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.

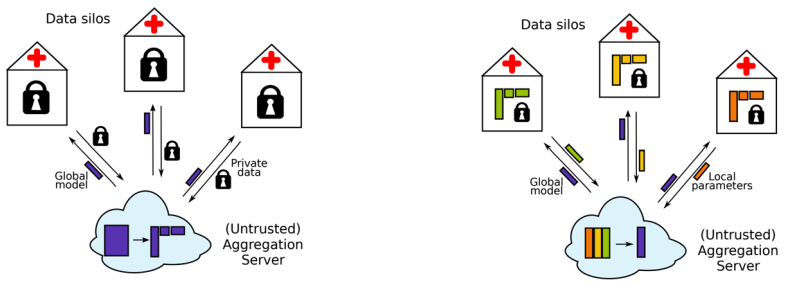

Fig. 1 Schematic comparison of traditional cloud base approaches (left) and federated learning (right).

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 lth 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).

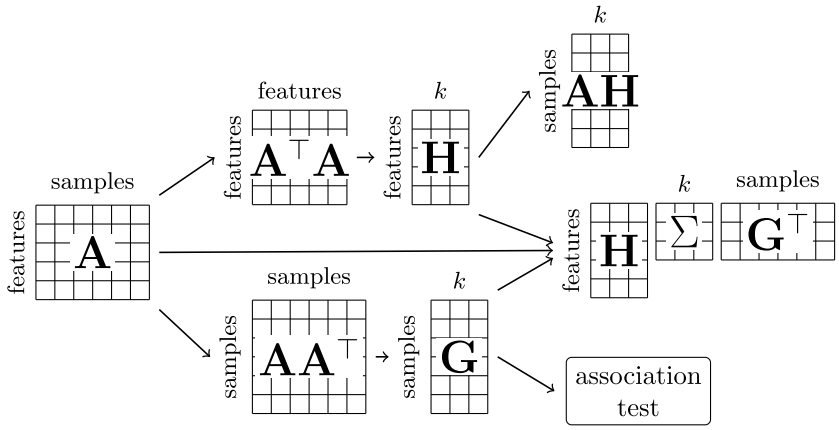

Fig. 2 Regular PCA for dimensionality reduction (top); GWAS PCA for sample stratification (bottom); and SVD (middle).

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 ith

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 $\quad \mathbf{U}^s \leftarrow [\mathbf{u}_1^s \ldots \mathbf{u}_s^k]$;
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 ith columns vector to the samples available at side s, and $\mathbf{u}_i^s$ be the restriction of the ith 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), $\mathbf{G}_0^s$ 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 $\mathbf{A}$.

for $s \in [S]$ **do**
  // Retrieve top $k \cdot c$ eigenvectors.
  $\mathbf{H}_s, \mathbf{\Sigma}_s, \mathbf{G}_s \leftarrow$ `singular-value-decomposition`$(\mathbf{A}_s, c \cdot k)$;
// Aggregate local subspaces to obtain approximate covariance matrix $\hat{\mathbf{M}}$ with $\dim(\hat{\mathbf{M}}) = c \cdot k \cdot S \times m$.
$\hat{\mathbf{M}} \leftarrow$ `stack-vertically`$([\mathbf{H}_s^\top])$;
// Use final dimensionality k
$\hat{\mathbf{H}} \leftarrow$ `singular-value-decomposition`$(\hat{\mathbf{M}}, k)$;
for $s \in [S]$ **do**
  $\hat{\mathbf{G}}_s \leftarrow \mathbf{A}_s^\top \hat{\mathbf{H}}$;
// Return approximate singular vector matrices of $\mathbf{A}$
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, the covariance matrix of the proxy data matrix is computed as $\hat{\mathbf{A}}_s \hat{\mathbf{A}}_s^\top \in \mathbb{R}^{k \cdot I' \times k \cdot I'}$ at the clients. The clients send this covariance to the aggregator which aggregates the covariance matrices by element wise addition $M_{\hat{\mathbf{A}}} = \sum_s \hat{\mathbf{A}}_s \hat{\mathbf{A}}_s^\top$, computes the eigenvectors $\mathbf{G}_a$ and shares them with the clients. The eigenvectors $\mathbf{G}_{\hat{\mathbf{A}}} \in \mathbb{R}^{k \cdot I' \times k}$ do not reflect properties of the original data but they can be used to recompute $\mathbf{G}$ as $\mathbf{G} = \hat{\mathbf{A}}_s^\top \mathbf{G}_{\hat{\mathbf{A}}} \in \mathbb{R}^{n \times k}$. $\mathbf{G}$ needs to be normalized using the federated orthonormalization subroutine (algorithm 3). 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 13 to 15).

