Federated Principal Component Analysis for Genome-Wide Association Studies

Anne Hartebrodt
University of Southern Denmark
Department of Mathematics and Computer Science
Odense, Denmark
hartebrodt@imada.sdu.dk

Reza Nasirigerdeh
Technical University of Munich
Chair of Experimental Bioinformatics
Freising, Germany
reza.nasirigerdeh@tum.de

David B. Blumenthal
Technical University of Munich
Chair of Experimental Bioinformatics
Freising, Germany
david.blumenthal@wzw.tum.de

Richard Roettger
University of Southern Denmark
Department of Mathematics and Computer Science
Odense, Denmark
roettger@imada.sdu.dk

ABSTRACT

PVLDB Reference Format:
Anne Hartebrodt, Reza Nasirigerdeh, David B. Blumenthal, and Richard Roettger. Federated Principal Component Analysis for Genome-Wide Association Studies. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Availability Tag:
The source code of this research paper has been made publicly available at http://vldb.org/pvldb/format_vol14.html.

1 INTRODUCTION

Federated learning (FL) has recently gained attraction as a privacy preserving alternative to centralised computation. Instead of consolidating the data on a central server, the data holder keep ownership of their data and send only parameters to an aggregation server [21]. 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 [37, 40]. Genetic data is extremely sensitive in its nature and data holders hence cannot make it publicly available. The practical feasibility of using FL for GWAS has been demonstrated recently [22]. Alternative privacy preserving techniques such as secure multi-party computation that have been used for GWAS are computationally very expensive and hence do not scale to large cohorts [7].

Since GWAS are often done on populations of mixed ancestry, cryptic population confounders should be controlled for when 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 principle component analysis (PCA), and including these eigenvectors as confounding variables to models used for the association tests [9, 23].

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

For federated GWAS, a PCA algorithm for vertically partitioned data is required for computing the eigenvectors. Although a few such algorithms are available [12, 16, 26, 41], none of them is suitable for federated GWAS. More precisely, the algorithms reviewed in [41] use client-to-client communication and are therefore unsuitable for the star-like FL architectures used in GWAS. The algorithms presented in [16] and [26] rely on estimating a proxy covariance matrix and hence do not scale to large GWAS datasets, which often contain genetic variation data for more than 300 000 individuals. To the best of our knowledge, the only covariance free PCA algorithm suitable for a star-like architecture has been presented in [12]. However, this algorithm broadcasts the complete first $k-1$ 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 eigenvectors 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 [13], a state-of-the-art centralized, covariance-free PCA algorithm. Moreover, we show in a large-scale empirical evaluation the eigenvectors computed by our approach converge to centrally computed eigenvectors after sufficiently many iterations.

Although FL is a promising concept for privacy preserving computation, it has been shown that open questions regarding privacy leaks during the entire learning process remain [21]. One possible mitigation are hybrid approaches, where the model parameters are encrypted before sending them to the aggregator [21, 39]. Following this paradigm, we suggest a symmetric homomorphic encryption

(HE) scheme for our algorithm [1, 31], which protects against data leakage at the aggregator. We empirically show that, although HE comes at a considerable computational cost, the overall runtime of our encrypted algorithm is still perfectly reasonable for application settings such as GWAS where data collection can take years.

Finally, we present Fever-PCA — a user-friendly web-service which implements our algorithm and hence makes it available to non-computer scientist researchers working in the GWAS field. Fever-PCA is available at http://federated.compbio.sdu.dk/. To the best of our knowledge, Fever-PCA 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 [4]. 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 HE scheme for our algorithm which protects against data leakage at the aggregator.
- We present Fever-PCA, a user-friendly 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 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 algorithm. In Section 5, we present Fever-PCA. In Section 6, we report the results of the experiments. Section 7 concludes the paper and points out to future work.

2 PRELIMINARIES

Federated Learning and Employed Data Model. Unlike in centralized machine learning where the data is consolidated at a central sever and a model is calculated on the combined data, in FL the data remains at the data owners machine. Instead of sending the data, only model parameters are sent to the central server which combines the local models into a global model. See Figure 1 for a schematic comparison of centralized (cloud) learning and federated learning. In the cloud based approach, data contributors send their data to a central server and thereby loose agency over what happens to it. The global model is computed on all the aggregated data. In FL, 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 sent back to the local sites. No raw data is exchanged in federated leaning.

Typically, a star-like client-aggregator architecture is used in biomedical federated solutions [22, 35], 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*.

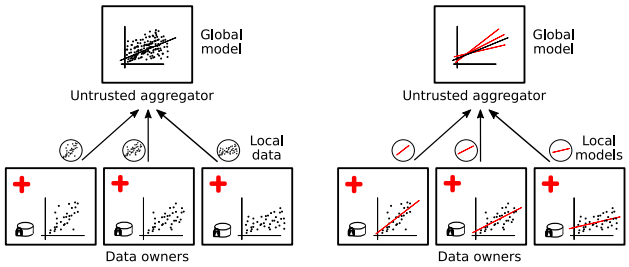

Figure 1: Schematic comparison of traditional cloud base approaches (left) and federated learning (right).

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) [27, 41]. 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. We explain this in detail in the following paragraphs.

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 [15]. 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.

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* [37].

Genome wide association studies try to identify SNPs that are linked to a specific phenotype [37, 40]. 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 sequences and tested for association with the trait of interest. Typically, simple models such as linear or logistic regression are used for this [22, 40]. 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 vector. 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.

The standard software for GWAS is the command line tool PLINK [4, 24]. Users choose their input files and run their analysis choosing the correct parameters. While a certain familiarity with the software is required, no actual programming is necessary.

Principal Component Analysis for Genome-Wide Association Studies. GWAS deal with possibly large cohorts of individuals which might come from several sub-populations with different ancestry or cryptic relatedness. These factors can confound the outcome of an association test and create false hits if not properly controlled for [37]. PCA has emerged as a popular strategy to infer population substructure, but is reported as lacking for decentralized learning [7]. More precisely, PCA is used to compute 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 $\mathbf{A}^\top\mathbf{A}$. Subsequently, these eigenvectors are included into the association test as covariates [9, 23]:

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

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. Federated PCA algorithms that are suitable for GWAS should hence 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 studies is usually not performed on the full set of SNPs. There seems to be no general consensus on how many SNPs should be used in population stratification. Gauch and colleagues identify 125 population studies using PCA [10]. Some of these PCA based stratification methods rely on a small ancestry informative markers [19], while others employ over 100 000 SNPs [5, 9, 28].

