Additional Use Cases for the Federated SVD Algorithm

Terje Jacobsson
School of Computation, Information and Technology
The Technical University of Munich
Munich, Germany
terjjac@stud.ntnu.no

***Abstract*—Federated learning is a machine learning method based on training ML models on data which is split into several parts. The reasons behind using federated learning might be to train a model when the full data set is too big to be stored in one piece, or to get the statistical benefits of other organizations' data without compromising data privacy. The Federated Singular Value Decomposition proposed by Chai et al. is an example of such a federated learning algorithm. It modifies the original Singular Value Decomposition (SVD) to guarantee data privacy, while still reaping some of its benefits. In their paper, Chai et al. proposes three algorithms that use FedSVD as a subroutine, namely linear regression (LR), principal component analysis (PCA) and latent semantic analysis (LSA).**

The aim of this paper is to contribute to the work of Chai et al. by proposing some additional use cases for FedSVD: to use FedSVD for total least squares minimization, and to get more statistical knowledge of the data generating process by extracting the rank and information about the fundamental subspaces of the full data matrix, without compromising privacy.

I. Introduction

Konečný et al. [1] defines Federated Learning as a machine learning setting where the goal is to train a high-quality centralized model while the training data remains distributed over a large number of clients. The motivations for doing so may vary, but two well-known examples are that the data sets may be too large to be stored in one place, or that access to the data is restricted in order to preserve data privacy. As data protection laws become more prominent, actors like companies and governmental agencies might have to adapt to this by incorporating federation into widely used machine learning techniques. Federated learning might also be used when actors want to collaborate on training a machine learning model, without revealing corporate data directly, thus reaping the benefits of large data sets, without compromising privacy.

In the paper titled "Practical Lossless Federated Singular Vector Decomposition over Billion-Scale Data" by Chai et al., the Federated SVD algorithm is introduced [2]. This Federated SVD algorithm distributes randomly generated orthogonal matrices to each user, so that only they know how to mask and un-mask their own data. This is done my multiplying each user's data matrix with their respective masking matrices, which only are removable by multiplying the masked data matrix with the transpose of the masking matrices. For people well-versed in linear algebra, this is a quite an intuitive form of data security, since it is solely based on the characteristics of orthogonal matrices.

There are a few papers that are directly comparable with Chai et al. in terms of functionality. Two of them are Liu et al. [3] and Hartebrodt et al. [4], which both introduce a federated SVD algorithm. Liu et al. relies more on cryptography techniques than on linear algebra to achieve results with respect to the privacy preservation. Therefore, Liu et al. is considerably more complicated to both implement and understand than Chai et al., for someone that is not well-acquainted with cryptography. The algorithm proposed by Hartebrodt et al. is heavily reliant on linear algebra, but its derivations are quite complex, which complicates the process of implementing new use cases for this algorithm.

The algorithm proposed by Chai et al. is beautifully simple. It is easily and elegantly explained, and its working principles incorporates fundamental linear algebra principles in order to preserve data privacy, instead of relying on long derivations or complex cryptography techniques. Because of these reasons, this paper will look into new use cases for the algorithm proposed by Chai et al., which from now on will be referred to as FedSVD. Since Chai et al. proves that their algorithm is privacy preserving, this paper will treat their algorithm like a black box, and not change the inner workings of it, since then the privacy will also be preserved in the algorithms using FedSVD as a subroutine. There are some implications to this, which will be talked about in the section V, "A Note Regarding the Preservation of Data Privacy".

II. Notation

Throughout the paper, the following notation will be used:

- i: the index referencing the i'th user.
- k: the total number of users.
- A_i: the data of user i. Dimensions are specified in each use case.

- A: the total data matrix consisting of the data of all users i. Dimensions and partition schemes are specified in each use case.
- X_i: each user's input data to FedSVD. $X_i \in \mathbb{R}^{m \times n_i}$.
- X: the matrix composed of all users' input matrices X_i, so that $X = \begin{bmatrix} X_1 \dots X_i \dots X_k \end{bmatrix}$. $X \in \mathbb{R}^{m \times n}$, where $n = \sum_{i=1}^{k} n_i$.

Other use case specific notation is also introduced when needed. The notation of both X and A is introduced to differentiate between the data the users want to work with, and what is actually input to the FedSVD algorithm.

III. Mathematical Background

A. *The Singular Value Decomposition (SVD)*

The SVD might be the most useful numerical algorithm related to linear algebra. It is used in a large number of different numerical computations, and is an extremely good tool for extracting fundamental knowledge about a data matrix. A very crude and fast description of this algorithm is that it decomposes an arbitrary matrix A into the product between two orthogonal matrices U and V and the diagonal matrix Σ so that $A = U\Sigma V^\top$.

