

SYSTEM AND METHOD FOR IMPLEMENTING DIFFERENTIALLY PRIVATE PRINCIPAL COMPONENT ANALYSIS (PCA) FOR VERTICALLY PARTITIONED DATA

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to systems and methods for implementing differentially private principal component analysis (PCA) for vertically partitioned data.

BACKGROUND

[0002] Data privacy is a critical concern in federated learning (FL), where a server and multiple clients train a model collaboratively on distributed data that contain sensitive information. As a rigorous framework for protecting sensitive information, differential privacy (DP) has been widely adopted in FL to address the privacy concern. However, traditionally the current approach for DP in FL predominantly focuses on horizontally partitioned data, where the sensitive dataset is partitioned among clients by records (i.e., rows). Federated learning naturally preserves some level of data privacy, by keeping the client's data localized, and providing the server and other clients with restricted access to the local data. In the meantime, secure multiparty computation (MPC) has been applied to FL to further enhance data privacy, in the sense that clients can jointly compute a function using MPC, without exposing their sensitive inputs to others directly. However, MPC alone only prevents privacy leakage during the process of the computation, and does not prevent an adversary from inferring the private input from the sensitive outcome of the computation. In other words, MPC solves the problem of how to compute the sensitive outcome without revealing the private inputs, but what MPC computes (i.e., the outcome) still leaks information about the inputs.

[0003] Currently, the use of DP in FL for vertically partitioned data (vertical FL), where the dataset is partitioned by attributes (e.g., columns), has not been done. According, efficient and trusted approaches for DP in FL for vertically partitioned data is desired.

SUMMARY

[0004] Various embodiments concern a computer system for implementing differentially private principal component analysis (PCA) for vertically partitioned data. The system includes

a first client possessing a first column vector, and a second client possessing a second column vector. The first client is configured to discretize the first column vector to obtain a first discretized column vector and the second client is configured to discretize the second column vector to obtain a second discretized column vector. The first client is configured to introduce a first noise to the first discretized column vector and the second client is configured to introduce a second noise to the second discretized column vector to obtain a PCA result.

[0005] According to one embodiment, the PCA result is a combination of the first discretized column vector with the first noise and the second discretized column vector with the second noise.

[0006] According to one embodiment, first client and the second client are configured to send the PCA result to a server for further processing using a secret sharing protocol.

[0007] According to one embodiment, the secret sharing protocol is a BGW protocol.

[0008] According to one embodiment, the BGW protocol is used to compute the first discretized column vector with the first noise and the second discretized column vector with the second noise into the PCA result without the first client knowing the second discretized column vector and without the second client knowing the first discretized column vector.

[0009] According to one embodiment, the server is configured to reduce the PCA result into a covariance matrix to reduce the size of data.

[0010] According to one embodiment, the first client is configured to introduce the first noise and the second client is configured to introduce the second noise using a predetermined noise parameter.

[0011] According to one embodiment, the first noise and the second noise are Skellam noises.

[0012] According to one embodiment, the first client is configured to discretize the first column vector and the second client is configured to discretize the second column vector using a predetermined discretization algorithm to reduce the size of the first column vector and the second column vector.

[0013] Various embodiments concern a computer implemented method for implementing differentially private principal component analysis (PCA) for vertically partitioned data including: discretizing a first column vector from a first client to obtain a first discretized column vector; discretizing a second column vector from a second client to obtain a second discretized column vector; introducing a first noise to the first discretized column vector using

the first client and introducing a second noise to the second discretized column vector using the second client to obtain a PCA result.

[0014] According to one embodiment, the PCA result is a combination of the first discretized column vector with the first noise and the second discretized column vector with the second noise.

[0015] According to one embodiment, first client and the second client are configured to send the PCA result to a server for further processing using a secret sharing protocol.

[0016] According to one embodiment, the secret sharing protocol is a BGW protocol.

[0017] According to one embodiment, the BGW protocol is used to compute the first discretized column vector with the first noise and the second discretized column vector with the second noise into the PCA result without the first client knowing the second discretized column vector and without the second client knowing the first discretized column.

