage, the publication of utility preserving PCA and SVD may be considered as in inherent opposition to strict privacy guarantees.

While investigating the federated QR algorithms in chapter 5, we showed that some algorithms are fundamentally unsuited to federated learning with secure aggregation. Notably, the recommended algorithms for centralized QR factorization, Givens rotation, and Householder reflection are not private, even if secure aggregation is used, due to the information encoded in their parameters. They could only be computed using more comprehensive secure-multiparty computation schemes. We came to the conclusion that the Gram-Schmidt algorithm is currently the most private algorithm, given the use of secure aggregation. The traceability of federated matrix computation make attacks cheap to implement and executable on limited hardware.

Lastly, a detailed investigation of the privacy of federated K-means will be required. The adaptive federated K-Means scheme intends to avoid the computation of multiple federated clusterings with multiple k over multiple initializations, preventing potential triangulation of individual data points. Nonetheless, in future work, an investigation of the potential privacy breach during the iterative subroutines will be necessary. Furthermore, the privacy loss due to the communication of the cluster covariance matrices needs to be evaluated.

7.5 Future directions

Currently, as part of the manuscripts proof-of-concept implementations and applications of various federated algorithms are available as pseudocode, simulation code, standalone federated tools and FeatureCloud apps. Providing secure and user-friendly tools including good documentation to the community is required to promote acceptance. Federated learning requires practitioners to understand two novel concepts at the same time: the federated algorithm and the relevant privacy-preserving techniques. Therefore, future work will include the continued development and deployment of the presented algorithms, for instance in the FeatureCloud AI Store. These user-friendly tools will include laymen friendly explanations and interfaces as well as privacy aware visualization of the results.

Regarding the practical application of federated tools for biomedical research, further effort needs to be spent on ensuring that the cumulative information disclosed during a federated study does not create privacy leaks. The majority of medical analyses use multiple algorithms and tools to generate or confirm hypotheses. One cannot assume that a workflow of individually private tools is privacy preserving in its entirety.

7.6 Conclusion

This thesis contributed to the field of federated biomedical machine learning by investigating, developing and testing algorithms with biomedical data in mind. At the current stage, proof-of-concept solutions have been implemented and the field is on the verge of shifting towards routine use. Unsupervised methods will play an important role in future federated studies.

At the same time, not all concerns regarding federated learning have been addressed. The research community is actively working on finding a compromise not only between the privacy of the individual and the utility of the model, but also between the privacy and mundane practicalities such as communication overhead.

Federated machine learning is a research area, that extends beyond the technical feasibility. By the means of technological solutions and the implementation of new policies, it aims to promote truly privacy-preserving machine learning.

References

Dwork, C. et al. (2014). "Analyze Gauss: Optimal bounds for privacy-preserving principal component analysis". In: *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 11–20. ISSN: 07378017. DOI: 10.1145/2591796.2591883.

Pustozerova, A. and Mayer, R. (2021). "Information Leaks in Federated Learning". In: February, pp. 1–6. DOI: 10.14722/diss.2020.23004.

Shokri, R. et al. (2017). "Membership Inference Attacks Against Machine Learning Models". In: *Proceedings - IEEE Symposium on Security and Privacy*, pp. 3–18. ISBN: 9781509055326. DOI: 10.1109/SP.2017.41. arXiv: 1610.05820.

Appendix A

Supplementary information

A.1 Web repositories

- `https://gitlab.com/hartebrodt/federated-k-means` – Simulation code for the federated K-Means algorithms and evaluation strategies presented in chapter 6.
- `https://gitlab.com/hartebrodt/federated_dp_pca` – PCA simulation code for horizontally and vertically partitioned PCA used for chapters 2 to 4.
- `https://github.com/AnneHartebrodt/fc-federated-pca` – The most recent version of the feature cloud App for federated PCA implementing various versions of PCA presented in chapter 2
- `https://github.com/AnneHartebrodt/hyfed-pca` – A standalone PCA tool based on the Hyfed framework (Nasirigerdeh et al. 2021). This tool is currently deployed at `federated.compbio.sdu.dk` however it will be replaced by the FeatureCloud app soon.
- `https://github.com/AnneHartebrodt/federated-qr` – Simulation code for QR decomposition and Linear regression used in chapter 5.

A.2 Additional Literature

The following tables contain annotated lists of relevant literature. The lists make no attempt to be complete, however they may be interesting to some readers.

Table A.1 – List of publications for federated PCA

FL Type	Title	Author (Year)	Privacy	Content
HFL	Global principal component analysis for dimensionality reduction in distributed data mining	Qi et al. 2003	-	Compute global covariance matrix at central aggregator
	Principal Component Analysis for Dimension Reduction in Massive Distributed Data Sets	Qu et al. 2002	-	Compute global covariance matrix
	Principal Component Analysis for Distributed Data Sets with Updating	Bai et al. 2005	-	Local QR decomposition, $\mathbf{R}$ is sent to the server, $\mathbf{R}$ matrices are merged using QR decomposition, SVD of the final $\mathbf{R}$
	Distributed PCA and k -Means Clustering	Liang et al. 2013	-	Compute Local SVD and announce $\Sigma_i^k, \mathbf{V}_i^{k\top}$ to the server. Reconstruct approximate covariance matrix $\mathbf{M} = \sum_i^I \mathbf{V}_i^k \Sigma_i^k \mathbf{V}_i^{k\top}$ and compute SVD of $\mathbf{M}$
	Improved Distributed Principal Component Analysis	Balcan, Kanchanapally, et al. 2014	-	Compute Local SVD and announce $\Sigma_i^k, \mathbf{V}_i^{k\top}$ to the server. Do not reconstruct approximate covariance matrix $\mathbf{M} = \sum_i^I \mathbf{V}_i^k \Sigma_i^k \mathbf{V}_i^{k\top}$ instead stack $\Sigma_i^k \mathbf{V}_i^{k\top}$ and compute SVD which is equivalent.
	An Improved Gap-Dependency Analysis of the Noisy Power Method	Balcan, Du, et al. 2016	DP	Distributed power iteration. The clients compute $\mathbf{H}_i = \mathbf{AG}$ and send it to the server. The computes $\mathbf{H}, X = \sum_i^I \mathbf{H}_i$ and shares $\mathbf{H}$ with the clients.
	Distributed stopping criteria for the power iteration applied to virus mitigation	Ramirez-Llanos and Martinez 2016	-	Introduces an alternative to the Rayleigh coefficient
	Privacy-preserving PCA on horizontally-partitioned data	Al-Rubaie et al. 2017	HE; Garbled circuits	Computation of the global covariance matrix by summing over the local covariance matrices.
	Distributed estimation of principal eigenspaces	Fan et al. 2019	-	Compute Local SVD and announce $\Sigma_i^k, \mathbf{V}_i^{k\top}$ to the server. Reconstruct approximate covariance matrix $\mathbf{M} = \sum_i^I \mathbf{V}_i^k \Sigma_i^k \mathbf{V}_i^{k\top}$ and compute SVD of $\mathbf{M}$
	A Review of Distributed Algorithms for Principal Component Analysis	Wu et al. 2018	-	Review article covering a subset of the articles described here
	A Distributed Principal Component Analysis Compression for Smart Seismic Acquisition Networks	B. Liu et al. 2018	-	Compute global covariance matrix

FL Type	Title	Author (Year)	Privacy	Content
	Differentially Private Distributed Principal Component Analysis	Imtiaz and Sarwate 2018	DP	Differentially private PCA based on sharing the noisy covariance matrices
	Distributed Differentially Private Computation of Functions with Correlated Noise	Imtiaz, Mohammadi, et al. 2019	DP	Differentially private PCA based on sharing the noisy covariance matrices with better noise scheme
	Privacy Preserving PCA for Multiparty Modeling	Y. Liu et al. 2020	HE & SMPC (Shamir)	Secure computation of the global covariance matrix
	Federated Principal Component Analysis	Grammenos et al. 2020	DP	Federated Streaming PCA with subspace merging based on
	Distributed Principal Component Analysis with Limited Communication	Alimisis et al. 2021	-	Riemannian Gradient Descent with Vector Quantization to reduce bit communication complexity.
	Distributed Estimation for Principal Component Analysis: An Enlarged Eigenspace Analysis	X. Chen, Lee, et al. 2021	-	Newtionian gradient descent using the Hessian on the first machine. Vectors of rank $k > 1$ are computed via matrix deflation.
	Federated Singular Vector Decomposition	Chai et al. 2021	Masking scheme & Secure aggregation	Data masking scheme allows to send masked data, compute the PCA globally and unmask the result locally.
VFL	Distributed Clustering Using Collective Principal Component Analysis	Kargupta et al. 2001		Projections of the data onto the first few Eigenvectors are sent to the aggregator which computes the global approximation of the data.
	A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data	Guo et al. 2012	-	Power iteration based on gradient descent.
	Federated Principal Component Analysis for Genome-Wide Association Studies	Hartebrodt et al. 2021		Power iteration based PCA for vertically partitioned data without sharing the sample eigenvectors.
	Differentially Private Principal Component Analysis over Horizontally Partitioned Data	S. Wang and Morris Chang 2019	HE & DP	Secure computation of noisy global covariance matrix
	Secure principal component analysis in multiple distributed nodes	Won et al. 2016	-	Computation of global covariance matrix and application case
HFL VFL	On the performance overhead tradeoff of distributed principal component analysis via data partitioning	An and Weber 2016	-	Empirical evaluation of Balcan, Kanchanapally, et al. 2014 and Kargupta et al. 2001 with respect to communication

FL Type	Title	Author (Year)	Privacy	Content
	Efficient protocols for principal eigenvector computation over private data	Pathak and Raj 2011	SMPC & HE	Secure PCA based on power iteration.
DFL	Efficient and Robust Fully Distributed Power Method with an Application to Link Analysis	Canright et al. 2005	-	Power iteration by weight propagation in a adjacency matrix
	Asynchronous Distributed Power Iteration with Gossip-Based Normalization	Jelasity et al. 2007		Extension of distributed power iteration
	A distributed eigensolver for loosely coupled networks	Straková and Gansterer 2013	-	QR based eigenvector computation using gossiping
	Asynchronous gossip principal components analysis	Fellus et al. 2015	-	PCA based on distributed averaging
	A Distributed Framework for Dimensionality Reduction and Denoising	Schizas and Aduroja 2015	-	PCA for sensor networks
CML	Principal Component Analysis	Jolliffe 2002	-	Reference literature for PCA
	Numerical Methods for Large Eigenvalue Problems	Saad 2011		General Introduction to Eigenvalue Problems
	Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions	Halko et al. 2011	-	Diverse PCA algorithms including randomized projection based PCA
	Analyze Gauss: Optimal bounds for privacy-preserving principal component analysis	Dwork et al. 2014	DP	Covariance perturbation method to compute private Eigenvectors
	The Noisy Power Method: A Meta Algorithm with Applications	Hardt and E. Price 2014	DP	Local Power Iteration with DP, Application to Streaming
	Coordinate-wise power method	Lei et al. 2016	-	Only coordinates which have not converged yet are updated
	Faster PCA and linear regression through hypercubes in HElib	Rathee et al. 2018	HE	Encrypted power iteration