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, the original covariance matrix is not disclosed exactly. However, as the final left eigenvectors are a common result of the analysis, the covariance matrix can still be approximated closely.

Algorithm 5: Federated randomized SVD (RANDOMIZED)

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.

1 $[\mathbf{H}_1, ..., \mathbf{H}_{I'}] \leftarrow$ `federated-subspace-iteration`$([\mathbf{A}_s], k, I')$;

2 $\mathbf{P} \leftarrow$ `stack-vertically`$([\mathbf{H}_1^\top, ..., \mathbf{H}_{I'}^\top])$;

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

4 $\quad \hat{\mathbf{A}}_s \leftarrow \mathbf{P}\mathbf{A}_s$;

5 $\quad \mathbf{M}^s_{\hat{\mathbf{A}}} = \hat{\mathbf{A}}_s \hat{\mathbf{A}}_s^\top$

6 $\mathbf{M}_{\hat{\mathbf{A}}} = \sum_s \hat{\mathbf{A}}_s \hat{\mathbf{A}}_s^\top$;

7 $\mathbf{G}_{\hat{\mathbf{A}}} \leftarrow$ `singular-value-decomposition`$(\mathbf{M}_{\hat{\mathbf{A}}}, k)$;

8 // Reconstruct the partial right singular vector

9 $\mathbf{G}_s \leftarrow \hat{\mathbf{A}}_s^\top \mathbf{G}_{\hat{\mathbf{A}}}$;

10 // Reorthogonalize the partial right singular vector

11 $\mathbf{G}_s \leftarrow$ `federated-gram-schmidt`$([\hat{\mathbf{G}}_s], k, I)$;

12 // Compute the full left singular vectors of A

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

14 $\mathbf{H} \leftarrow \sum_{s=1}^{S} \mathbf{H}_s$;

15 $\mathbf{H} \leftarrow$ `orthonormalize`$(\mathbf{H})$;

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

16 **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 the 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 $\mathcal{O}((S \cdot (I' + 1) \cdot k \cdot m) + (k \cdot I')^2)$ and $\mathcal{N} = \mathcal{O}((I' + 2) \cdot S)$.*
- *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 $\mathbf{M}_{\hat{\mathbf{A}}}$ has transmission complexity $(k \cdot I')^2$. An additional communication of the final $\mathbf{H}$ is required which costs $S \cdot k \cdot m$. Thereby, the total complexity is $\mathcal{D} = \mathcal{O}(I' \cdot S \cdot k \cdot m + (I' \cdot k)^2 + S \cdot k \cdot m) = \mathcal{O}((S \cdot (I' + 1) \cdot k \cdot m) + (k \cdot I')^2)$. 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 and can be executed in a constant number of communication rounds. This is especially important in the intended GWAS setting, since the number of samples and features may ber very large (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 (RANDOMIZED). 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	RI-FULL/FED-GS	$\mathcal{O}(I \cdot (S \cdot k \cdot m + k^2))$	$\mathcal{O}(I \cdot S \cdot k)$.
Algorithm 5	RANDOMIZED	$\mathcal{O}((S \cdot (I' + 1) \cdot k \cdot m) + (k \cdot I')^2)$	$\mathcal{O}((I' + 2) \cdot S)$.
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.

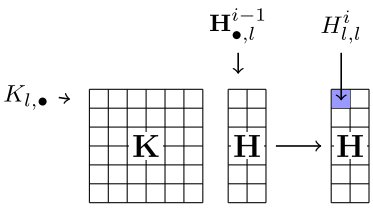

Fig. 3 Eigenvector update using the feature-by-feature covariance matrix.

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:

$$\mathbf{K}_{l,\bullet}\left[\mathbf{H}^0_{\bullet,1} \cdots \mathbf{H}^{m-1}_{\bullet,1}\right] = \left[H^1_{l,1} \cdots H^m_{l,1}\right]$$

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:

$$\mathbf{K}_{l,\bullet}\overbrace{\left[\mathbf{H}^0_{\bullet,1} \cdots \mathbf{H}^0_{\bullet,k} \cdots \mathbf{H}^{\frac{m}{k}-1}_{\bullet,1} \cdots \mathbf{H}^{\frac{m}{k}-1}_{\bullet,k}\right]}^{\mathbf{A}} = \overbrace{\left[H^1_{l,1} \cdots H^1_{l,k} \cdots H^{\frac{m}{k}}_{l,1} \cdots H^{\frac{m}{k}}_{l,k}\right]}^{\mathbf{b}}$$

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), and federated randomized subspace iteration (RANDOMIZED). 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).