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

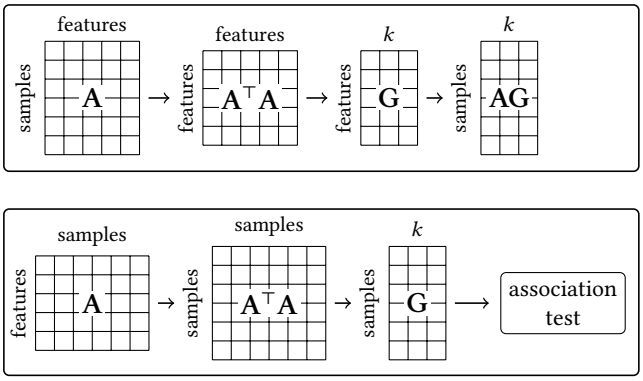

Figure 2: Regular PCA for dimensionality reduction (upper panel); GWAS PCA for sample stratification (lower panel).

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

It is clear that $\mathbf{A}^\top\mathbf{A}$ cannot be computed directly without sharing patient level data. Moreover, with growing number of samples, this matrix can become very large and computing it hence becomes infeasible. For instance, the UK Biobank — a large cohort frequently used for GWAS — contains GWAS data for around 300 000 individuals. Furthermore, approximating $\mathbf{A}^\top\mathbf{A}$ matrix would introduce 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.

Gram-Schmidt Orthonormalization. The Gram-Schmidt algorithm allows to transform a set of linearly independent vectors into a set of mutually orthogonal vectors, see [3] 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 the residuals are defined as $r_{i,j} = \mathbf{u}_j^\top \mathbf{v}_i / n_j$ 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 [12].

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 $s \in [S]$
$\mathbf{G}_i \in \mathbb{R}^{n \times k}$	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 $\text{span}(\mathbf{U}) = \text{span}(\mathbf{V})$

3 RELATED WORK

Centralized, Iterative, Covariance-Free Principal Component Analysis. While classical PCA algorithms rely on computing the covariance matrix $\mathbf{A}^\top\mathbf{A}$ [references missing], several covariance-free approaches exist which iteratively approximate the top k eigenvalues and eigenvectors [29]. These schemes avoid computing the covariance matrix by using a two-step approach where the candidate eigenvector is first multiplied by the transpose of the data matrix and then again by the data matrix, thereby achieving the same outcome as if iterating with the covariance matrix.

Algorithm 1: Centralized Vertical Subspace Iteration [13]

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 $\mathbf{A}^\top\mathbf{A}$.

// Initialize candidate eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
1 generate $\mathbf{G}_0 \in \mathbb{R}^{n \times k}$ randomly;
2 $\mathbf{G}_0 \leftarrow$ `orthonormalize`$(\mathbf{G}_0)$;
// Initialize iteration counter.
3 $i \leftarrow 1$;
4 **while** *termination criterion not met* **do**
  // Update candidate eigenvector matrix of $\mathbf{A}\mathbf{A}^\top$.
5 $\mathbf{H}_i = \mathbf{A}\mathbf{G}_{i-1}$;
6 $\mathbf{H}_i =$ `orthonormalize`$(\mathbf{H}_i)$;
  // Update candidate eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
7 $\mathbf{G}_i = \mathbf{A}^\top\mathbf{H}_i$;
8 $\mathbf{G}_i \leftarrow$ `orthonormalize`$(\mathbf{G}_i)$;
  // Increment iteration counter.
9 $i \leftarrow i + 1$;
// Return converged eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
10 $\mathbf{G} \leftarrow \mathbf{G}_i$;
11 **return** $\mathbf{G}$;

Algorithm 1 summarizes the centralized, iterative, covariance-free PCA algorithm suggested in [13], 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 5). Once a suitably defined termination criterion is met (eigenvectors converged, maximal number of iterations reached, time limit reached, etc.), the last candidate eigenvectors are returned (lines 10 to 11).

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 not only be used for sample stratification but also for feature reduction. Our federated version will inherit this property.

Federated Principal Component Analysis for Vertically Partitioned Data. A few algorithms to perform federated computation of PCA on vertically partitioned static data sets have been proposed [12, 16, 26, 41]. However, none of them is suitable for the GWAS use-case considered in this paper: The algorithms reviewed in [41] 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 [16] and [26] rely on estimating a proxy covariance matrix and hence do not meet Constraint 2 introduced above. Unlike these approaches, the algorithm described in [12] is covariance-free and suitable for the intended star-like architecture. However, it broadcasts the eigenvectors with the all sites and hence does not fulfill the requirements of Constraint 1.

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, Givens rotation and Householder reflection are popular approaches [references missing]. 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 [33, 34, 36]. However, these procedures 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.

Federated Principal Component Analysis for Horizontally Partitioned Data. Although not directly relevant for this work, 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 send to the aggregator [**?**]. 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 this algorithm, the solution quality is not independent of the number of transmitted dimensions. Furthermore, iterative schemes been proposed , where the eigenvectors are computed locally, sent to the aggregator, where an aggregation step is performed to obtain a new candidate subspace which is then sent back to the clients. The candidate subspace is refined iteratively [2, 6, 14]. Furthermore, a number of specific schemes for streaming [11], and other applications [30] exist. These concepts rely on the fact that at least an approximation of the entire eigenvector is possible at the clients or a global covariance matrix is approximated. As we have discussed earlier these assumptions do not hold true for vertically partitioned data.

4 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 is a federated version of centralized the vertical subspace iteration algorithm [13], which we have summarized in Algorithm 1 above. In Section 4.1, we describe our algorithm and prove that it is equivalent to centralized vertical subspace iteration. In Section 4.2, 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 4.3, we analyze the network transmission costs of the proposed algorithms.

4.1 Federated Vertical Subspace Iteration

Algorithm 2 and Algorithm 3 describe our federated vertical subspace iteration algorithm from, respectively, the client and the aggregator view: At the beginning of the algorithm, the first partial candidate eigenvector matrices $\mathbf{G}_0^s$ of $\mathbf{A}^\top\mathbf{A}$ are initialized randomly and orthonormalized (lines 1 to 6 in Algorithm 2 and lines 1 to 9 in Algorithm 3). Note that no privacy issues arise if centralized orthonormalization is chosen here, since the $\mathbf{G}_0^s$ matrices contain only random values. However, using federated orthonormalization as described in Section 4.2 can be beneficial to reduce network transmission costs (see Section 4.3 for details).