Table A.2 – Examples of applications of PCA in bioinformatics

FL Type	Title	Author (Year)	Content
CML	Principal components analysis corrects for stratification in genome-wide association studies	A. L. Price et al. 2006	Interesting PCA
	Outlier-Robust PCA: The High-Dimensional Case	Xu et al. 2013	PCA which can deal with corrupted data
	Gene expression analysis identifies global gene dosage sensitivity in cancer	Fehrmann et al. 2015	Application of PCA
	Second-generation PLINK: Rising to the challenge of larger and richer datasets	Chang et al. 2015	plink – PCA for population stratification
	Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia	Galinsky et al. 2016	Fast principal component analysis for GWAS
	Consequences of PCA graphs, SNP codings, and PCA variants for elucidating population structure	Gauch et al. 2019	Application of PCA in GWAS
	Efficient integration of heterogeneous single-cell transcriptomes using Scanorama	Hie et al. 2019	Application of PCA for single cell integration
	Robust principal component analysis for accurate outlier sample detection in RNA-Seq data	X. Chen, Zhang, et al. 2020	PCA for outlier detection
	Unlabeled Principal Component Analysis	Yao et al. 2021	PCA which can deal with corrupted data (e.g. block permutations).
	FastPop: A rapid principal component derived method to infer intercontinental ancestry using genetic data	Li et al. 2016	PCA for population stratification
	Robust Subspace Learning: Robust PCA, Robust Subspace Tracking, and Robust Subspace Recovery	Vaswani et al. 2018	Outlier robust PCA

FL Type	Title	Author (Year)	Content
HFL	Fault detection in wastewater treatment plants using distributed PCA methods	Sanchez-Fernandez et al. 2015	Application of QR based PCA to wastewater treatment plant fault detection

Table A.3 – Publications related to other unsupervised learning

FL Type	UFL Type	Title	Author (Year)	Content
HFL	Tensor factorization	Federated Tensor Factorization for Computational Phenotyping	Kim et al. 2017	Tensor facotrization to detect phenotypes. Matrix is decomposed into a feature and a patient component, similar to PCA but non-linear.
HFL	C-Means	Federated FCM: Clustering Under Privacy Requirements	Pedrycz 2021	Fuzzy C-Means, objective function based clustering. Local rounds followed by global gradient update
HFL	Spectral clustering	Federated Multi-View Spectral Clustering	H. Wang et al. 2020	Spectral clustering with Homomorphic encryption (Based on federated PCA)

Table A.4 – Publications related to clustered FL

FL Type	Title	Author (Year)	Content
HFL	Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning	Z. Chen et al. 2021	Clustered Federated Learning: Minmax initialization and cluster splitting/merging strategy to cluster gradient updated. Clustering happens at the aggregator, the updates not the data itself is clustered.
HFL	ClusterGrad: Adaptive Gradient Compression by Clustering in Federated Learning	Cui et al. 2020	Clustered Federated Learning: Gradients are clustered before sending, to identify informative updates and average the rest.
HFL	FedGroup: Efficient Federated Learning via Decomposed Similarity-Based Clustering	Duan et al. 2021	Clustered Federated Learning: A novel Gradient update strategy

References

Alimisis, F. et al. (2021). "Distributed Principal Component Analysis with Limited Communication". In: NeurIPS. arXiv: 2110.14391. URL: http://arxiv.org/abs/2110.14391.

An, N. and Weber, S. (2016). "On the performance overhead tradeoff of distributed principal component analysis via data partitioning". In: *2016 50th Annual Conference on Information Systems and Sciences, CISS 2016*, pp. 578–583. DOI: 10.1109/CISS.2016.7460567.

Bai, Z.-J., Chan, R. H., and Luk, F. T. (2005). "Principal Component Analysis for Distributed Data Sets with Updating". In: *Advanced Parallel Processing Technologies*. Ed. by J. Cao, W. Nejdl, and M. Xu. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 471–483.

Balcan, M.-F., Du, S. S., et al. (2016). "An Improved Gap-Dependency Analysis of the Noisy Power Method". In: *29th Annual Conference on Learning Theory*. Proceedings of Machine Learning Research 49. Ed. by V. Feldman, A. Rakhlin, and O. Shamir, pp. 284–309. URL: http://proceedings.mlr.press/v49/balcan16a.html.

Balcan, M.-F., Kanchanapally, V., et al. (2014). "Improved Distributed Principal Component Analysis". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, pp. 3113–3121.

Canright, G., Engø-Monsen, K., and Jelasity, M. (2005). "Efficient and Robust Fully Distributed Power Method with an Application to Link Analysis". In: *Technical Report UBLCS-2005-17* September.

Chai, D. et al. (2021). *Federated Singular Vector Decomposition*. Vol. 1. 1. Association for Computing Machinery. arXiv: 2105.08925. URL: http://arxiv.org/abs/2105.08925.

Chang, C. C. et al. (2015). "Second-generation PLINK: Rising to the challenge of larger and richer datasets". In: *GigaScience* 4.1, pp. 1–16. ISSN: 2047217X. DOI: 10.1186/s13742-015-0047-8. arXiv: 1410.4803.

Chen, X., Lee, J. D., et al. (2021). "Distributed Estimation for Principal Component Analysis: An Enlarged Eigenspace Analysis". In: *Journal of the American Statistical Association*. ISSN: 1537274X. DOI: 10.1080/01621459.2021.1886937. arXiv: 2004.02336.

Chen, X., Zhang, B., et al. (2020). "Robust principal component analysis for accurate outlier sample detection in RNA-Seq data". In: *BMC Bioinformatics* 21.1, pp. 1–20. ISSN: 14712105. DOI: 10.1186/s12859-020-03608-0.

Chen, Z. et al. (2021). "Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning". In: *IEEE Transactions on Network Science and Engineering* 8.2, pp. 1070–1083. ISSN: 2327-4697. DOI: 10.1109/TNSE.2020.3002796. URL: https://ieeexplore.ieee.org/document/9119145/.

Cui, L. et al. (2020). "ClusterGrad: Adaptive Gradient Compression by Clustering in Federated Learning". In: *2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings*. DOI: 10.1109/GLOBECOM42002.2020.9322527.

Duan, M. et al. (2021). "FedGroup: Efficient Federated Learning via Decomposed Similarity-Based Clustering". In: *2021 IEEE Intl Conf on Parallel Distributed Processing with Applications*, pp. 228–237. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00042.

Dwork, C. et al. (2014). "Analyze Gauss: Optimal bounds for privacy-preserving principal component analysis". In: *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 11–20. ISSN: 07378017. DOI: 10.1145/2591796.2591883.

Fan, J. et al. (2019). "Distributed estimation of principal eigenspaces". In: *Annals of Statistics* 47.6, pp. 3009–3031. ISSN: 21688966. DOI: 10.1214/18-AOS1713. arXiv: 1702.06488.

Fehrmann, R. S. et al. (2015). "Gene expression analysis identifies global gene dosage sensitivity in cancer". In: *Nature Genetics* 47.2, pp. 115–125. ISSN: 15461718. DOI: 10.1038/ng.3173.

Fellus, J., Picard, D., and Gosselin, P.-H. (2015). "Asynchronous gossip principal components analysis". In: *Neurocomputing* 169, pp. 262–271. ISSN: 09252312. DOI: 10.1016/j.neucom.2014.11.076. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231215003628.

Galinsky, K. J. et al. (2016). "Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia". In: *The American Journal of Human Genetics* 98.3, pp. 456–472. ISSN: 00029297. DOI: 10.1016/j.ajhg.2015.12.022. URL: http://dx.doi.org/10.1016/j.ajhg.2015.12.022%20https://linkinghub.elsevier.com/retrieve/pii/S0002929716000033.

Gauch, H. G. et al. (2019). "Consequences of PCA graphs, SNP codings, and PCA variants for elucidating population structure". In: *PLoS ONE* 14.6, pp. 1–26. ISSN: 19326203. DOI: 10.1371/journal.pone.0218306.

Grammenos, A. et al. (2020). "Federated Principal Component Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 6453–6464. URL: https://proceedings.neurips.cc/paper/2020/file/47a658229eb2368a99f1d032c8848542-Paper.pdf.

Guo, Y. F. et al. (2012). "A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data". In: *Pattern Recognition* 45.3, pp. 1211–1219. ISSN: 00313203. DOI: 10.1016/j.patcog.2011.09.002. URL: http://dx.doi.org/10.1016/j.patcog.2011.09.002.

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions". In: pp. 1–74. arXiv: arXiv:0909.4061v2.

Hardt, M. and Price, E. (2014). "The Noisy Power Method: A Meta Algorithm with Applications". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14, pp. 2861–2869.

Hartebrodt, A. et al. (2021). "Federated Principal Component Analysis for Genome-Wide Association Studies". In: Icdm, pp. 1090–1095. DOI: 10.1109/ICDM51629.2021.00127.

Hie, B., Bryson, B., and Berger, B. (2019). "Efficient integration of heterogeneous single-cell transcriptomes using Scanorama". In: *Nature Biotechnology* 37.6, pp. 685–691. ISSN: 15461696. DOI: 10.1038/s41587-019-0113-3. URL: http://dx.doi.org/10.1038/s41587-019-0113-3.

Imtiaz, H., Mohammadi, J., and Sarwate, A. D. (2019). "Distributed Differentially Private Computation of Functions with Correlated Noise". In: pp. 1–40. arXiv: arXiv:1904.10059v1.

Imtiaz, H. and Sarwate, A. D. (2018). "Differentially Private Distributed Principal Component Analysis". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 2206–2210. ISBN: 978-1-5386-4658-8. DOI: 10.1109/ICASSP.2018.8462519. URL: https://ieeexplore.ieee.org/document/8462519/.

Jelasity, M., Canright, G., and Engø-Monsen, K. (2007). "Asynchronous Distributed Power Iteration with Gossip-Based Normalization". In: *Euro-Par 2007*, pp. 514–525. DOI: 10.1007/978-3-540-74466-5_55. URL: http://link.springer.com/10.1007/978-3-540-74466-5_55.

Jolliffe, I. (2002). *Principal Component Analysis*. Springer-Verlag. DOI: 10.1007/b98835. URL: https://doi.org/10.1007/b98835.

Kargupta, H. et al. (2001). "Distributed Clustering Using Collective Principal Component Analysis". In: *Knowledge and Information Systems*. DOI: 10.4324/9781315799476-12.

Kim, Y. et al. (2017). "Federated Tensor Factorization for Computational Phenotyping". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. 176. 12. New York, NY, USA: ACM, pp. 887–895. ISBN: 9781450348874. DOI: 10.1145/3097983.3098118. URL: https://dl.acm.org/doi/10.1145/3097983.3098118.

Lei, Q., Zhong, K., and Dhillon, I. S. (2016). "Coordinate-wise power method". In: *Advances in Neural Information Processing Systems* Nips, pp. 2064–2072. ISSN: 10495258.

Li, Y. et al. (2016). "FastPop: A rapid principal component derived method to infer intercontinental ancestry using genetic data". In: *BMC Bioinformatics* 17.1, pp. 1–8. ISSN: 14712105. DOI: 10.1186/s12859-016-0965-1. URL: http://dx.doi.org/10.1186/s12859-016-0965-1.

Liang, Y., Balcan, M.-f., and Kanchanapally, V. (2013). "Distributed PCA and k -Means Clustering". In: *The Big Learning Workshop in NIPS 2013*, pp. 1–8.

Liu, B. et al. (2018). "A Distributed Principal Component Analysis Compression for Smart Seismic Acquisition Networks". In: *IEEE Transactions on*

Geoscience and Remote Sensing 56.6, pp. 3020–3029. ISSN: 01962892. DOI: 10.1109/TGRS.2018.2789354.

Liu, Y. et al. (2020). "Privacy Preserving PCA for Multiparty Modeling". In: arXiv: 2002.02091. URL: http://arxiv.org/abs/2002.02091.

Nasirigerdeh, R. et al. (2021). "HyFed: A Hybrid Federated Framework for Privacy-preserving Machine Learning". In: *CoRR* abs/2105.10545. arXiv: 2105.10545. URL: https://arxiv.org/abs/2105.10545.

Pathak, M. A. and Raj, B. (2011). "Efficient protocols for principal eigenvector computation over private data". In: *Transactions on Data Privacy* 4.3, pp. 129–146. ISSN: 18885063.

Pedrycz, W. (2021). "Federated FCM: Clustering Under Privacy Requirements". In: *IEEE Transactions on Fuzzy Systems* 6706.c, pp. 1–6. ISSN: 19410034. DOI: 10.1109/TFUZZ.2021.3105193.