B. *The Federated Singular Value Decomposition (FedSVD) of Chai et al.*

The FedSVD, proposed by Chai et al., is a powerful tool for analyzing the statistical properties of the underlying data generating mechanisms of a data set, composed of data from several users, while preserving each user's privacy.

The following is a brief summary of how the FedSVD algorithm works, retrieved from Chai et al. [2]:

1) Each user defines some transformation of their data to be used as input argument, $X_i = f(A_i)$, e.g. $X_i = A_i$ or $X_i = {A_i}^\top$, where $X \in \mathbb{R}^{m \times n}$ always holds.
2) A Trusted Authority (TA) generates two removable random orthogonal masking matrices $P \in \mathbb{R}^{m \times m}$ and $Q \in \mathbb{R}^{n \times n}$.
3) The TA splits the Q matrix into k horizontal parts, $Q^\top = [{Q_1}^\top \dots {Q_i}^\top \dots {Q_K}^\top]$, and sends matrix Q_i to user i.
4) All users mask their own data matrix X_i, by computing ${X_i}' = PX_iQ_i$.
5) A Computation Service Provider (CSP) gets the full masked data matrix X' through the following aggregation of the masked submatrices: $X' = PXQ = P[X_1 \dots X_i \dots X_k][{Q_1}^\top \dots {Q_i}^\top \dots {Q_k}^\top]^\top = \sum_{i=1}^{k} PX_iQ_i$.
6) The CSP runs the standard SVD algorithm, which performs the following factorization: $X' = U'\Sigma{V'}^\top$.
7) The masks could be removed if the CSP would broadcast U' and V' to all users, since then U could be recovered by $U = P^\top U'$, and V_i by $V_i = {V'}^\top {Q_i}'^\top$. This could however compromise privacy, since then each user would get access to all the masked right singular vectors. Instead Q' and ${V_i}'$ are broadcast to each user i, and a more complex method is used to retrieve V_i. Details regarding this can be found in chapter "3.3 Removing the Masks" in [2]. It should be noted that the paper claims that ${V_i}^\top$ could be retrieved by performing ${V_i}'^\top = {Q_i}'^\top {V'}^\top$, however the matrix dimensions are not compatible here.
8) Σ is broadcast to all users.

Thus, all users get their own input data matrix decomposed like $X_i = U\Sigma {V_i}^\top$. Each user could of course perform the SVD on their own data matrix by performing $X_i = U_i\Sigma_i {V_i}^\top$ with $U \neq U_i$ and $\Sigma \neq \Sigma_i$. The reason for using FedSVD is that the matrices U and Σ contain more underlying information about the statistics related to the data generating process of the whole data matrix X, than U_i and Σ_i do, without compromising privacy.

The use cases presented in the next sections treat FedSVD as a black box, only caring about the input and output parameters. The internal parameter X is also emphasized due to its importance.

Important parameters of the FedSVD:

- **Input parameter:** data matrix $X_i = f(A_i)$
- **Important internal parameter:** complete data matrix $X = \begin{bmatrix} X_1 \dots X_i \dots X_k \end{bmatrix} = U\Sigma V^\top = U\Sigma \begin{bmatrix} {V_1}^\top \dots {V_i}^\top \dots {V_k}^\top \end{bmatrix}$
- **Output:** decomposed data matrix $X_i = U\Sigma {V_i}^\top$

IV. Use Cases for FedSVD

Chai et al. [2] have already implemented linear regression (LR), principal component analysis (PCA) and latent semantic analysis (LSA) for their algorithm. This section will present new use cases for FedSVD. The first new use case is how to use FedSVD for total least squares minimization, i.e. to find an x^* so that $x^* = \arg\min_x \|Ax\|_2$. The second new use case is how to use FedSVD to derive the rank, as well as information regarding the fundamental subspaces of the full data matrix A, when each user originally only has access to A_i.