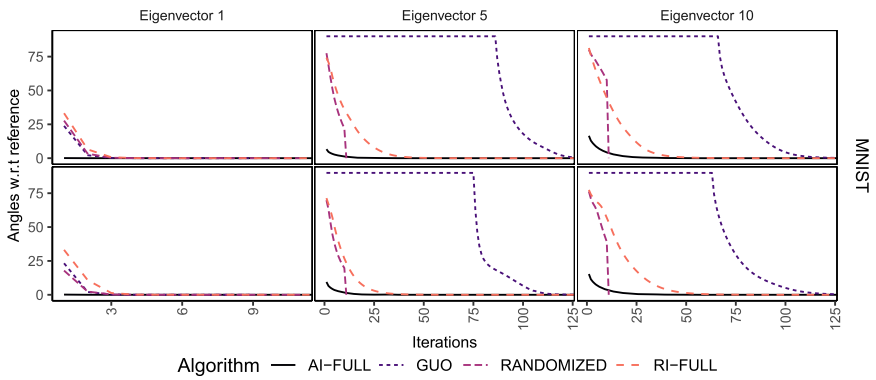

Fig. 4 Angles between selected reference eigenvectors and the federated eigenvectors for the MNIST data. The omitted eigenvectors show similar behaviors.

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.

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://github.com/AnneHartebrodt/federated-svd. 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-svd. We also created an AIME report (Matschinske et al, 2021a) to promote accessibility of machine learning research (Hartbrodt, 2022).

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, RANDOMIZED 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, RANDOMIZED 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 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 algorithm RANDOMIZED, followed by GUO and with a significant distance AI-FULL and RI-FULL. RANDOMIZED also spends 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 RANDOMIZED, followed with large gap 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, RANDOMIZED performs 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