Price, A. L. et al. (2006). "Principal components analysis corrects for stratification in genome-wide association studies". In: *Nature Genetics* 38.8, pp. 904–909. ISSN: 10614036. DOI: 10.1038/ng1847.

Qi, H., Wang, T. W., and Birdwell, J. D. (2003). "Global principal component analysis for dimensionality reduction in distributed data mining". In: *Statistical Data Mining and Knowledge Discovery*. Chapman and Hall/CRC, pp. 323–338. ISBN: 9780203497159. DOI: 10.1201/9780203497159.ch19. URL: http://www.crcnetbase.com/doi/10.1201/9780203497159.ch19.

Qu, Y. et al. (2002). "Principal Component Analysis for Dimension Reduction in Massive Distributed Data Sets". In: *Workshop on High Performance Data Mining at the Second SIAM International Conference on Data Mining*, pp. 4–9.

Ramirez-Llanos, E. and Martinez, S. (2016). "Distributed stopping criteria for the power iteration applied to virus mitigation". In: *Conference Record - Asilomar Conference on Signals, Systems and Computers* 2016-Febru, pp. 1328–1332. ISSN: 10586393. DOI: 10.1109/ACSSC.2015.7421358.

Rathee, D., Mishra, P. K., and Yasuda, M. (2018). "Faster PCA and linear regression through hypercubes in HElib". In: *Proceedings of the ACM Conference on Computer and Communications Security* 1, pp. 42–53. ISSN: 15437221. DOI: 10.1145/3267323.3268952.

Al-Rubaie, M. et al. (2017). "Privacy-preserving PCA on horizontally-partitioned data". In: *2017 IEEE Conference on Dependable and Secure Computing*, pp. 280–287. DOI: 10.1109/DESEC.2017.8073817.

Saad, Y. (2011). *Numerical Methods for Large Eigenvalue Problems*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics. ISBN: 978-1-61197-072-2. DOI: 10.1137/1.9781611970739. URL: https://epubs.siam.org/doi/book/10.1137/1.9781611970739.

Sanchez-Fernandez, A., Fuente, M., and Sainz-Palmero, G. (2015). "Fault detection in wastewater treatment plants using distributed PCA methods". In: *2015 IEEE 20th Conference on Emerging Technologies & Factory Automa-*

tion (ETFA). IEEE, pp. 1–7. ISBN: 978-1-4673-7929-8. DOI: 10.1109/ETFA.2015.7301504. URL: http://ieeexplore.ieee.org/document/7301504/.

Schizas, I. D. and Aduroja, A. (2015). "A Distributed Framework for Dimensionality Reduction and Denoising". In: *IEEE Transactions on Signal Processing* 63.23, pp. 6379–6394. ISSN: 1053587X. DOI: 10.1109/TSP.2015.2465300.

Straková, H. and Gansterer, W. N. (2013). "A distributed eigensolver for loosely coupled networks". In: *Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013*, pp. 51–57. DOI: 10.1109/PDP.2013.18.

Vaswani, N. et al. (2018). "Robust Subspace Learning: Robust PCA, Robust Subspace Tracking, and Robust Subspace Recovery". In: *IEEE Signal Processing Magazine* 35.4, pp. 32–55.

Wang, H. et al. (2020). "Federated Multi-View Spectral Clustering". In: *IEEE Access* 8, pp. 202249–202259. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3036747.

Wang, S. and Morris Chang, J. (2019). "Differentially Private Principal Component Analysis over Horizontally Partitioned Data". In: *DSC 2018 - 2018 IEEE Conference on Dependable and Secure Computing*, pp. 1–8. DOI: 10.1109/DESEC.2018.8625131.

Won, H.-S. et al. (2016). "Secure principal component analysis in multiple distributed nodes". In: *Security and Communication Networks* 9.14, pp. 2348–2358. ISSN: 19390114. DOI: 10.1002/sec.1501. arXiv: 0806.0557. URL: http://arxiv.org/abs/0806.0557%20http://doi.wiley.com/10.1002/sec.1501.

Wu, S. X. et al. (2018). "A Review of Distributed Algorithms for Principal Component Analysis". In: *Proceedings of the IEEE* 106.8, pp. 1321–1340. ISSN: 00189219. DOI: 10.1109/JPROC.2018.2846568.

Xu, H., Caramanis, C., and Mannor, S. (2013). "Outlier-Robust PCA: The High-Dimensional Case". In: *IEEE Transactions on Information Theory* 59.1, pp. 546–572.

Yao, Y., Peng, L., and Tsakiris, M. C. (2021). "Unlabeled Principal Component Analysis". In: *CoRR* abs/2101.09446. arXiv: 2101.09446. URL: https://arxiv.org/abs/2101.09446.

Appendix B

Supplementary Material for Manuscript 1: Federated Horizontally Partitioned Principal Component Analysis for Biomedical Applications

Supplementary Material to Federated Horizontally Partitioned Principal Component Analysis for Biomedical Applications

Anne Hartebrodt* and Richard Röttger*

February 13, 2022

1 Visualization of the sample distribution in TCGA according to TSS

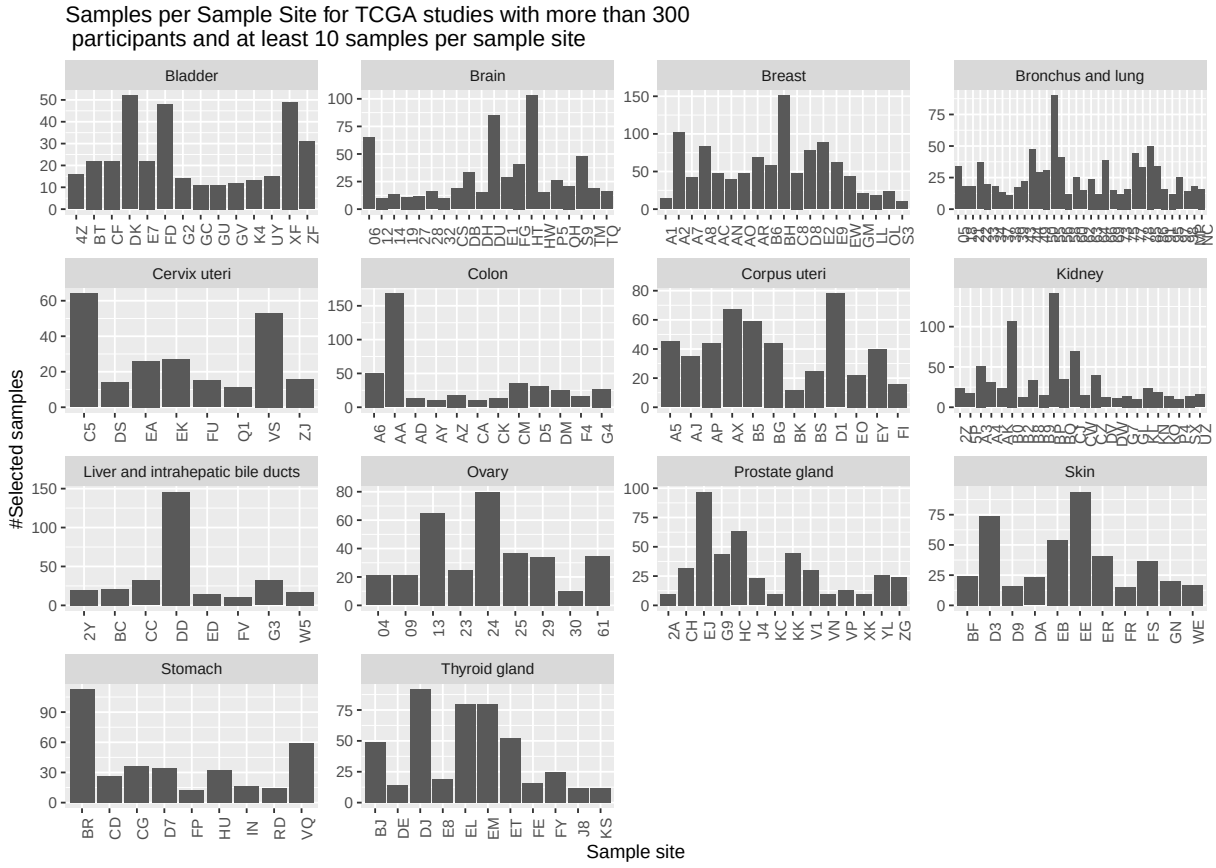


Figure 1: Distribution of samples over the tissue collection sites for different cancer types. The data was downloaded from TCGA. Selected studies had to contain at least 300 participants and study sites with fewer than 10 samples were excluded.

*Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230 Odense, Denmark

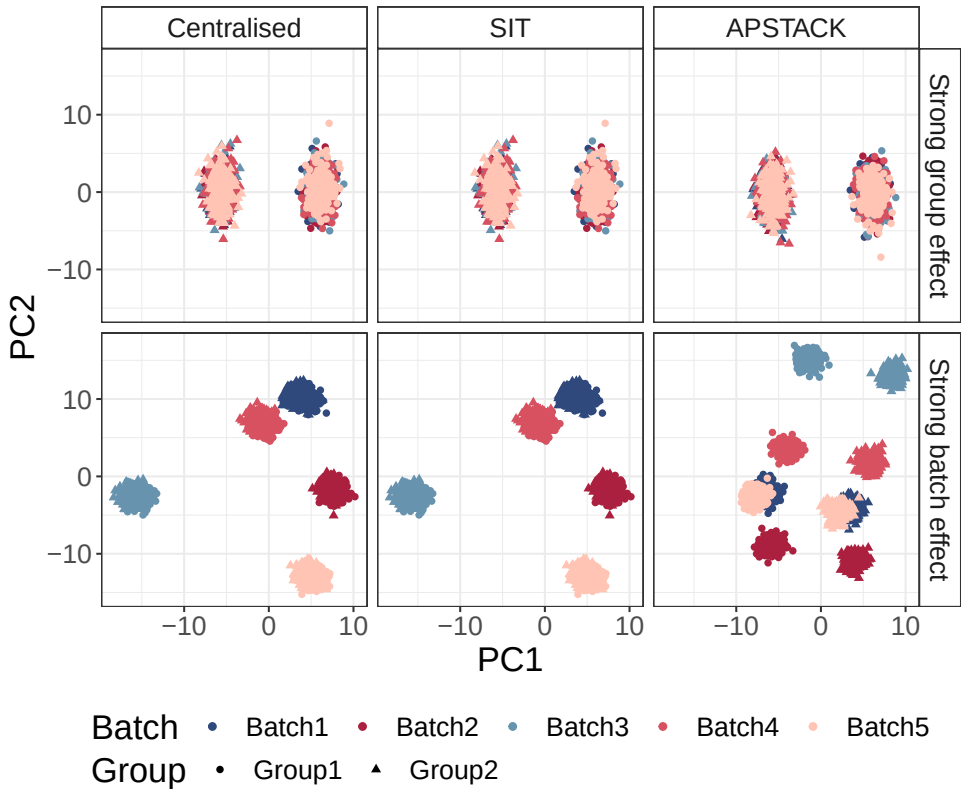


Figure 2: Comparison of centralized PCA, `SUB-IT` and `AP-STACK` using simulated single cell count data. The upper panel depicts data where the group (e.g. case-control) is the strongest variance driving effect across all sites. The lower panel shows data with a strong batch effect. `AP-STACK` is not able to reconstruct the embedding faithfully with strong batch effects.