Inside the main loop, the candidate eigenvectors $\mathbf{H}_i$ of $\mathbf{A}\mathbf{A}^\top$ are updated and orthonormalized at the aggregator (lines 9 to 11 in Algorithm 2 and lines 12 to 15 in Algorithm 3). Next, the clients update the partial candidate eigenvectors $\mathbf{G}_i^s$ of $\mathbf{A}^\top\mathbf{A}$ (line 12 in Algorithm 2). Now, the candidate eigenvectors $\mathbf{G}_i$ of $\mathbf{A}^\top\mathbf{A}$ need to be orthonormalized. If centralized orthonormalization is chosen, the clients send $\mathbf{G}_i^s$ back to the aggregator, which carries out the orthonormalization (lines 14 to 15 in Algorithm 2 and lines 17 to 21 in Algorithm 3). For federated GWAS, this should be avoided, since the aggregator might use $\mathbf{G}_i$ to approximately reconstruct sensitive patient data. Consequently, we have developed a federated Gram-Schmidt orthnormalization algorithm (presented in Section 4.2), which can be called by the clients and the aggregator to ensure that the partial candidate eigenvectors $\mathbf{G}_i^s$ remain at the local sites (line 17 in Algorithm 2 and line 23 in Algorithm 3).

Like the original centralized version described in Algorithm 1 above, our algorithm can be run with various termination criteria.

Algorithm 2: Federated Vertical Subspace Iteration – Client

Input: Partial data matrix $\mathbf{A}^s \in \mathbb{R}^{m\times n^s}$ at site 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 site s.

// Initialize partial candidate eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
1 generate $\mathbf{G}_0^s \in \mathbb{R}^{n^s\times k}$ randomly;
2 **if** *use centralized orthonormalization* **then**
3 send-to-aggregator($\mathbf{G}_i^0$);
4 $\mathbf{G}_i^0 \leftarrow$ get-from-aggregator();
5 **else**
  // Use approach described in Algorithm 4 and Algorithm 5.
6 $\mathbf{G}_i^0 \leftarrow$ federated-gram-schmidt($\mathbf{G}_i^0$);
// Initialize iteration counter.
7 $i \leftarrow 1$;
8 **while** *termination criterion not met* **do**
  // Update partial candidate eigenvector matrix of $\mathbf{A}\mathbf{A}^\top$.
9 $\mathbf{H}_i^s \leftarrow \mathbf{A}^s\mathbf{G}_{i-1}^s$;
10 send-to-aggregator($\mathbf{H}_i^s$);
11 $\mathbf{H}_i \leftarrow$ get-from-aggregator();
  // Update partial candidate eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
12 $\mathbf{G}_i^s \leftarrow \mathbf{A}^{s\top}\mathbf{H}_i$;
13 **if** *use centralized orthonormalization* **then**
14 send-to-aggregator($\mathbf{G}_i^s$);
15 $\mathbf{G}_i^s \leftarrow$ get-from-aggregator();
16 **else**
    // Use approach described in Algorithm 4 and Algorithm 5.
17 $\mathbf{G}_i^s \leftarrow$ federated-gram-schmidt($\mathbf{G}_i^s$);
  // Increment iteration counter.
18 $i \leftarrow i + 1$;
// Return converged partial eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
19 $\mathbf{G}^s \leftarrow \mathbf{G}_i^s$;
20 **return** $\mathbf{G}^s$;

In our implementation, we use the convergence criterion

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

using the angle as a global measure as suggested in [18], 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. Other convergence criteria could be used as drop-in replacements.

We now prove that our federated algorithm is equivalent to the centralized version described in Algorithm 1. Consequently, 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 [13].

Proposition 1. *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 $\tilde{\mathbf{H}}_i^s$

Algorithm 3: Federated Vertical Subspace Iteration – Aggregator

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 $s \in [S]$.

// Initialize candidate eigenvector matrix of $\mathbf{A}^\top\mathbf{A}$.
1 generate $\mathbf{G}_0 \in \mathbb{R}^{n \times k}$ randomly;
2 **if** *use centralized orthonormalization* **then**
3 **for** $s \in [S]$ **do** $\mathbf{G}_0^s \leftarrow$ `get-from-client`(s);
4 $\mathbf{G}_0 \leftarrow$ `stack-vertically`$(\mathbf{G}_0^1, \ldots, \mathbf{G}_0^S)$;
5 $\mathbf{G}_0 \leftarrow$ `orthonormalize`$(\mathbf{G}_i)$;
6 $\mathbf{G}_0^1, \ldots, \mathbf{G}_0^S \leftarrow$ `split-vertically`$(\mathbf{G}_0)$;
7 **for** $s \in [S]$ **do** `send-to-client`$(\mathbf{G}_0^s, s)$;
8 **else**
  // Use approach described in Algorithm 4 and Algorithm 5.
9 `federated-gram-schmidt`();
// Initialize iteration counter.
10 $i \leftarrow 1$;
11 **while** *termination criterion not met* **do**
  // Update partial candidate eigenvector matrices of $\mathbf{A}\mathbf{A}^\top$.
12 **for** $s \in [S]$ **do** $\mathbf{H}_i^s \leftarrow$ `get-from-client`(s);
13 $\mathbf{H}_i \leftarrow \sum_{s=1}^S \mathbf{H}_i^s$;
14 $\mathbf{H}_i \leftarrow$ `orthonormalize`$(\mathbf{H}_i)$;
15 **for** $s \in [S]$ **do** `send-to-client`$(\mathbf{H}_i, s)$;
  // Update partial candidate eigenvector matrices of $\mathbf{A}^\top\mathbf{A}$.
16 **if** *use centralized orthonormalization* **then**
17 **for** $s \in [S]$ **do** $\mathbf{G}_i^s \leftarrow$ `get-from-client`(s);
18 $\mathbf{G}_i \leftarrow$ `stack-vertically`$(\mathbf{G}_i^1, \ldots, \mathbf{G}_i^S)$;
19 $\mathbf{G}_i \leftarrow$ `orthonormalize`$(\mathbf{G}_i)$;
20 $\mathbf{G}_i^1, \ldots, \mathbf{G}_i^S \leftarrow$ `split-vertically`$(\mathbf{G}_i)$;
21 **for** $s \in [S]$ **do** `send-to-client`$(\mathbf{G}_i^s, s)$;
22 **else**
  // Use approach described in Algorithm 4 and Algorithm 5.
23 `federated-gram-schmidt`();
  // Inrement iteration counter.
24 $i \leftarrow i + 1$;

be the (partial) eigenvector matrices maintained by our federated algorithm described in Algorithm 2 and Algorithm 3 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 14 of Algorithm 3, 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 12 of Algorithm 2 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. □

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 would be to send them to the aggregator which performs the aggregation and then sends the orthonormal matrices back to the clients (lines 14 to 15 in Algorithm 2 and lines 17 to 21 in Algorithm 3). However, in this naïve scheme, the transmission cost scales with the number of variables (samples in GWAS) and all eigenvectors are known to the aggregator.

To address these two problems, we suggest a federated Gram-Schmidt orthonormalization algorithm. Algorithm 4 and Algorithm 5 describe the algorithm from, respectively, the aggregator's and clients' perspectives. 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 $s \in [S]$.