A. *Federated Total Least Squares Minimization*

*1) **Algorithm Description**:* Assume that each user i has a data matrix $A_i \in \mathbb{R}^{n_i \times m}$. We assume that there are k participants, such that $n = \sum_{i=1}^{k} n_i$, which gives the full data matrix

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_i \\ \vdots \\ A_k \end{bmatrix} \in \mathbb{R}^{n \times m}. \tag{1}$$

The users want to collaborate, to find a solution x^* to the total least squares minimization problem defined by

$$x^* = \arg\min_x \|Ax\|_2, \tag{2}$$

without exposing any of their data to another user. We define the input to FedSVD as

$$X_i = A_i^\top. \tag{3}$$

This gives the following composition of each user's individual data matrix:

$$X = \begin{bmatrix} X_1 \dots X_i \dots X_k \end{bmatrix} = \begin{bmatrix} {A_1}^\top \dots {A_i}^\top \dots {A_k}^\top \end{bmatrix} = A^\top, \tag{4}$$

where $X \in \mathbb{R}^{m \times n}$, as defined in Chai et al. [2]. By analyzing eq. (2), x^* would be the last of the right-hand singular vectors of A, because this is the singular vector associated with the smallest singular value of A. In other words, the following relation holds:

$$Ax^* = A\sigma_{min}, \tag{5}$$

where σ_{min} is the smallest singular value of A. By performing the SVD on $X = A^\top$, we see that $X = U\Sigma V^\top = A^\top$, and by transposing this equation we see that $A = V\Sigma^\top U^\top$. This is a valid SVD of A, since both V and U are orthogonal matrices, and $\Sigma^\top$ is a diagonal matrix that contains the singular values of A. Therefore the solution x^* to the total least squares minimization problem is the last column vector of U.

We now assume that FedSVD is a black box. Each user i inputs the transformation $X_i = f(A_i) = {A_i}^\top$ into FedSVD, and receives the decomposed $X_i = U\Sigma {V_i}^\top$, where U and Σ are shared among all participants, and V_i is private. Since x^* is the last column vector of U, everyone gets the solution x^* to the problem, without sacrificing confidentiality.

*2) **Example of Practical Use of Method**:* This specific use case could be beneficial to companies wanting to collaborate on finding solutions to total least squares minimization problems. If several companies with data sets with equal structures do not have enough data themselves in order to find a trustworthy solution to the TLSM problem, they could use this federated method to find the solution x^* satisfying all companies' data sets, instead of only their own.

B. Federated Computation of Rank and Fundamental Subspaces

*1) **Algorithm Description**:* Three of the most important quantities of a matrix A that may be derived from the SVD are the null space, range and rank of A. As demonstrated in the previous section, we are able to solve the total least squares minimization problem by tweaking the input parameter of the FedSVD algorithm. What we essentially do here is finding the null space of the matrix A, and then choosing the null vector associated with the smallest singular value as the answer. In this section it will be shown how to derive the rank of A and how to extract information about its fundamental subspaces using FedSVD.

Each user i has data matrix A_i, and defines the input to FedSVD as $X_i = A_i$. The total data matrix A is in this use case defined as

$$A = [A_1 \dots A_i \dots A_k]. \tag{6}$$

After running the algorithm, each user gets the decomposed data matrix $A_i = X_i = U\Sigma {V_i}^\top$. The diagonal matrix Σ has $p = \min\{m, n\}$ singular values, where the first r are non-zero, and the last $p - r$ are zero.

This tells us that the rank of A is equal to r, which is equivalent to the row and column space of A having dimension r. From this, we also know that the null space of A is of dimension $n-r$, and the left null space is of dimension $m-r$.

From the left singular matrix U, we may deduce the range of A, which is the space that is spanned by the columns of A. The first r columns of U span the same space as A. Thus, we have a description of the space spanned by A in terms of orthogonal unit vectors. The last $m - r$ columns of U is an orthogonal basis for the left null space of A.

The last matrix provided by FedSVD is V_i. The right singular matrix V of A, is composed of V_i in the following way: $V = \begin{bmatrix} V_1 \\ \vdots \\ V_i \\ \vdots \\ V_k \end{bmatrix}$. This means that V_i is composed of some of the rows of V, and V_i therefore does not contain any full columns of V. We therefore can not deduce much valuable information about the subspaces of A from V_i.

*2) **Example of Practical Use of Method**:* This use case of FedSVD may be used to gain more insight about how samples might behave in a test taking scenario. By assuming that A is composed of A_i like $A = [A_1 \dots A_i \dots A_k]$, and that each row of A represents one feature, and each column of A represents one sample, then every user i shares the same

feature space, but a different sample space.