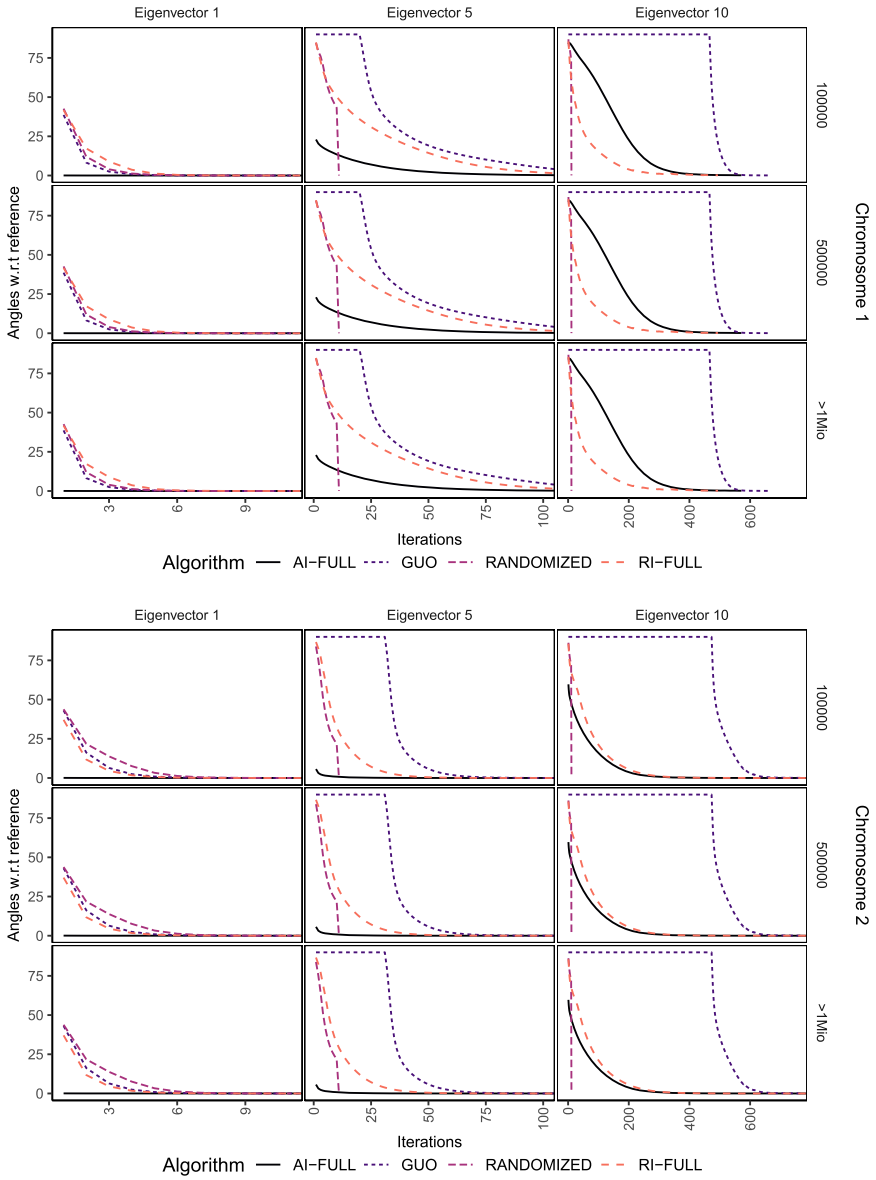

Fig. 5 Angles between selected reference eigenvectors and the federated eigenvectors on chromosome 1 and 2. The omitted eigenvectors show similar behaviors.

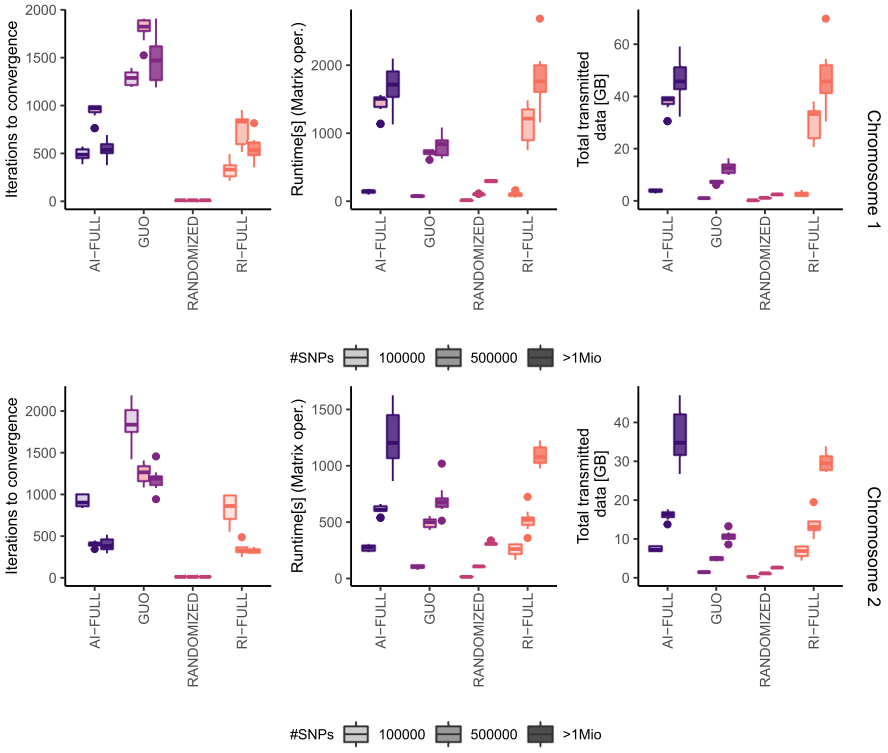

Fig. 6 Iterations, Runtime for matrix computations, and total transmitted data for each algorithm and each of the three data sets. The boxplots are grouped, the shading indicates the number of features ($0.1 * 10^6$, $0.5 * 10^6$, and roughly $1 * 10^6$).

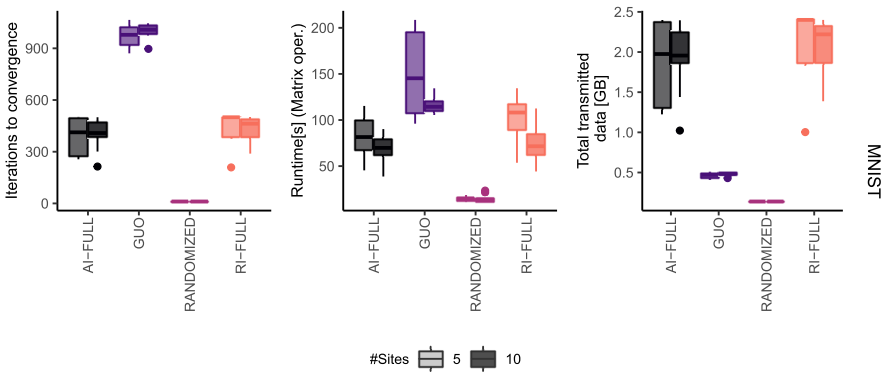

Fig. 7 Iterations, Runtime for matrix computations, and total transmitted data for each algorithm for the MNIST data. The boxplots are grouped, the shading indicates the number of simulated clients (5, and 10).