Algorithm 4: Federated Gram-Schmidt – Client

Input: Data matrix $\mathbf{V}^s = [\mathbf{v}_1^s \ldots \mathbf{v}_k^s] \in \mathbb{R}^{r \times k}$ at site s.
Output: Orthonormalized data matrix $\mathbf{U}^s$ at site s.

// Compute squared norm of first orthogonal vector.
1 $\mathbf{u}_1^s \leftarrow \mathbf{v}_1^s$;
2 $n_1^s \leftarrow \mathbf{u}_1^{s\top}\mathbf{u}_1^s$;
3 `send-to-aggregator`(n_1^s);
4 $n_1 \leftarrow$ `get-from-aggregator`();
// Orthogonalize all subsequent vectors.
5 **for** $i \in [k] \setminus \{1\}$ **do**
  // Compute residuals for vector which should be orthogonalized.
6 **for** $j \in [i-1]$ **do**
7 $r_{ij}^s \leftarrow \mathbf{u}_j^{s\top}\mathbf{v}_i^s / n_j$;
8 `send-to-aggregator`$((r_{ij}^s)_{j=1}^{i-1})$;
9 $(r_{ij})_{j=1}^{i-1} \leftarrow$ `get-from-aggregator`();
  // Orthogonalize the vector and compute squared norm.
10 $\mathbf{u}_i^s \leftarrow \mathbf{v}_i^s - \sum_{j=1}^{i-1} r_{ij} \cdot \mathbf{u}_j^s$;
11 $n_i^s \leftarrow \mathbf{u}_i^{s\top}\mathbf{u}_i^s$;
12 `send-to-aggregator`(n_i^s);
13 $n_i \leftarrow$ `get-from-aggregator`();
// After orthogonalization, scale all k vectors to unit norm and return them.
14 **for** $i \in [k]$ **do**
15 $\mathbf{u}_i^s \leftarrow \frac{1}{\sqrt{n_i}} \cdot \mathbf{u}_i^s$
16 $\mathbf{U}^s \leftarrow [\mathbf{u}_1^s \ldots \mathbf{u}_s^k]$;
17 **return** $\mathbf{U}^s$;

In a first step, the squared norm of the first orthogonal vector is computed (lines 1 to 4 in Algorithm 4 and lines 1 to 3 in Algorithm 5). Subsequently, the remaining $k-1$ vectors are orthogonalized. For

Algorithm 5: Federated Gram-Schmidt – Aggregator

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

// Compute squared norm of first orthogonal vector.
1 **for** $s \in [S]$ **do** $n_1^s \leftarrow$ `get-from-client`(s);
2 $n_1 \leftarrow \sum_{s=1}^{S} n_1^s$;
3 **for** $s \in [S]$ **do** `send-to-client`(n_1, s);
// Orthogonalize all subsequent vectors.
4 **for** $i \in [k] \setminus \{1\}$ **do**
  // Compute residuals for vector which should be orthogonalized.
5 **for** $s \in [S]$ **do** $(r_{ij}^s)_{j=1}^{i-1} \leftarrow$ `get-from-client`(s);
6 **for** $j \in [i-1]$ **do**
7 $r_{ij} \leftarrow \sum_{s=1}^{S} r_{ij}^s$;
8 **for** $s \in [S]$ **do** `send-to-client`$((r_{ij})_{j=1}^{i-1}, s)$;
  // Compute squared norm of orthogonalized vector.
9 **for** $s \in [S]$ **do** $n_i^s \leftarrow$ `get-from-client`$()$;
10 $n_i \leftarrow \sum_{s=1}^{S} n_1^s$;
11 **for** $s \in [S]$ **do** `send-to-client`(n_i, s);

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 5 to 9 in Algorithm 4 and lines 5 to 8 in Algorithm 5). Next, $\mathbf{v}_i$ is orthogonalized at the clients (line 10 in Algorithm 4) and the squared norm of the resulting orthogonal vector $\mathbf{u}_i$ is computed (lines 11 to 13 in Algorithm 4 and lines 9 to 11 in Algorithm 5). After orthogonalization, all orthogonal vectors are scaled to unit norm at the clients (lines 14 to 15 in Algorithm 4).

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 side 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 identities $r_{ij} = \tilde{r}_{ij}$ established before and the inductive assumption. 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 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 centralized orthonormalization is used, it holds that $\mathcal{D} = O(I \cdot S \cdot k \cdot (m+n))$ and $\mathcal{N} = O(I \cdot S)$.*
- *If federated Gram-Schmidt orthonormalization is used, it holds that $\mathcal{D} = O(I \cdot (S \cdot k \cdot m + k^2))$ and $\mathcal{N} = O(I \cdot S \cdot k)$.*

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 $O(S \cdot k \cdot m)$ and $O(S)$, respectively. For orthonormalizing the eigenvector matrices $\mathbf{G}_i \in \mathbb{R}^{n \times k}$, we need to transmit a data volume of $O(S \cdot k \cdot m)$ in $O(S)$ rounds of communication, if centralized orthonormalization is used. If our federated Gram-Schmidt algorithm is employed, the transmitted data volume is reduced to $O(S \cdot k^2)$ but the number of communications increases to $O(S \cdot k)$. By summing over the iterations i, this implies 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 [19, 20]. 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 our federated Gram-Schmidt algorithm is preferable not only because it protects against data leakage at the aggregator, but also because it greatly improves the scalability of our federated vertical subspace iteration algorithm.

5 WEB-SERVICE

Many federated principal component algorithms exist on paper, but lack implementations. Therefore, we provide a demonstration software to show, that federated principal component analysis can be done in practice. This software can help familiarize researchers to the federated learning approach, without highly specialised computer setups and with limited programming knowledge. Researchers performing genome-wide association studies rely on ready made software such as plink [4, 24] to compute their results. In order to make federated GWAS available to these researchers community a user-friendly tool is required, otherwise it is unlikely to be adopted in practice.