1.1 Application of federated PCA to simulated count data

Figure 2 shows a comparison of centralized, approximate (`AP-STACK`) and exact federated PCA (`SUB-IT`) using two different simulated data sets. The first data set contains simulated count data without strong batch effects, meaning that the group effect is the major driver of variation in the data. In this scenario, both federated methods reconstruct the low dimensional embedding faithfully to the centralized algorithm which clusters the data according to the group label. The second data set contains count data with a simulated batch effect that is stronger that the group effect. The centralized algorithm clusters the data according to the batch variable. `SUB-IT` computes the same representation, whereas `AP-STACK` does not properly separate the batches, but instead separates the groups. The reason for this behavior is the fact, that at one site, the major driver of variance is the group. The separation of the groups seems still reasonable. This can be explained by the fact that the batch effect simulation tool adds the same systematic batch effect to all 'sites', therefore the first eigenvector recovers this effect. This is an unrealistic assumption on batch effects. In reality, there would likely be different batch effects at each site, leading to eigenvectors which point in different directions. In that case the groups would not be recovered like they are in the present example. Nonetheless, this simple example is sufficient to show the unsuitability of federated approximate PCA to recover batch effects.

2 Drop out experiments

It is a known problem of centralized principal component analysis that it is vulnerable to outliers [7]. Outliers are data points which deviate strongly from the overall trend. Since PCA is a variance based linear decomposition of the covariance matrix, single data points which are 'far' from the rest can have a very strong influence on the eigenvectors. The fewer samples are available, the higher the impact of an outlier. In order to quantify this problematic with respect to our quality criterion the angle between the eigenvectors, we include an simulation, where we drop single data points from the data. Specifically, we performed the following computations: The exact PCA of a data set was computed. Then a random sample was removed from the data and the exact PCA was computed again. The angle between the original eigenvector and the 'dropout' eigenvector was computed. Figure 3 shows the effect of dropping single samples from the data and recomputing the PCA. From the box plots it can be seen that most samples do not lead to a strong deviation of the eigenvector from the original eigenvector, but some distort the result strongly. Therefore, in a low sample regime, the approximate methods do not perform well. Compared to the angles expected through approximate PCA in fig. 4, these angles are generally lower. This means that one can expect a higher 'instability' of of the results through

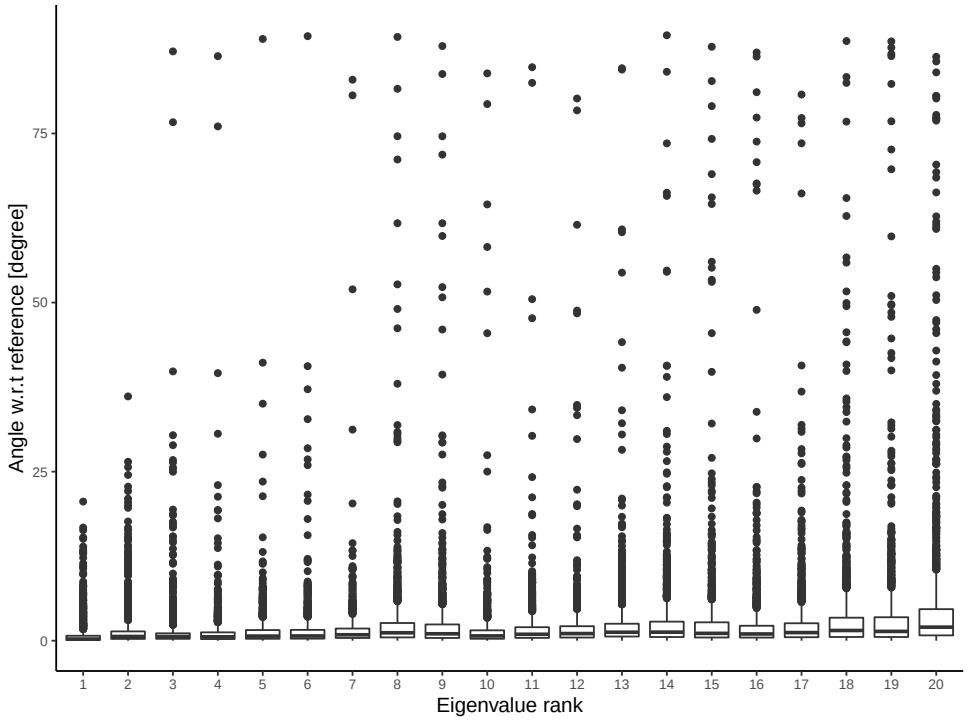


Figure 3: [Power iteration] Angles with respect to the gold standard when leaving out an individual sample. The plot presents aggregated statistics over all the data sets from TCGA, but all the cancer types were treated as individual data sets.

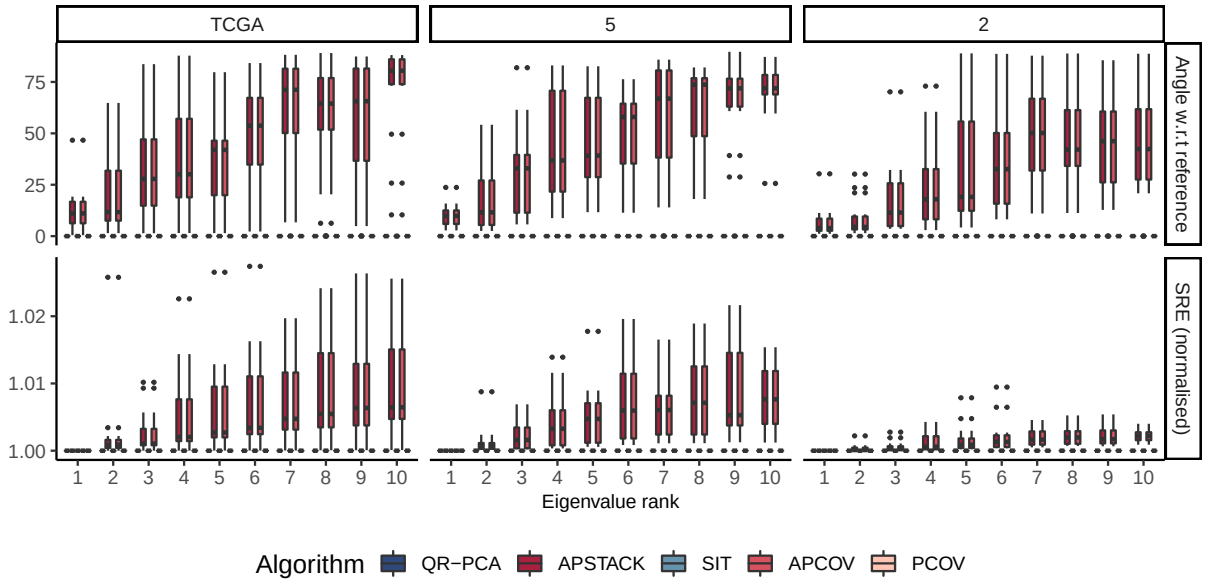


Figure 4: Extended PCA accuracy plot: Comparison of the different PCA algorithms w. r. t. to the angle between the leading eigenvectors with the TCGA data distributed according to the tissue sample site, and combined into 2 and 5 meta sites respectively. Proxy naive and the Power method achieve perfect accuracy both according to the angle between the eigenvectors (upper panel) and the subspace reconstruction error (SRE). With higher rank, the accuracy of the proxy method deteriorates.

3 Practical utility study

3.1 Data

For the practical application case, we use two publicly available single-cell data sets, a PBMC dataset from 10X Genomics (`PBMC`), and a myeloid progenitor data set [5] (`PAUL`). The data sets consist of 2638 cells $\times$ 1838 genes and 2730 cells $\times$ 3451 genes respectively. After preprocessing according to standard protocols as presented in the scanpy vignettes [2, 3], the single cell measurements are split horizontally into 5 batches which simulate the sites. The preprocessing of the vignettes is not replicated in a federated fashion, however it could be reproduced as it relies mostly on summary statistics such as mean and variance which can be computed easily in a federated fashion. This arbitrary split disregards potential batch effects related to the experimental protocol, as all the cells stem from the same experiment, but is sufficient to illustrate important pitfalls and challenges of federated machine learning in general.

3.2 Setup

In order to put the simulation results into perspective, we choose three popular single-cell RNASeq applications [1], namely data visualization, data clustering and gene importance scoring as application studies. To illustrate the use of federated PCA in these use cases, we implement standard workflows[2, 3] using regular PCA and using simulated federated code for approximate PCA on the two single cell data sets (`PAUL`, `PBMC`). We follow the same steps as the vignettes. We set $k = 50$, the suggested default, resulting in 100 transmitted intermediate dimensions, when using a factor $k' = 2$. Here, we only compare the results of the approximate federated PCA algorithms since the exact methods do not differ from the centralized case at the expense of computational and communication costs. However, due to the lower number of communication rounds and reduced amount of communicated data, the approximate methods speed up the analysis significantly. Therefore, we investigate whether the results of approximate PCA would be acceptable in practice.Many standard single cell analysis pipelines rely on UMAP [4] as a low dimensional representation of the data.

Often, the UMAP is computed on the projections of the data onto the first few eigenvectors, in an attempt to overcome the curse of dimensionality. Since we use the projections of the data, federated PCA methods which lead to a small subspace reconstruction error may be sufficient to reproduce the conclusions of the centralized protocol even when the angles of the eigenvectors differ considerably from the centralized version. As an additional downstream analysis relying on the projected data, we include a reproduction of clustering of the `PAUL` data set results using the Leiden clustering [6] and the PAGA [8] algorithm. In order to identify correlated and important genes, the eigenvectors themselves (loadings) can be analyzed. For example, the importance of the genes can be scored using the most extreme positive or negative entries of the eigenvector.