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.

Funding

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.

Competing interests

The authors have no relevant financial or non-financial interests to disclose.

References

Balcan MF, Kanchanapally V, Liang Y, et al (2014) Improved distributed principal component analysis. Advances in Neural Information Processing Systems 4(January):3113–3121. URL http://arxiv.org/abs/1408.5823, https://arxiv.org/abs/arXiv:1408.5823

Balcan MF, Du SS, Wang Y, et al (2016) An improved gap-dependency analysis of the noisy power method. Journal of Machine Learning Research 49(June):284–309. URL http://arxiv.org/abs/1602.07046, https://arxiv.org/abs/arXiv:1602.07046

Chen X, Lee JD, Li H, et al (2020) Distributed estimation for principal component analysis: a gap-free approach. CoRR abs/2004.02336. https://arxiv.org/abs/arXiv:2004.02336

Cho H, Wu DJ, Berger B (2018) Secure genome-wide association analysis using multiparty computation. Nature Biotechnology 36(6):547–551. https://doi.org/10.1038/nbt.4108

Cramer R, Damgård IB, et al (2015) Secure multiparty computation. Cambridge University Press

Dua D, Graff C (2017) UCI machine learning repository. URL http://archive.ics.uci.edu/ml

Galinsky KJ, Bhatia G, Loh PR, et al (2016) Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. The American Journal of Human Genetics 98(3):456–472. https://doi.org/10.1016/j.ajhg.2015.12.022

Gauch HG, Qian S, Piepho HP, et al (2019) Consequences of PCA graphs, SNP codings, and PCA variants for elucidating population structure. PLoS ONE 14(6):1–26. https://doi.org/10.1371/journal.pone.0218306

Grammenos A, Mendoza Smith R, Crowcroft J, et al (2020) Federated principal component analysis. In: Larochelle H, Ranzato M, Hadsell R, et al (eds) Advances in Neural Information Processing Systems, vol 33. Curran Associates, Inc., pp 6453–6464, URL https://proceedings.neurips.cc/paper/2020/file/47a658229eb2368a99f1d032c8848542-Paper.pdf

Guo YF, Lin X, Teng Z, et al (2012) A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data. Pattern Recognition 45(3):1211–1219. https://doi.org/10.1016/j.patcog.2011.09.002

Hadri B, Ltaief H, Agullo E, et al (2010) Tile qr factorization with parallel panel processing for multicore architectures. In: 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), pp 1–10, https://doi.org/10.1109/IPDPS.2010.5470443

Halko N, Martinsson PG, Tropp JA (2011) Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review 53(2):217–288. https://doi.org/10.1137/090771806

Hartbrodt A (2022) Federated singular value decomposition for high dimensional data [aime lp0kqt]. URL https://aime.report/lP0kqT

Hartebrodt A, Nasirigerdeh R, Blumenthal DB, et al (2021) Federated Principal Component Analysis for Genome-Wide Association Studies. ICDM 2021

Hoemmen M (2011) A communication-avoiding, hybrid-parallel, rank-revealing orthogonalization method. In: 2011 IEEE International Parallel Distributed Processing Symposium, pp 966–977, https://doi.org/10.1109/IPDPS.2011.93

Imtiaz H, Sarwate AD (2018) Differentially private distributed principal component analysis. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, p 2206–2210, https://doi.org/10.1109/ICASSP.2018.8462519

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

Kairouz P, McMahan HB, Avent B, et al (2021) Advances and open problems in federated learning. Foundations and Trends in Machine Learning 14(1-2):1–210. https://doi.org/10.1561/2200000083, https://arxiv.org/abs/arXiv:1912.04977

Kargupta H, Huang W, Sivakumar K, et al (2001) Distributed Clustering Using Collective Principal Component Analysis. Knowledge and Information Systems https://doi.org/10.4324/9781315799476-12

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

Lei Q, Zhong K, Dhillon IS (2016) Coordinate-wise power method. In: Lee D, Sugiyama M, Luxburg U, et al (eds) Advances in Neural Information Processing Systems, vol 29. Curran Associates, Inc., p 2064–2072, URL https://proceedings.neurips.cc/paper/2016/file/8b4066554730ddfaa0266346bdc1b202-Paper.pdf