We provide a software designed for a server-client architecture which allows several users to collaboratively compute a PCA. It consists of three components:

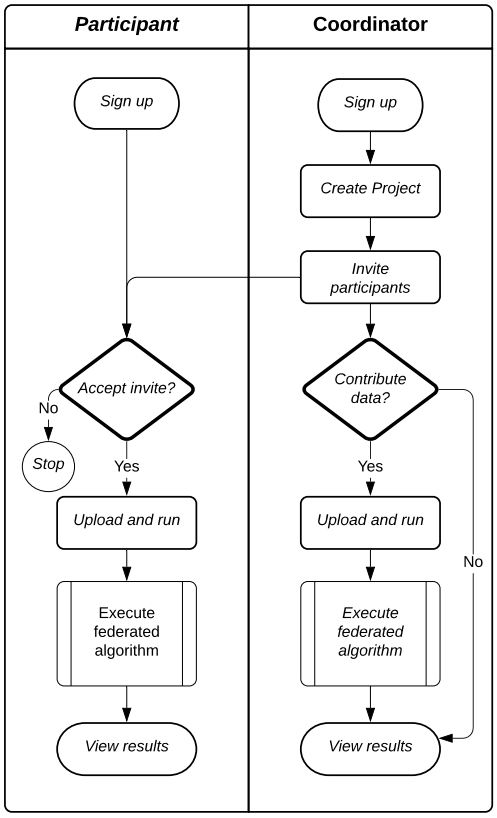

Figure 3: Schematic overview of federated study.

- a client, which runs on the data holder's computer. There the local parameters are computed and send to the server. The data is never shared to the server.
- An aggregation server which receives and aggregates the parameters received by the clients and sends them back to the client.
- A web interface which can be used to set up the study. This includes the generation of tokens for all participants to ensure only authenticated users participate in the study.

The following basic workflow is required to set up a federated study.

- Every user sets up an account using a web interface.
- The study coordinator (owner) sets up a study by selecting the algorithm and parameters and invites participants.
- Each participant securely receives their invitation and can accept or decline.
- All data contributing participants load their data into the client using a graphical user interface and trigger the run.
- At the end of the computation, the results are downloaded to the user's computers, such that everyone has the same shared result.

This tool targets the less technically inclined GWAS community and intends to demonstrate the feasibility of federated genome-wide association studies with population stratification. As it is experimental software under development it has not been technically optimised. Nevertheless, we were able to compute a federated principal component analysis in ... using a matrix of 2500 individuals with 100,000 features.

Security Enhancing Measures in Federated Learning. In the star-like architecture assumed here, a potentially untrusted party performs the aggregation step. This means that the aggregator sees all the intermediate results, and specifically gains more information about the intermediate parameters than the data holding parties. In order to increase the privacy of the proposed scheme, we consider a hybrid approach. A hybrid approach is a combination of Fl with additional privacy enhancing mechanisms, such as differential privacy (DP) [8], homomorphic encryption (HE) [1, 31] or secure multiparty computation [7] which used to hide the parameters from third parties, such as the aggregator.

Federated medical studies will still be required to follow all ethical guidelines. Therefore ad-hoc studies on federated data are unlikely. More concretely, the federated system, we have in mind is a setting where high-trust parties, such as hospitals and research institutes compute results together, using a study protocol they previously agreed on. Based on these assumptions on the system, we evaluate whether homomorphic encryption is a practical method to increase the privacy of the learning set up. If the parties are concerned about the aggregator learning from the transmitted parameters, we propose to use a basic symmetric HE scheme where all participating parties privately gain access to a shared key and encrypt their parameters under said key before sending them to the aggregator. The aggregation server does not have access to the common key and only blindly aggregates the data.

Two things are to be noted here. First, strictly speaking, a symmetric encryption library would be sufficient for this purpose, which would speed up to process up to a factor of 8 [31], but there is no library beyond prototype status yet. Secondly, naturally this scheme does only protect against the aggregator. A multiparty homomorphic encryption scheme would be better suited to protect everyone's parameters against all participants, but such libraries are likewise not readily available yet.

6 EMPIRICAL EVALUATION

In the following section, we will show the results of the experiments we conducted to demonstrate the accuracy and feasibility of federated PCA with vertical partitioning empirically. We evaluate the 4 scenarios, sequential iteration of the vectors (Guo) and concurrent iteration of the vectors with and without federated QR normalisation. See table 2 for notation. The use case in this article is PCA for Genome-Wide association studies, because the goal is not feature reduction, but sample stratification. Nevertheless, this algorithm can be applied to general data sets as well. Therefore we also demonstrate, the usefulness of this approach for general audiences using more widely used data sets.

6.1 Data

In this study, we use 3 publicly available data sets to measure the empirical performance of the algorithm, the MNIST handwritten

Table 2: Methods compared in the experiments.

Method	Paper	PCA	QR
SEQ-CENT	[12]	sequential	central
SEQ-FED	this paper	sequential	federated
SIM-CENT	this paper	simultaneous	central
SIM-FED	this paper	simultaneous	federated

digit repository, a larger gene expression data set from the cancer genome atlas and genetic data from the 1000 Genomes Project. Mnist [17] contains 60,000 gray scale images of handwritten numerals of 784 pixels. Every image is a sample, where the pixels are the features. We apply minimal preprocessing, which includes centering and scaling the data to unit variance, both frequently applied steps before PCA. Gene expression data comes in a tabular format, with the genes acting as the features on one axis and the sample on the other axis. Here we used data from the MMRF-COMMPASS study from TCGA with approximately 20,000 coding genes and 859 samples. We complete our data set selection with data from the 1000 genomes project [38] which contains 2502 individuals and several millions of SNP measurements. The input to the PCA is encoded as a tabular format where for each SNP an individual has either 0, 1 or 2 minor alleles. These values are then normalized according to the formulas given in [9]. We applied standard preprocessing steps (MAF filtering, LD pruning) and use a random sub sample of the remaining SNPs. As mentioned before, selecting a high number of SNPs might be prohibitive for the runtime, but methods requiring fewer SNPs are available. We emphasize that we do not intend to perform a 'realistic' GWAS study, which required careful preprocessing and might include additional steps. Our goal is to show, that the prinicpal component analysis is equivalent to the centralised PCA.

6.2 Vertically partitioned federated PCA

In order to assess the performance of the algorithms we split the data sets into differing numbers of vertical chunks to simulated clients which hold a subset of the features for all the samples. In the case of the 1000 Genomes data, we distributed the samples, meaning every client gets a sub set of the samples and holds measurements for every SNP. We then run the federated algorithm 20 times on the test data and report the average angle between the eigenvectors computed with the federated version of PCA and the centralised version of PCA.

The Eigenvectors converge to the real eigenvector to very high accuracy, meaning the angle between the federated and the real eigenvector tend to 0 as the number of iterations grows. The higher the eigenvector rank, the worse the convergence behaviour. This is normal, since the convergence depends on the eigenvalue gap between the two consecutive eigenvalues which becomes smaller with higher rank. Figure 6 and 7 show exemplary convergence behaviours for the first eigenvector of the SNP data set (chromosome 1), which behaves nicely and the 8th eigenvector of the mnist data set, which is the most difficult to retrieve due to the small Eigengap between eigenvalue 8 and 9. All algorithms behave similarly. The small differences are due to the random initialisation rather than one algorithm performing systematically better than another one. The number of sites does not impede the convergence properties, when splitting the data into equal chunks. Notably, using federated QR factorisation and not exchanging the eigenvectors comes at no loss of accuracy and equal convergence behaviour.