3.3 Results

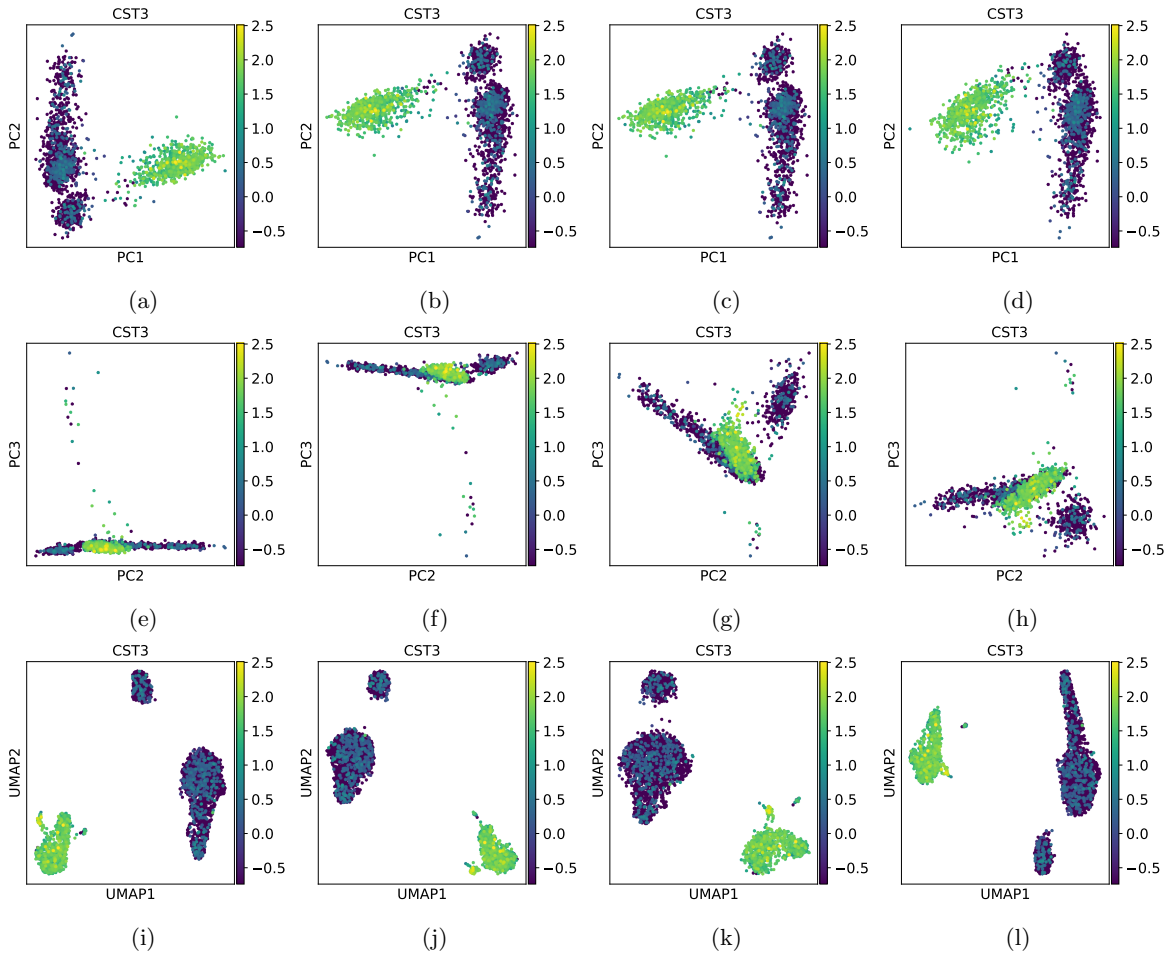


Figure 5: Example application of PCA for single cell RNASeq analysis with different data distributions. a-d) PC1 vs. PC2; e-h) P2 vs. P3; i-l) UMAP1 vs. UMAP2; a,e&i) centralised PCA (baseline); b,f&j) Approximate federated PCA with favorable data/outlier distribution; c,g&k) approximate PCA with unfavorable outlier distribution (one site did not contain outliers, removing an axis of variation); d,h&l) approximated PCA with data split according to the Leiden clustering of data, see supplementary fig. 6. Most visual representations are quite faithful, disregarding the flip of the eigenvectors. Subfigure (g) deviates from the centralized baseline, and the angle between the corresponding eigenvectors is high (see also 9 (a)).

UMAP visualization In fig. 5 the results of approximate federated PCA and the original results for the PBMC data are shown side by side with different data configurations. The first column shows the baseline PC1 vs. PC2; PC2 vs. PC3; and UMAP1 vs. UMAP2 plots, the second and third columns show the same

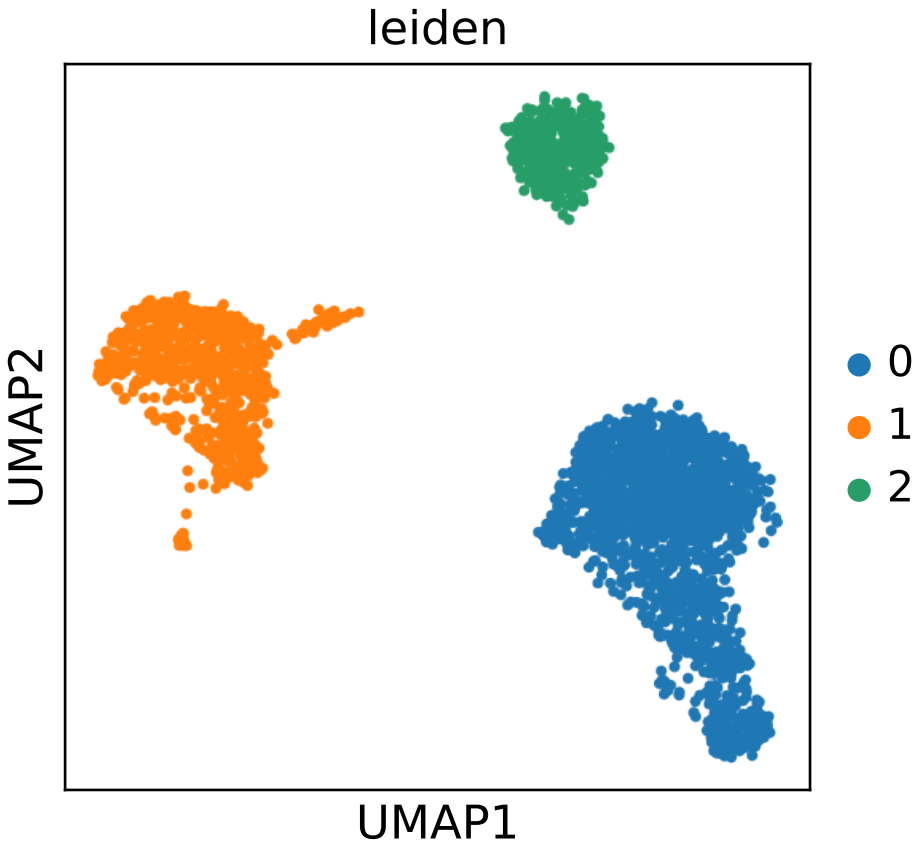
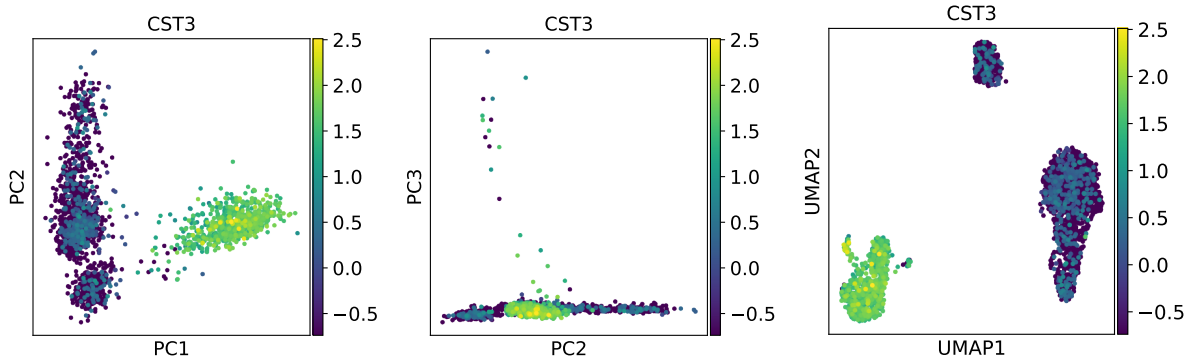
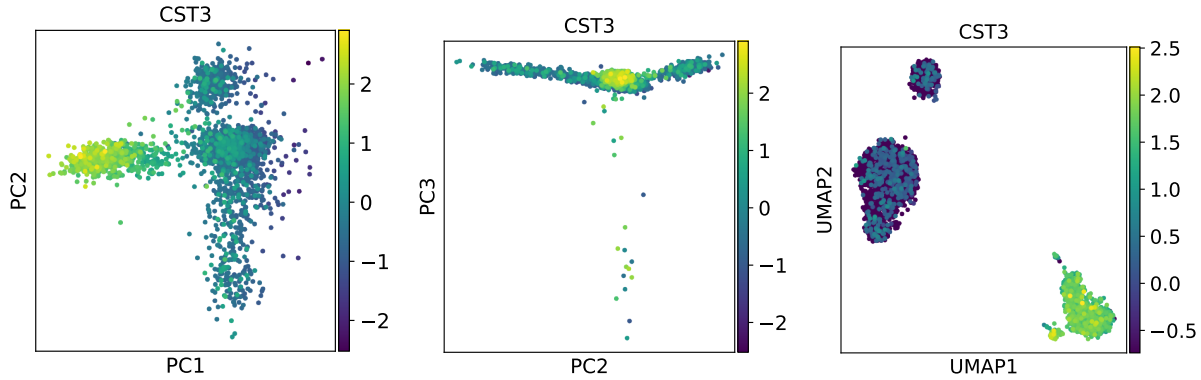


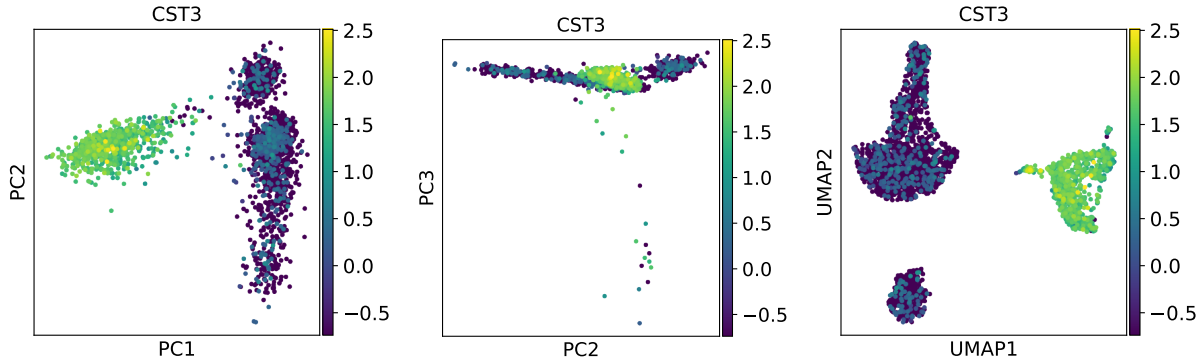
Figure 6: Leiden groups with 3 clusters, using resolution 0.1 to obtain a coarse clustering of the PBMC data. This was done with the centralized workflow. This clustering was then used to create three artificial data sites, each receiving one cluster.



(a) Exact PCA



(b) Approximate PCA (k=10)



(c) Approximate PCA (k=5)

Figure 7: Example application of PCA for single cell RNASeq analysis using the PBMC data set: UMAP visualizations a) reproduced from the vignette and b) reproduced using approximate federated PCA with k=10 and c) k=5 respectively. Shown are the first vs. second and second vs. third PCs plotted against each other, as well as the UMAP plot for the first two dimensions. The cluster structure is maintained even with lower dimensional approximations of the data. For UMAP visualisation purposes, only few eigenvectors/projections need to be transmitted which is potentially more privacy preserving. Outliers were distributed equally among the sites.

plots for different randomized data splits via different random seeds. The last column contains the figures generated when splitting the data such that each of the three clusters identified by Leiden clustering of the original UMAP embedded space represents a data site (sup. fig. 6). There is not a large visual difference for the plots showing the first two PCs (up to a rotation) for any of the configurations. For the PC2 vs. PC3 plots (2nd row), however, the embedding fig. 5 (g) looks different. Table 9a shows the associated angles between the two randomized splits, corresponding to columns 2 and 3, and the baseline, which reflect a quite strong deviation of the eigenvectors from the centralized baseline for the latter. The associated UMAP plot (fig. 5 (k)) preserves the three clusters apparent in the original plot.

Upon close investigation, the difference between the eigenvectors using different seeds stem from an unequal distribution of the outliers, for instance the outliers identifiable visually on the third principal axis. These data points create an axis of high variablity, but considering the bulk of the data they may represent erroneous measurements or cells that should be excluded (e.g. dying cells). In the tests, we created arbitrary data partitions, which included those outliers. If the outliers are distributed equally, meaning every artificial site contains some outliers, then the results remain similar to the centralized solution, because the axes of variation in the data are retained. If there are sites which do not have these outliers, this axis of variation is removed from the data and the results are distorted with respect to the centralized solution. In figure fig. 5 (g) the result deteriorates with respect to the centralized solution. A similar result is presented in subfigure (h) where the data is split according to the clustering.

In fig. 7 we include additional figures using a lower k showing essentially the same low dimensional embeddings. This means, the representation can be reproduced using a lower number of eigenvectors. This is favorable for the transmission cost and the privacy.

Downstream analysis – clustering As a downstream analysis relying on the projected data, we include a reproduction of clusterings using the Leiden clustering [6] and the PAGA [8] algorithm. These results are visualized in fig. 8. Using approximate PCA, the pipeline identifies 2 states more than with canonical PCA. The layout of the diffusion maps and graphs are different. While the general structure of the graph seems to be preserved, the upper module of the diffusion map may be prone to different interpretation.