Li Y, Byun J, Cai G, et al (2016) FastPop: A rapid principal component derived method to infer intercontinental ancestry using genetic data. BMC Bioinformatics 17(1):1–8. https://doi.org/10.1186/s12859-016-0965-1

Liu Y, Chen C, Zheng L, et al (2020) Privacy Preserving PCA for Multiparty Modeling. arXiv URL http://arxiv.org/abs/2002.02091, https://arxiv.org/abs/2002.02091

Londin ER, Keller MA, Maista C, et al (2010) Coaims: A cost-effective panel of ancestry informative markers for determining continental origins. PLoS One 5:e13,443. https://doi.org/10.1371/journal.pone.0013443

Matschinske J, Alcaraz N, Benis A, et al (2021a) The AIMe registry for artificial intelligence in biomedical research. Nature Methods https://doi.org/10.1038/s41592-021-01241-0, URL https://doi.org/10.1038/s41592-021-01241-0

Matschinske J, Späth J, Nasirigerdeh R, et al (2021b) The FeatureCloud AI store for federated learning in biomedicine and beyond. 2105.05734

Mothukuri V, Parizi RM, Pouriyeh S, et al (2021) A survey on security and privacy of federated learning. Future Generation Computer Systems 115:619–640. https://doi.org/10.1016/j.future.2020.10.007

Nasirigerdeh R, Torkzadehmahani R, Matschinske J, et al (2020) splink: A federated, privacy-preserving tool as a robust alternative to meta-analysis in genome-wide association studies. bioRxiv https://doi.org/10.1101/2020.06.05.136382

Nasirigerdeh R, Torkzadehmahani R, Baumbach J, et al (2021) On the privacy of federated pipelines. In: SIGIR 2021. ACM, New York, p 5, https://doi.org/10.1145/3404835.3462996

Price AL, Patterson NJ, Plenge RM, et al (2006) Principal components analysis corrects for stratification in genome-wide association studies. Nature Genetics 38(8):904–909. https://doi.org/10.1038/ng1847

Qi H, Wang TW, Birdwell JD (2003) Global principal component analysis for dimensionality reduction in distributed data mining. In: Statistical Data

Mining and Knowledge Discovery. Chapman and Hall/CRC, p 323–338, https://doi.org/10.1201/9780203497159.ch19

Rodríguez MÁ, Fernández A, Peregrín A, et al (2017) A Review of Distributed Data Models for Learning. Springer International Publishing, Cham

Saad Y (2011) Numerical Methods for Large Eigenvalue Problems. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, https://doi.org/10.1137/1.9781611970739

Sanchez-Fernandez A, Fuente M, Sainz-Palmero G (2015) Fault detection in wastewater treatment plants using distributed pca methods. In: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA). IEEE, p 1–7, https://doi.org/10.1109/ETFA.2015.7301504

Sluciak O, Straková H, Rupp M, et al (2016) Distributed Gram-Schmidt orthogonalization with simultaneous elements refinement. Eurasip Journal on Advances in Signal Processing 2016(1):1–13. https://doi.org/10.1186/s13634-016-0322-6

Steed A, Fradinho Duarte de Oliveira M (2010) More than two. Network Graphics pp 125–168. https://doi.org/10.1016/B978-0-12-374423-4.00004-5

Straková H, Gansterer WN, Zemen T (2012) Distributed qr factorization based on randomized algorithms. In: Wyrzykowski R, Dongarra J, Karczewski K, et al (eds) Parallel Processing and Applied Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 235–244

Tam V, Patel N, Turcotte M, et al (2019) Benefits and limitations of genome-wide association studies. Nature Reviews Genetics 20(8):467–484. https://doi.org/10.1038/s41576-019-0127-1

The 1000 Genomes Consortium, Auton, A. (2015) A global reference for human genetic variation. Nature 526(7571):68–74. https://doi.org/10.1038/nature15393

Visscher PM, Wray NR, Zhang Q, et al (2017) 10 Years of GWAS Discovery: Biology, Function, and Translation. American Journal of Human Genetics 101(1):5–22. https://doi.org/10.1016/j.ajhg.2017.06.005

Wu SX, Wai HT, Li L, et al (2018) A Review of Distributed Algorithms for Principal Component Analysis. Proceedings of the IEEE 106(8):1321–1340. https://doi.org/10.1109/JPROC.2018.2846568