Figure 4 shows the performance of simulated federated PCA using data from human chromosome 1. Shown is the angle between the reference eigenvector computed using plink 2.0 (using excat PCA due to the small number of samples). The eigenvectors converge to the eigenvectors as computed by plink, essentially showing that the behaviour of this standard tool can be replicated using federated PCA.

6.3 Federated QR factorisation

To show the performance of the orthonormalisation, we randomly generate matrices of dimensions $50,000x20$, a realistic dimensions for PCA, and split them into horizontal chunks. We then run federated matrix orthonormalisation and compare the result to a matrix orthonormalized using a standard version (LAPACK interfaced via scipy). To show the feasibility of federated QR factorisation with aggregation under homomorphic encryption, we used a python interface to SEAL [25, 32], an open source asymmetric homomorphic encryption library. In a simulation we encrypt parameters that are sent to the server and the aggregation is done on the encrypted values. Before the client operations the values are decrypted again. The overhead due to the HE-scheme can be mainly attributed to the time required for the encryption. In our simulation the average time increases from a fraction of seconds to ca. 15 seconds without actual transmission between clients. Depending on the use case, this can be an acceptable increase in run time or a prohibitive overhead. The accuracy loss due to the encryption is negligible. The fact that the aggregation step only required addition, which can be done efficiently and at minimal accuracy loss, is beneficial here.

6.4 Scalability

One main bottleneck in federated learning is the amount of transmitted data. In this federation scheme, the H matrix, which needs to be transmitted no matter the chosen version of the algorithm, is dependent on the number of selected features. Thanks to the decentralized QR orthonormalisation, the federation scheme does not scale with the number of individuals in the study. Therefore, the goal of facilitation GWAS with higher sample sizes in a federated fashion can be achieved, as the main scaling factor for the transmission cost is the number of SNPs. We hope that reducing the number of SNPs used in the PCA is a trade-off investigators are willing to make in order to increase their sample size and cohort diversity by running federated GWAS. More generally, since the scheme allows to retrieve both left and right eigenvectors, it is also possible to retrieve the eigenvectors of the feature-by-feature matrix, where the amount of transmitted data does not scale with the number of samples.

- Figure convergence comparison
- scalability

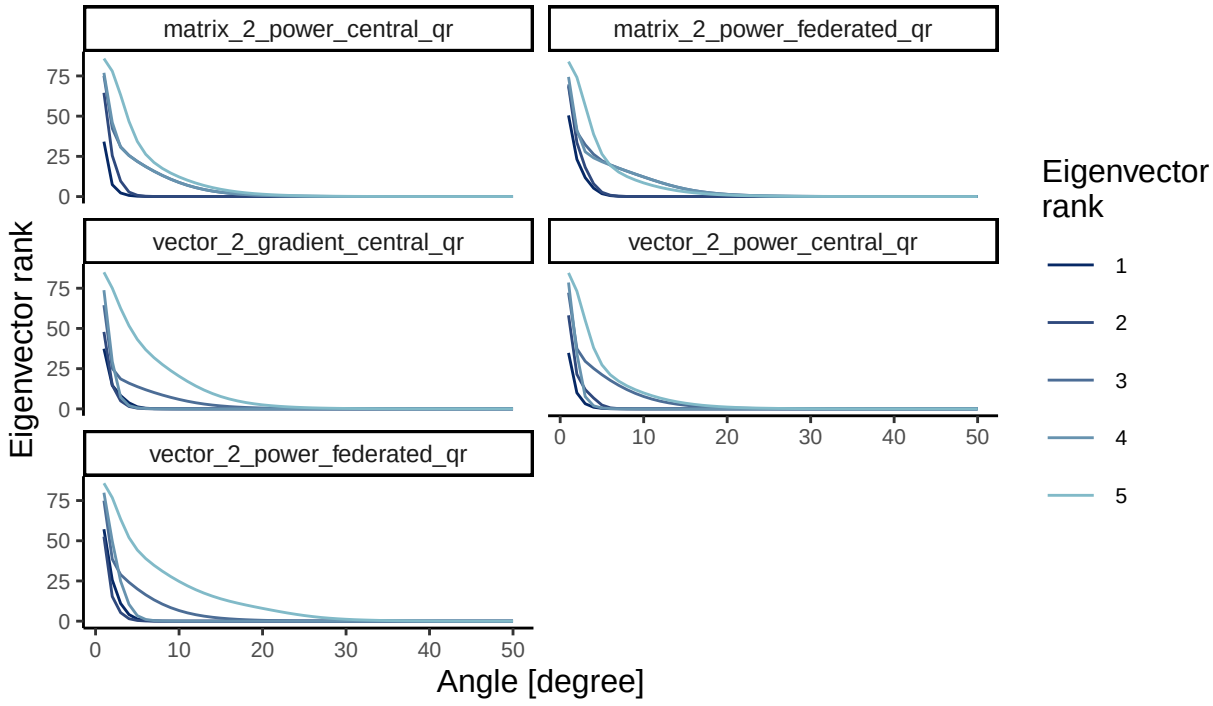

Figure 4: Angles between the eigenvectors computed by plink and the eigenvectors computed in a federated fashion.

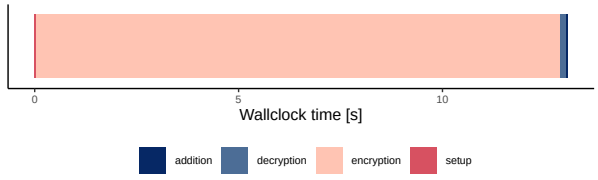

Figure 5: Compute overhead of encrypted federated QR procedure.

7 CONCLUSIONS AND OUTLOOK

We presented an improved federated principal component analysis which can for instance be used federated population stratification in GWAS. Unlike previous algorithms, the eigenvectors are not shared among the participants due to the use of fully federated QR orthonormalisation. This not only increases the scalabiliy of the proposed approach in terms of transmission cost, but also improves the privacy of the algorithm. We provide a user friendly tool to promote federated learning in less technically inclined communities. Future work will include to reduce the transmission cost further possibly by introducing concentrated updates instead of updating all values at every iteration.

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] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A Survey on Homomorphic Encryption Schemes. *Comput. Surveys* 51, 4 (2018), 1–35. https://doi.org/10.1145/3214303

[2] Maria-Florina Balcan. [n.d.]. An Improved Gap-Dependency Analysis of the Noisy Power Method. ([n. d.]), 26.