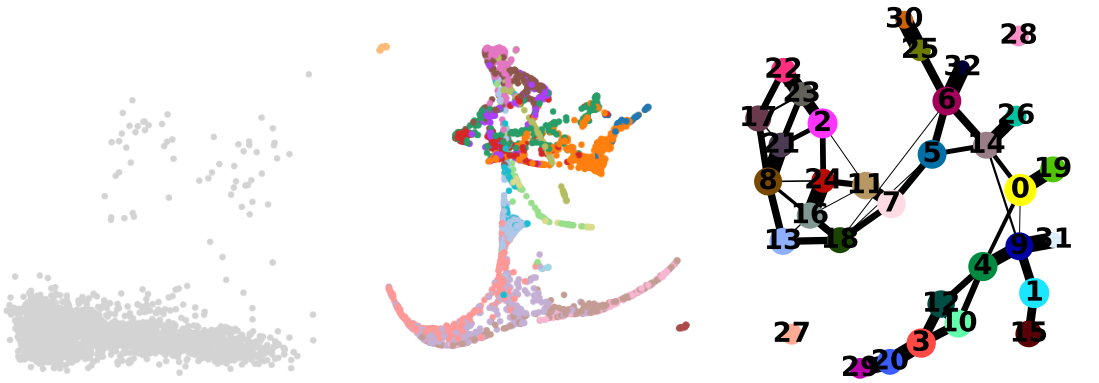
In order to quantify the likeness of the clusterings we compute the macro F1 score, and cluster specific precision and recall. The procedure is as following: first the cluster labels are matched using a global contingency matrix. Labels as matched if they have the highest number of points in common. Then for every cluster precision and recall are calculated:

$$P = \frac{TP}{TP + FP} \tag{1}$$

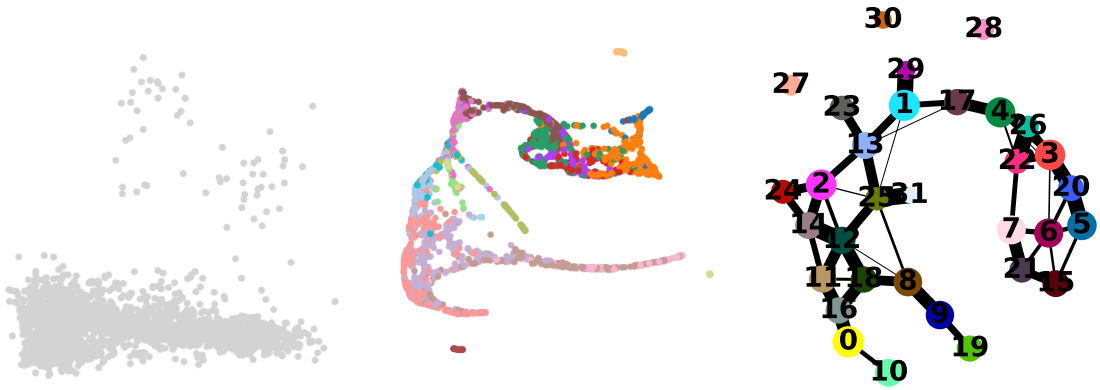
$$R = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = \sum \frac{2 \cdot P \cdot R}{P + R} * \frac{1}{\#Samples} \tag{3}$$

The macro F1 score is 0.47, cluster wise precision and recall are shown in section 3.3. Generally, there is some overlap between the clusters but the overall results differ quite strongly from the centralized original solution. We deliverately chose the standard settings, it may be possible to obtain better results which are closer to the original analysis using other parameters, however, we conclude that even for downstream analyses, other than visualization of the first few PCs, approximate PCA is unsuited.



(a) Exact PCA



(b) Approximate PCA

Figure 8: Example application of PCA for single cell RNASeq analysis of the `PAUL` data set: Denoised Leiden clustering and PAGA trajectories a) adapted from the vignette and b) reproduced using approximate federated PCA. Section 3.3 contains quantitative measures on the reproduced figures. Although they look similar, many clusters are not identical.

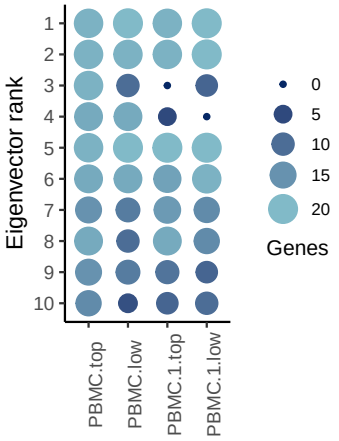
Cluster	precision	recall	Cluster	precision	recall
0	0.68	0.54	17	0.88	0.72
1	0.71	0.53	18	0.61	0.46
2	0.88	0.79	19	0.81	0.55
3	0.54	0.68	20	0.00	0.00
4	0.57	0.35	21	0.46	0.90
5	0.00	0.00	22	0.00	0.00
6	0.48	0.64	23	0.29	0.44
7	0.45	0.54	24	0.41	0.55
8	0.60	0.66	25	0.00	0.00
9	0.51	0.55	26	0.00	0.00
10	0.70	0.62	27	1.00	1.00
11	0.00	0.00	28	1.00	1.00
12	0.54	0.44	29	0.28	0.97
13	0.00	0.00	30	0.26	1.00
14	0.59	0.55	31	1.00	1.00
15	0.79	0.69	32	0.13	1.00
16	0.35	0.65			

Table 1: Clusterwise precision and recall comparing clustering results of exact and approximate clusterings. The macro F1 score is 0.47. The cluster labels are the labels from the approximate clustering because it yielded more clusters.

Analysis of loadings In figure 9 we show the cardinality of the overlap of the set of genes identified by using the top 20 largest positive and negative coordinates of the eigenvectors and the associated angle between the eigenvectors for two different random data partitions of the `PBMC` data. While small changes in the angle only lead to minor changes in the gene set, high deviations in the angles also lead to different sets of genes identified. This process is especially vulnerable to outliers which is backed by the observation that the overlap between the baseline genes and the genes identified with approximate PCA is lower for the more distorted eigenvectors where the 'outliers' are distributed unfavorably.

1	1.11	1.61
2	2.18	2.21
3	18.15	74.76
4	17.76	74.74
5	8.63	8.34
6	11.48	11.79
7	17.48	19.39
8	26.61	22.32
9	31.92	47.34
10	40.82	36.87
	PBMC	PMBC.1

(a) Angles



(b) Overlap

Figure 9: Single cell examples: 9a) Angles between the leading centralized and approximate eigenvectors for the `PBMC` dataset using different random seeds (PMBC and PBMC.1), where PBMC.1 has unequally distributed outliers, where some sites did not recieve any. Especially the 3rd and 4th eigenvector have high divergence. 9b Overlap between the genes with the top 20 most positive and most negative loadings identified using approximate and exact PCA. The higher the angle between the eigenvector the lower the concordance of the identified gene sets.

4 Application to Psoriasis data

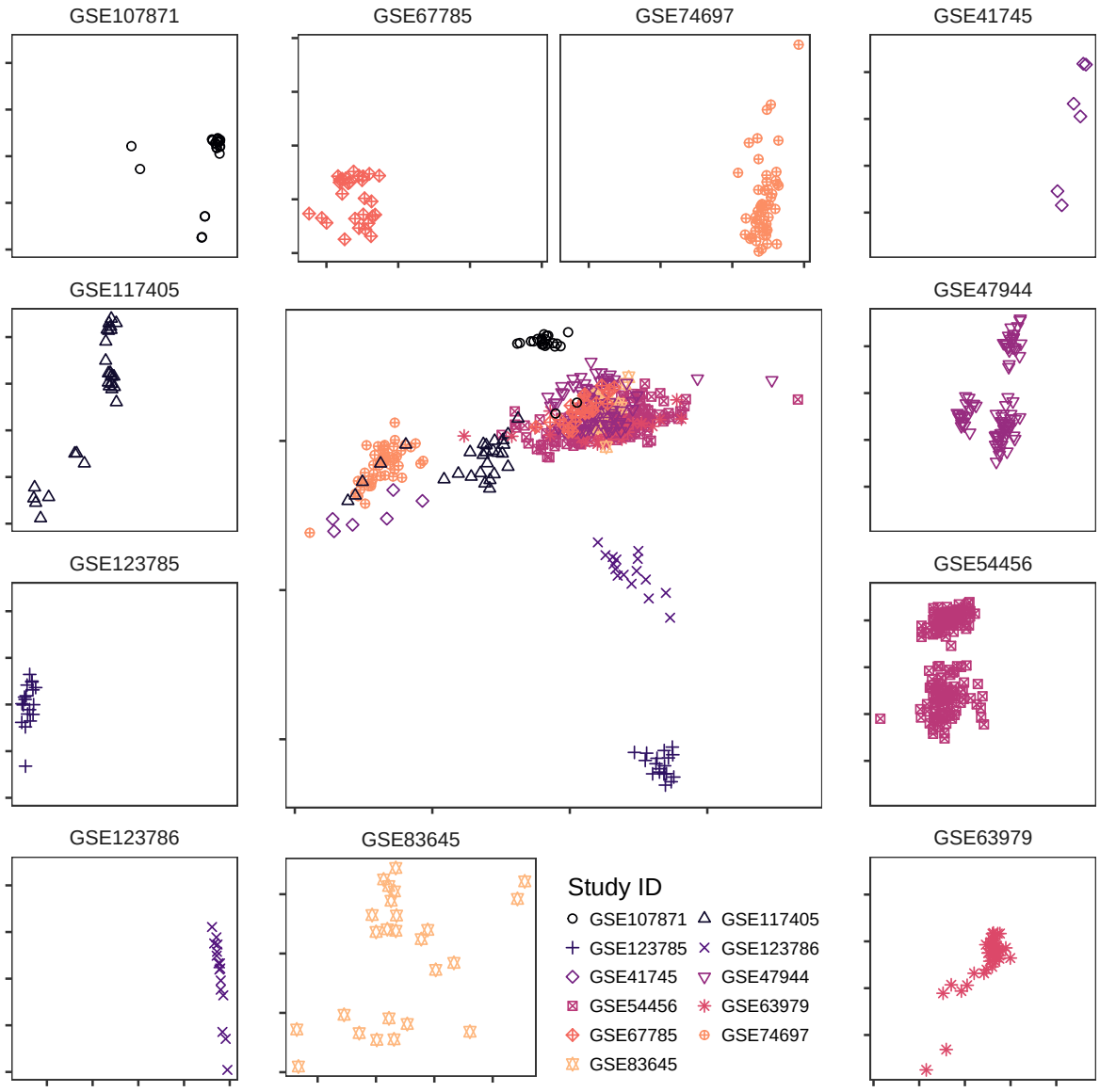


Figure 10: Comparison of the centralized and local PCA. The individual plots do not allow to obtain an overview of the data in the same manner than federated PCA. Furthermore, apparent outliers such as in GSE107871 may not be outliers in the larger analysis.