An example of a practical setting where the above statements could apply, and that the data matrices A_i should be kept private, is in a medical setting. My knowledge of medicine is very limited, but I will still try to present a plausible use case for this method. For example, blood test data from a large number of patients could be stored in A, where the feature space (the row space of A) could be spanned by features such as blood type, blood pressure, etc., while the sample space (the column space of A) could be spanned by blood samples of different patients. By performing FedSVD on the data matrix A, we could get information about the space in which all samples of A live, thus getting more information about how future samples might look like, as the user i collects more samples. This is because we know the range of A, represented by the first r columns of the left hand singular matrix U. If some samples collected in the future would seem to be out of the ordinary with respect to A_i (i.e. not part of the space spanned by the columns of A_i), then knowing that this sample is ordinary with respect to A (i.e. part of the space spanned by the columns of A), then this sample need not be as unordinary as first expected.

If we assume the same example of blood sampling, then the rank of A could also give valuable information about the data generating process. The rank of A is the same as the number of linearly independent columns of A, i.e. the number of columns that can not be expressed as a scaled sum of the other columns in the matrix. The rank could then give an indication of how many different types of blood samples there could be. This could in turn be used to group people's blood into different categories, and in turn create medicines tailored to each specific type.

V. A Note Regarding the Preservation of Data Privacy

The use cases presented in this paper have used the FedSVD algorithm as described in Chai et al. [2], but tweaked the input and output parameters in order to gain new insight from the algorithm. Chai et al. designed this algorithm with privacy preservation in mind, and proves in their paper that their algorithm really preserves data privacy.

In the first use case introduced in this paper, "Federated Total Least Squares Minimization", the data matrix A is structured like in eq. (1), which is differently than in "Federated Computation of Rank and Fundamental Subspaces", where it is structured like in eq. (6). In addition, the input X is a different transform of A in the two cases: in the first case, the input is $X_i = {A_i}^\top$, while in the second case it is $X_i = A_i$. The FedSVD algorithm guarantees privacy for input ${X_i}^{(1)} = f(A_i)$ [2], but it does not guarantee privacy if it also is run a second time with input ${X_i}^{(2)} = g(A_i) \neq f(A_i) = {X_i}^{(1)}$, since this could reveal more about the underlying data matrix A_i to the other users than the developers of FedSVD intended.

In the section "Federated Total Least Squares Minimization" the computation of a right hand singular matrix of eq. (1) is utilized in order to find a Total Least Squares Minimization. In the section "Federated Computation of Rank and Fundamental Subspaces", the left hand singular matrix of eq. (6) is computed. Even though the data matrices A used in these two examples differ from one another, it might be possible to reconstruct other users' data, if both these methods are run consecutively. I therefore advice not to do this, unless it can be proven that it still preserves privacy.

VI. GitHub Implementation

The FedSVD algorithm, as well as how it is used in Principal Component Analysis (PCA), Linear Regression (LR), and Latent Semantic Analysis (LSA) may be found on the GitHub page of Di Chai, here.

The algorithms explained under "Federated Total Least Squares Minimization" and "Federated Computation of Rank and Fundamental Sub-spaces", as well as the algebraical aspects of FedSVD are implemented on my GitHub page, here. Here there is also a script demonstrating the before-mentioned algorithms.

A. *FedSVD.m*

The linear algebra aspects of the FedSVD algorithm by Chai et al. Takes X_i from each user i as input, and returns the matrices U and Σ, as well as the cell structure V_i.

B. *FedTLS.m*

The Federated Total Least Squares Minimization algorithm. Takes all A_i as input, and returns the vector x^*.

C. *FedRankAndSubspaces.m*

A method for extracting information about the rank and some of the fundamental subspaces of the data matrix. Takes all A_i as input, and returns an orthogonal basis of A, an orthogonal basis for the left nullspace of A, as well as numbers p (the number of singular values of A) and r (the number of nonzero singular values of A).

D. *testFunctions.m*

A script showcasing the three above-mentioned functions.

REFERENCES

[1] Konečný, J., McMahan, H. B., Yu, F. X., Richtarik, P, Suresh, A. T., & Bacon, D. (2016). Federated Learning: Strategies for Improving Communication Efficiency. *arXiv preprint*. arXiv:1610.05492.

[2] Chai, D., Wang, L., Zhang, J., Yang, L., Cai, S., Chen, K., & Yang, Q. (2022). Practical Lossless Federated Singular Vector Decomposition over Billion-Scale Data. *arXiv preprint*. arXiv:2105.08925.

[3] Liu, B., Pejó, B., & Tang, Q. (2023). Privacy-Preserving Federated Singular Value Decomposition. *Cryptology ePrint Archive*. URL: https://eprint.iacr.org/2022/1271

[4] Hartebrodt, A., Röttger, R., & Blumenthal, D. B. (2022). Federated singular value decomposition for high dimensional data. *arXiv preprint*. arXiv:2205.12109v1.