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

[4] Christopher C. Chang, Carson C. Chow, Laurent C.A.M. Tellier, Shashaank Vattikuti, Shaun M. Purcell, and James J. Lee. 2015. Second-generation PLINK: Rising to the challenge of larger and richer datasets. *GigaScience* 4, 1 (2015), 1–16. https://doi.org/10.1186/s13742-015-0047-8 arXiv:1410.4803

[5] Gyaneshwer Chaubey, Manvendra Singh, Niraj Rai, Mini Kariappa, Kamayani Singh, Ashish Singh, Deepankar Pratap Singh, Rakesh Tamang, Deepa Selvi Rani, Alla G. Reddy, and et al. 2016. Genetic affinities of the Jewish populations of India. *Scientific Reports* 6, November 2015 (2016), 1–10. https://doi.org/10.1038/srep19166

[6] Xi Chen, Jason D. Lee, He Li, and Yun Yang. 2020. Distributed Estimation for Principal Component Analysis: a Gap-free Approach. *arXiv:2004.02336 [cs, stat]* (Sep 2020). http://arxiv.org/abs/2004.02336 arXiv: 2004.02336.

[7] Hyunghoon Cho, David J. Wu, and Bonnie Berger. 2018. Secure genome-wide association analysis using multiparty computation. *Nature Biotechnology* 36, 6 (2018), 547–551. https://doi.org/10.1038/nbt.4108

[8] Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2013), 211–407. https://doi.org/10.1561/0400000042

[9] Kevin J. Galinsky, Gaurav Bhatia, Po-Ru Loh, Stoyan Georgiev, Sayan Mukherjee, Nick J. Patterson, and Alkes L. Price. 2016. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. *The American Journal of Human Genetics* 98, 3 (mar 2016), 456–472. https://doi.org/10.1016/j.ajhg.2015.12.022

[10] Hugh G. Gauch, Sheng Qian, Hans Peter Piepho, Linda Zhou, and Rui Chen. 2019. Consequences of PCA graphs, SNP codings, and PCA variants for elucidating

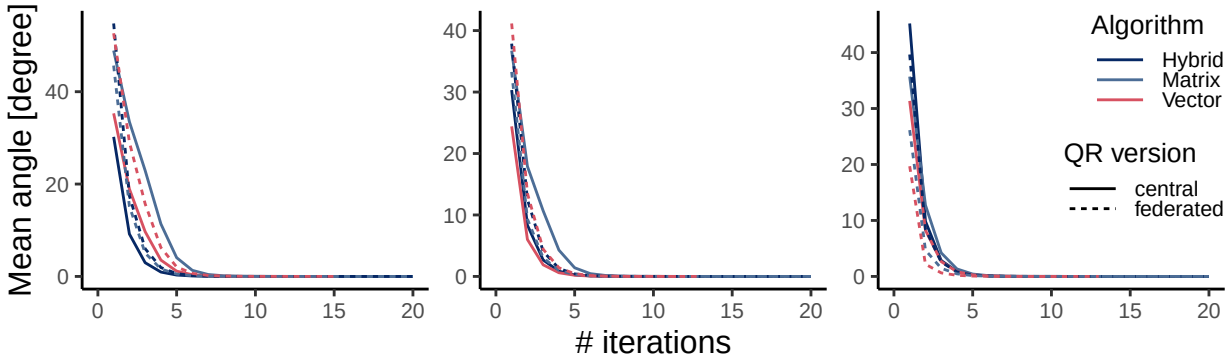

Figure 6: Angle between the reference and federated eigenvector of rank 1 run in function of the number of iterations. The randomized data has been split into 2, 5, or 10 data sets of equal size. Shown are the three proposed schemes, sequential iteration using a vector, a simultaneous iteration using a matrix and the hybrid scheme. Shown here: Eigendecomposition of GRM of SNPs on chromosome 1

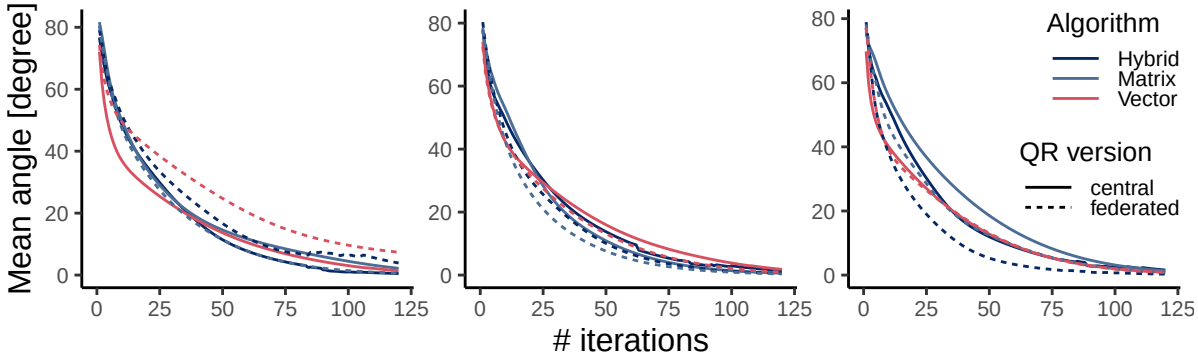

Figure 7: Angle between the reference and federated eigenvector of rank 8 run in function of the number of iterations. Data used: Mnist. The eigengap between the 8th and 9th eigenvalue is an order of magnitude smaller than the other eigengaps, therefor all algorithms show poor convergence behaviour.

population structure. *PLoS ONE* 14, 6 (2019), 1–26. https://doi.org/10.1371/journal.pone.0218306

[11] Andreas Grammenos, Rodrigo Mendoza-Smith, Jon Crowcroft, and Cecilia Mascolo. [n.d.]. Federated Principal Component Analysis. ([n. d.]), 12.