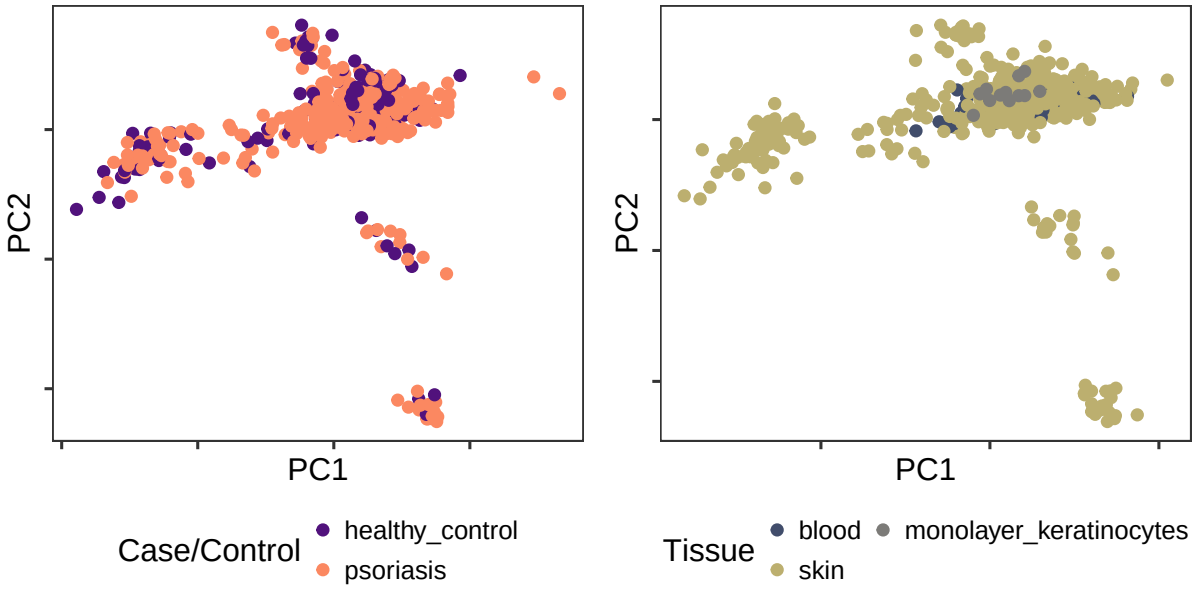


Figure 11: Further sample visualization to visualize the samples with respect to other covariates. For further downstream clustering, the batch effect (see fig. 10) would have to be removed. Differential Gene expression tools such as Flimma could be used to find differentially expressed genes in a federated fashion. The model employed in Flimma [9] accounts for batch effects. In this case, the PCA could be used for sample selection. For this figure, instead of using the original data projections, the mean and covariance matrix of the projected data was computed. Then, using a multivariate Gaussian distribution, artificial data points were created to visualize the samples at the aggregator. This way, the analysis is more private.

References

[1] Kevin Blighe and Aaron Lun. Pcatools: everything principal component analysis. https://bioconductor.org/packages/release/bioc/vignettes/PCAtools/inst/doc/PCAtools.html#a-loadings-plot, 2021.

[2] Scanpy Documentation. Preprocessing and clustering 3k pbmcs. https://scanpy-tutorials.readthedocs.io/en/latest/pbmc3k.html, 2021.

[3] Scanpy Documentation. Trajectory inference for hematopoiesis in mouse. https://scanpy-tutorials.readthedocs.io/en/latest/paga-paul15.html, 2021.

[4] Leland Mcinnes, John Healy, and James Melville. UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction arXiv : 1802 . 03426v2 [stat . ML] 6 Dec 2018. 2018.

[5] Franziska Paul, Ya'Ara Arkin, Amir Giladi, Diego Adhemar Jaitin, Ephraim Kenigsberg, Hadas Keren-Shaul, Deborah Winter, David Lara-Astiaso, Meital Gury, Assaf Weiner, Eyal David, Nadav Cohen, Felicia Kathrine Bratt Lauridsen, Simon Haas, Andreas Schlitzer, Alexander Mildner, Florent Ginhoux, Steffen Jung, Andreas Trumpp, Bo Torben Porse, Amos Tanay, and Ido Amit. Transcriptional Heterogeneity and Lineage Commitment in Myeloid Progenitors. *Cell*, 163(7):1663–1677, 2015.

[6] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):1–12, 2019.

[7] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy. Robust subspace learning: Robust pca, robust subspace tracking, and robust subspace recovery. *IEEE Signal Processing Magazine*, 35(4):32–55, 2018.

[8] F. Alexander Wolf, Fiona K. Hamey, Mireya Plass, Jordi Solana, Joakim S. Dahlin, Berthold Göttgens, Nikolaus Rajewsky, Lukas Simon, and Fabian J. Theis. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome Biology*, 20(1):1–9, 2019.

[9] Olga Zolotareva, Reza Nasirigerdeh, Julian Matschinske, Reihaneh Torkzadehmahani, Mohammad Bakhtiari, Tobias Frisch, Julian Späth, David B. Blumenthal, Amir Abbasinejad, Paolo Tieri, Georgios Kaissis, Daniel Rückert, Nina K. Wenke, Markus List, and Jan Baumbach. Flimma: a federated and privacy-aware tool for differential gene expression analysis. *Genome Biology*, 22(1):338, dec 2021.

Appendix C

Supplementary Material for Chapter 6: Federated K-Means – evaluation of initialization, clustering strategies, and k-selection

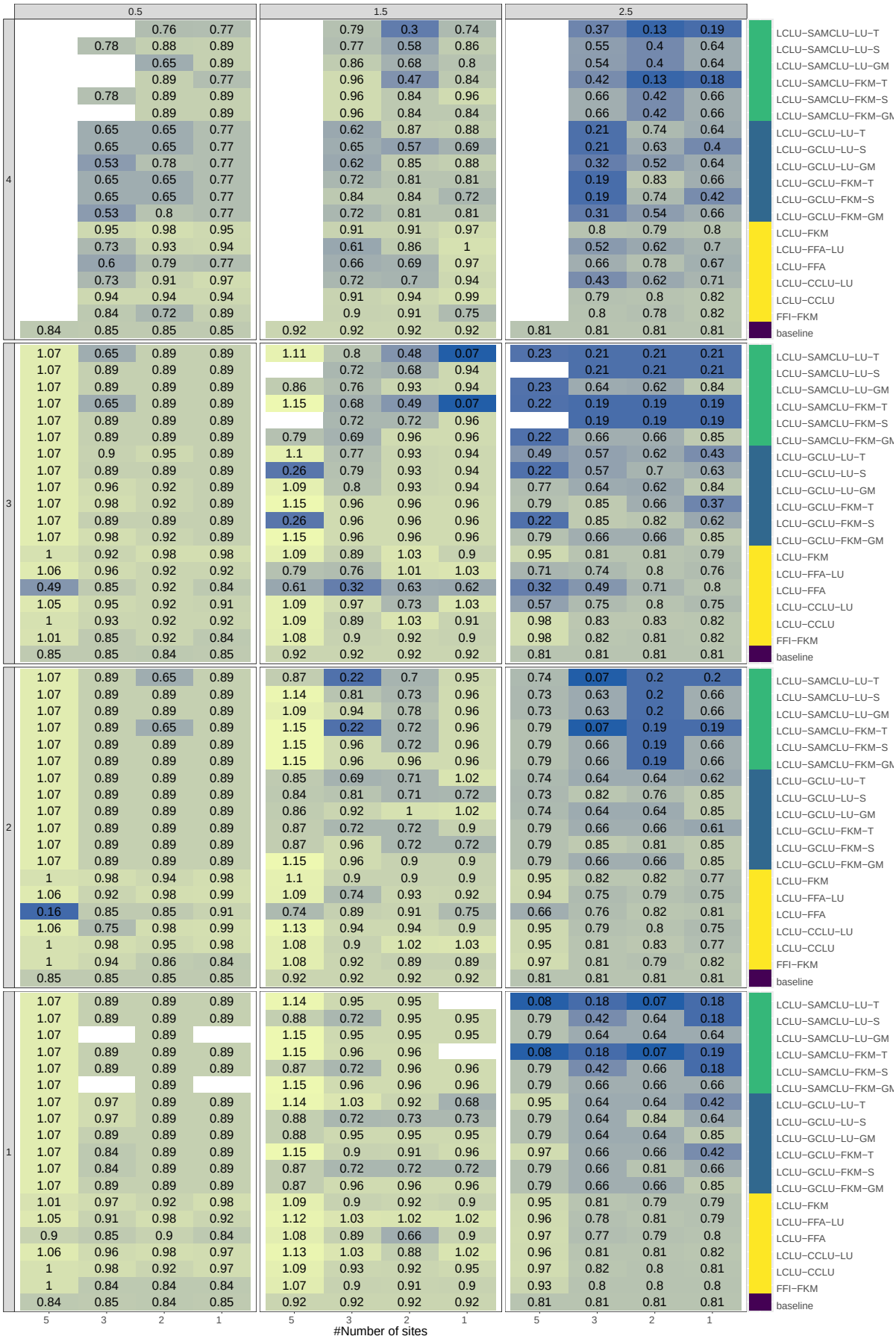

0.5
1.5
2.5
4
3
2
1
LCLU−SAMCLU−LU−T
LCLU−SAMCLU−LU−S
LCLU−SAMCLU−LU−GM
LCLU−SAMCLU−FKM−T
LCLU−SAMCLU−FKM−S
LCLU−SAMCLU−FKM−GM
LCLU−GCLU−LU−T
LCLU−GCLU−LU−S
LCLU−GCLU−LU−GM
LCLU−GCLU−FKM−T
LCLU−GCLU−FKM−S
LCLU−GCLU−FKM−GM
LCLU−FKM
LCLU−FFA−LU
LCLU−FFA
LCLU−CCLU−LU
LCLU−CCLU
FFI−FKM
baseline
5
3
2
1
#Number of sites

Figure C.1 – (Previous page) Evaluation results of K-Means using data containing 1% outliers using the grid evaluation strategy. The results are generally worse than on clean data without outliers. The figure is structured as follows. Each row of panels determines the number of clusters not available at a number of sites. The number of sites is indicated in the columns withing each panel. The "panel-columns" indicate the variance of the clusters in the data. The rows within each panel shows the algorithm configuration. The score indicated is the F1 score of the clustering normalized by the F1 score of baseline averaged over 5 different datasets of the same type. We observe the following: Edge cases lead to the failure of the runs due to the splintering of the data into many extremely small clusters. The sampling based strategies perform worse than the strategy which clusters the centroids. However, when using the GM selection strategy for k the methods perform the same. Extreme cases where each site receives the entire cluster lead to better performance of the clustering (first column per panel).