[12] Yue Fei Guo, Xiaodong Lin, Zhou Teng, Xiangyang Xue, and Jianping Fan. 2012. A covariance-free iterative algorithm for distributed principal component analysis on vertically partitioned data. *Pattern Recognition* 45, 3 (2012), 1211–1219. https://doi.org/10.1016/j.patcog.2011.09.002

[13] N. Halko, P. G. Martinsson, and J. A. Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.* 53, 2 (Jan 2011), 217–288. https://doi.org/10.1137/090771806

[14] Hafiz Imtiaz and Anand D. Sarwate. 2018. Differentially Private Distributed Principal Component Analysis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2206–2210. https://doi.org/10.1109/ICASSP.2018.8462519

[15] Ian T. Joliffe. 2002. *Principal Component Analysis*. Springer-Verlag. https://doi.org/10.1007/b98835

[16] Hillol Kargupta, Weiyun Huang, Krishnamoorthy Sivakumar, and Erik Johnson. 2001. Distributed Clustering Using Collective Principal Component Analysis. *Knowledge and Information Systems* (2001). https://doi.org/10.4324/9781315799476-12

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

[18] Qi Lei, Kai Zhong, and Inderjit S Dhillon. 2016. 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., 2064–2072. https://proceedings.neurips.cc/paper/2016/file/8b4066554730ddfaa0266346bdc1b202-Paper.pdf

[19] Yafang Li, Jinyoung Byun, Guoshuai Cai, Xiangjun Xiao, Younghun Han, Olivier Cornelis, James E. Dinulos, Joe Dennis, Douglas Easton, Ivan Gorlov, Michael F. Seldin, and Christopher I. Amos. 2016. FastPop: A rapid principal component derived method to infer intercontinental ancestry using genetic data. *BMC Bioinformatics* 17, 1 (2016), 1–8. https://doi.org/10.1186/s12859-016-0965-1

[20] Eric R. Londin, Margaret A. Keller, Cathleen Maista, Gretchen Smith, Laura A. Mamounas, Ran Zhang, Steven J. Madore, Katrina Gwinn, and Roderick A. Corriveau. 2010. CoAIMs: A Cost-Effective Panel of Ancestry Informative Markers for Determining Continental Origins. 5 (Oct 2010), e13443. https://doi.org/10.1371/journal.pone.0013443

[21] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (Feb 2021), 619–640. https://doi.org/10.1016/j.future.2020.10.007

[22] Reza Nasirigerdeh, Reihaneh Torkzadehmahani, Julian Matschinske, Tobias Frisch, Markus List, Julian Späth, Stefan Weiss, Uwe Völker, Nina Kerstin Wenke, Tim Kacprowski, and Jan Baumbach. 2020. sPLINK: A Federated, Privacy-Preserving Tool as a Robust Alternative to Meta-Analysis in Genome-Wide Association Studies. *bioRxiv* (2020). https://doi.org/10.1101/2020.06.05.136382

[23] Alkes L. Price, Nick J. Patterson, Robert M. Plenge, Michael E. Weinblatt, Nancy A. Shadick, and David Reich. 2006. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics* 38, 8 (2006), 904–909. https://doi.org/10.1038/ng1847

[24] Shaun Purcell and Christopher Chang. [n.d.]. Plink 2.0. www.cog-genomics.org/plink/2.0/. Accessed: October 2020.

[25] Pyfhel 2020. Pyfhel (PYthon For Homomorphic Encryption Libraries). https://github.com/ibarrond/Pyfhel. Ibarrond, Alberto.

[26] Hairong Qi, Tse Wei Wang, and J. Douglas Birdwell. 2003. Global principal component analysis for dimensionality reduction in distributed data mining. *Statistical Data Mining and Knowledge Discovery* (2003), 323–338. https://doi.org/10.1201/9780203497159.ch19

[27] Miguel Ángel Rodríguez, Alberto Fernández, Antonio Peregrín, and Francisco Herrera. 2017. *A Review of Distributed Data Models for Learning*. Springer International Publishing, Cham. 88–97 pages.

[28] Juan L. Rodriguez-Flores, Khalid Fakhro, Francisco Agosto-Perez, Monica D. Ramstetter, Leonardo Arbiza, Thomas L. Vincent, Amal Robay, Joel A. Malek, Karsten Suhre, Lotfi Chouchane, and et al. 2016. Indigenous Arabs are descendants of the earliest split from ancient Eurasian populations. *Genome Research* 26, 2 (2016), 151–162. https://doi.org/10.1101/gr.191478.115

[29] Yousef Saad. 2011. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970739

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

[31] Savvas Savvides, Darshika Khandelwal, and Patrick Eugster. 2020. Efficient confidentialitypreserving data analytics over symmetrically encrypted datasets. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1290–1303. https://doi.org/10.14778/3389133.3389144

[32] SEAL 2020. Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA.

[33] Ondrej Sluvciak, 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. https://doi.org/10.1186/s13634-016-0322-6

[34] Ondrej Sluciak, Hana Strakova, Markus Rupp, and Wilfried N. Gansterer. 2012. Distributed Gram-Schmidt orthogonalization based on dynamic consensus. *Conference Record - Asilomar Conference on Signals, Systems and Computers* (2012), 1207–1211. https://doi.org/10.1109/ACSSC.2012.6489213

[35] Anthony Steed and Manuel Fradinho Duarte de Oliveira. 2010. More than two. *Network Graphics* (12 2010), 125–168. https://doi.org/10.1016/B978-0-12-374423-4.00004-5

[36] Hana Strakov, Wilfried N Gansterer, and Thomas Zemen. 2012. Based on Randomized Algorithms. (2012), 235–244.

[37] Vivian Tam, Nikunj Patel, Michelle Turcotte, Yohan Bossé, Guillaume Paré, and David Meyre. 2019. Benefits and limitations of genome-wide association studies. *Nature Reviews Genetics* 20, 8 (2019), 467–484. https://doi.org/10.1038/s41576-019-0127-1

[38] The 1000 Genomes Project Consortium., Corresponding authors., Auton, A. et al. 2015. A global reference for human genetic variation. *Nature* 526, 7571 (2015), 68–74. https://doi.org/10.1038/nature15393

[39] Reihaneh Torkzadehmahani, Reza Nasirigerdeh, David B. Blumenthal, Tim Kacprowski, Markus List, Julian Matschinske, Julian Späth, Nina Kerstin Wenke, Béla Bihari, Tobias Frisch, Anne Hartebrodt, Anne-Christin Hausschild, Dominik Heider, Andreas Holzinger, Walter Hötzendorfer, Markus Kastelitz, Rudolf Mayer, Cristian Nogales, Anastasia Pustozerova, Richard Röttger, Harald H. H. W. Schmidt, Ameli Schwalber, Christof Tschohl, Andrea Wohner, and Jan Baumbach. 2020. Privacy-preserving Artificial Intelligence Techniques in Biomedicine. (2020). arXiv:2007.11621 http://arxiv.org/abs/2007.11621

[40] Peter M. Visscher, Naomi R. Wray, Qian Zhang, Pamela Sklar, Mark I. McCarthy, Matthew A. Brown, and Jian Yang. 2017. 10 Years of GWAS Discovery: Biology, Function, and Translation. *American Journal of Human Genetics* 101, 1 (2017), 5–22. https://doi.org/10.1016/j.ajhg.2017.06.005

[41] Sissi Xiaoxiao Wu, Hoi To Wai, Lin Li, and Anna Scaglione. 2018. A Review of Distributed Algorithms for Principal Component Analysis. *Proc. IEEE* 106, 8 (2018), 1321–1340. https://doi.org/10.1109/JPROC.2018.2846568