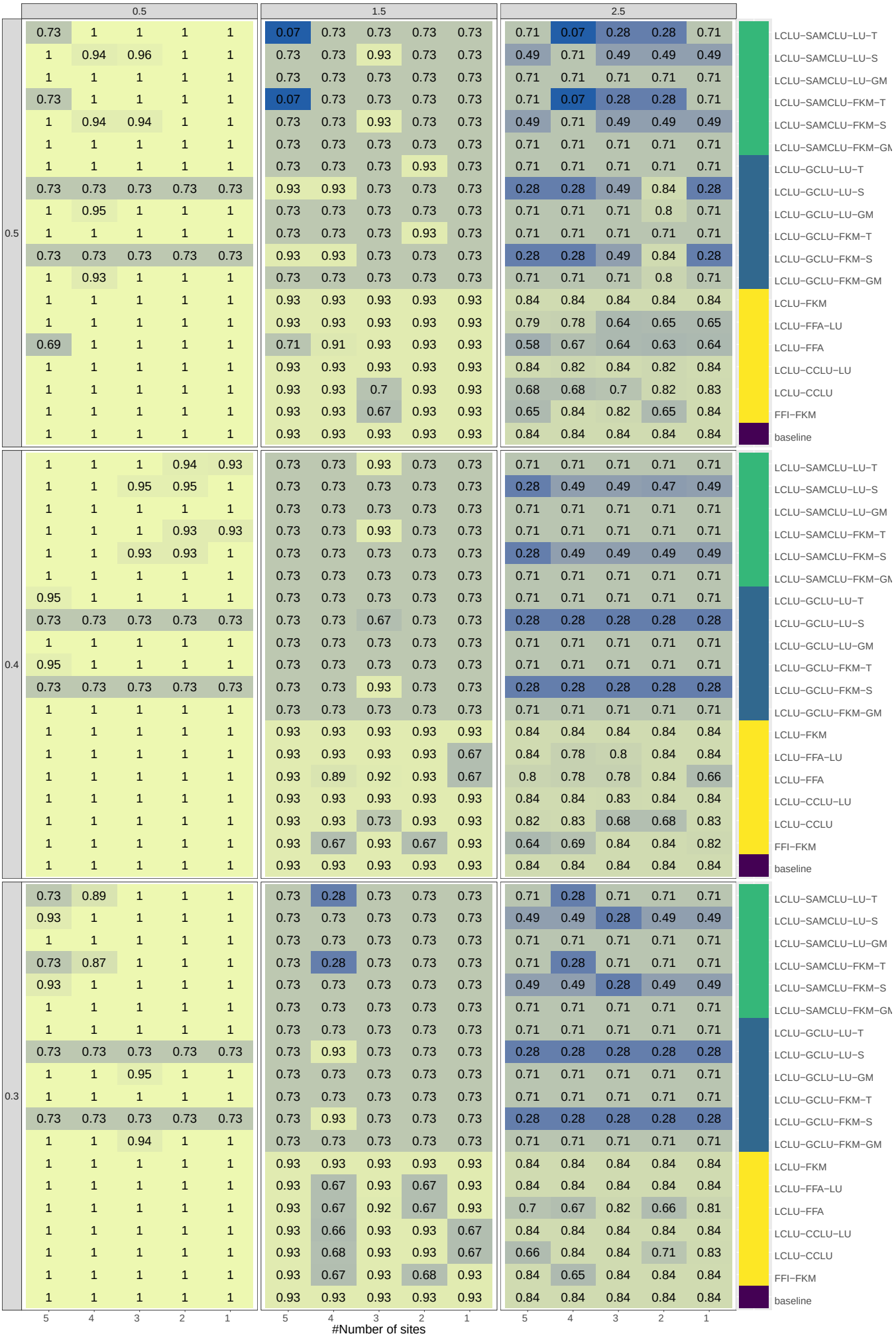

0.5
1.5
2.5
0.5
0.4
0.3
5 4 3 2 1
#Number of sites
LCLU−SAMCLU−LU−T
LCLU−SAMCLU−LU−S
LCLU−SAMCLU−LU−GM
LCLU−SAMCLU−FKM−T
LCLU−SAMCLU−FKM−S
LCLU−SAMCLU−FKM−GM
LCLU−GCLU−LU−T
LCLU−GCLU−LU−S
LCLU−GCLU−LU−GM
LCLU−GCLU−FKM−T
LCLU−GCLU−FKM−S
LCLU−GCLU−FKM−GM
LCLU−FKM
LCLU−FFA−LU
LCLU−FFA
LCLU−CCLU−LU
LCLU−CCLU
FFI−FKM
baseline

Figure C.2 – (Previous page) Evaluation of the clustering using the Stair evaluation strategy on data without outliers. The figure is stuctured similarly as above, the difference is that the panel-rows now contain the fraction of points which is attributed to the large site. For example in the first row the largest site for each non-iid cluster obtains 50% of the points. We observe the same global trend as previously: the worse the cluster separation, the worse the performance. The silhouette selection strategy does not perform well on this kind of data, most likely because there are sites which receive very few data points so that the silhouette coefficient cannot be accurately determined. Not all schemes with known k obtain scores as good as the baseline.

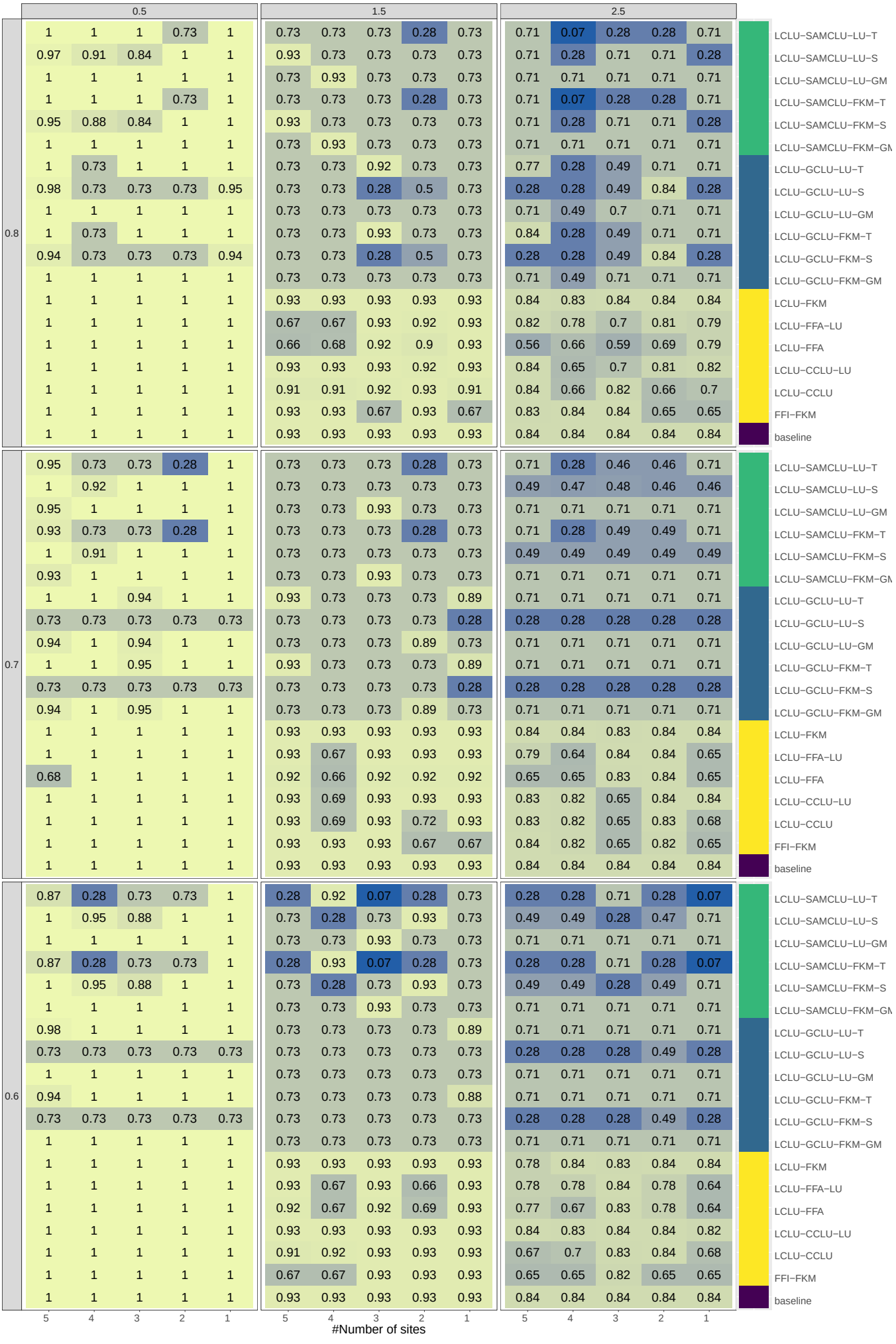

0.5
1.5
2.5
0.8
0.7
0.6
LCLU−SAMCLU−LU−T
LCLU−SAMCLU−LU−S
LCLU−SAMCLU−LU−GM
LCLU−SAMCLU−FKM−T
LCLU−SAMCLU−FKM−S
LCLU−SAMCLU−FKM−GM
LCLU−GCLU−LU−T
LCLU−GCLU−LU−S
LCLU−GCLU−LU−GM
LCLU−GCLU−FKM−T
LCLU−GCLU−FKM−S
LCLU−GCLU−FKM−GM
LCLU−FKM
LCLU−FFA−LU
LCLU−FFA
LCLU−CCLU−LU
LCLU−CCLU
FFI−FKM
baseline
5 4 3 2 1
#Number of sites

Figure C.3 – (Previous page)Evaluation of the clustering using the High-low evaluation strategy on data without outliers. In this figure the panel rows contains the fraction of points that go to the larges sites, e.g. with 0.8 and 2 large sites, each large site obtains 40% of the points and the rest is distributed to the small sites. Again, the silhouette coeffcient performs badly. We observe the same global trend as previously: the worse the cluster separation, the worse the performance. Not all schemes with known k obtain scores as good as the baseline.

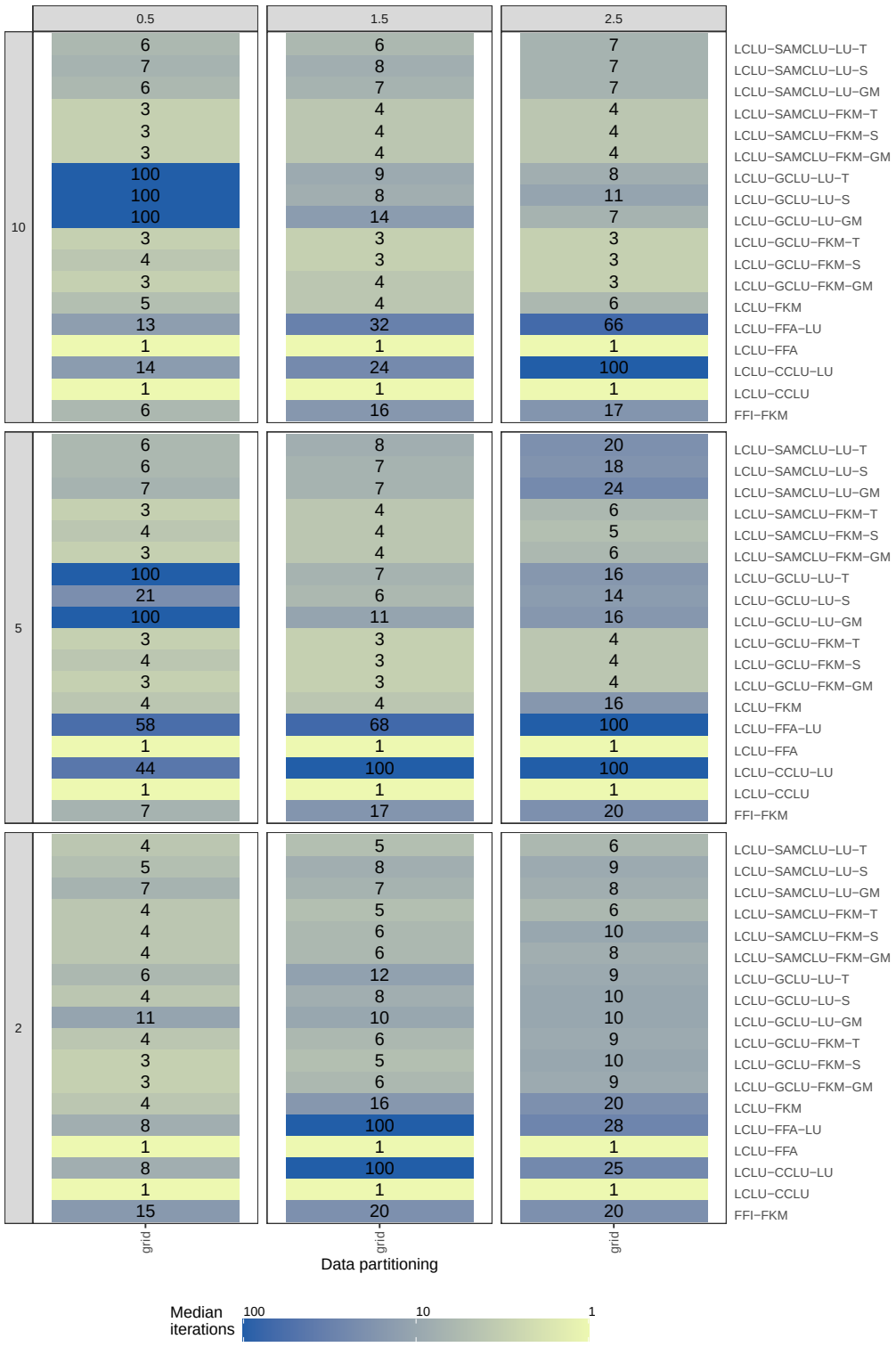


Figure C.4 – Iterations until convergence using the grid evaluation strategy with 1% outliers in the data

List of Figures

List of Tables