[0018] According to one embodiment, the server is configured to reduce the PCA result into a covariance matrix to reduce the size of data.

[0019] According to one embodiment, the first client is configured to introduce the first noise and the second client is configured to introduce the second noise using a predetermined noise parameter.

[0020] According to one embodiment, the first noise and the second noise are Skellam noises.

[0021] According to one embodiment, the first client is configured to discretize the first column vector and the second client is configured to discretize the second column vector using a predetermined discretization algorithm to reduce the size of the first column vector and the second column vector.

[0022] According to one embodiment, a computer program element is provided comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method described above.

[0023] According to one embodiment, a computer-readable medium is provided comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method described above.

[0024] It should be noted that embodiments described in context of the method for using a system for implementing differentially private principal component analysis (PCA) for vertically partitioned data are analogously valid for the system and vice versa.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

- FIG. 1 shows an exemplary system for implementing differentially private principal component analysis (PCA) for vertically partitioned data according to an embodiment.
- FIG. 2 shows an exemplary method for implementing differentially private principal component analysis (PCA) for vertically partitioned data according to an embodiment.
- FIG. 3 shows charts showing performance of PCA under various conditions according to an embodiment.
- FIG. 4 shows charts showing performance results under various conditions according to an embodiment.

DETAILED DESCRIPTION

[0026] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0027] Embodiments described in the context of one of the systems or methods are analogously valid for the other systems or methods. Similarly, embodiments described in the context of a system are analogously valid for a method, and vice-versa.

[0028] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0029] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0030] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0031] In the following, embodiments will be described in detail.

[0032] FIG. 1 shows an exemplary system for implementing differentially private principal component analysis (PCA) for vertically partitioned data according to an embodiment.

[0033] In various embodiments, data privacy is a central concern in federated learning (FL) where a server and multiple clients train a model collaboratively on distributed data. Federated

[0034] learning naturally preserves some level of data privacy, by keeping the client’s data localized, and providing the server and other clients with restricted access to the local data. In the meantime, secure multiparty computation (MPC) has been applied to FL to further enhance data privacy, in the sense that clients can jointly compute a function using MPC, without exposing their sensitive inputs to others directly. However, MPC alone only prevents privacy leakage during the process of the computation, and does not prevent an adversary from inferring the private input from the sensitive outcome of the computation,

[0035] In other words, MPC solves the problem of how to compute the sensitive outcome without revealing the private inputs, but what MPC computes (i.e., the outcome) still leaks information about the inputs.

[0036] In various embodiments, it is desirable to design a mechanism that (approximately) computes principal components of an input dataset, while preserving two levels of differential privacy simultaneously. These privacy guarantees encompass the individual records within the dataset as well as the local partition of each participating client in the FL process.

[0037] The system which may be referred to as SSS-PCA (Skellam Secret Sharing-PCA), neatly integrates Skellam noise, which enables DP in FL, and employs Secret Sharing, which further enhances privacy. More details of the system will be described below.

[0038] In various embodiments, the system 100 includes a first client 101, a second client 111 and a server 121.

[0039] In various embodiments, the first client 101 may possess a first column vector 102 containing data of the first client 101. The first client may include a first client processor 103 for performing any computer implemented functions regarding the first column vector 102.

[0040] In various embodiments, the second client 111 may possess a second column vector 112 containing data of the second client 111. The second client 111 may include a second client processor 113 for performing any computer implemented functions regarding the second column vector 112.

[0041] In various embodiments, the server 121 may include a server processor 123 for performing any computer implemented functions regarding the column information or data of the first client 101 and the second client 111.

[0042] In various embodiments, there is a processor 130 for processing information from the first client 101 and the second client 111. The processor 130 may be a part of server 121 or may be separated from server 121.

[0043] In various embodiments, the first client is configured to discretize the first column vector to obtain a first discretized column vector.

[0044] In various embodiments, the second client is configured to discretize the second column vector to obtain a second discretized column vector.

[0045] In various embodiments, the first client is configured to introduce a first noise to the first discretized column vector and the second client is configured to introduce a second noise to the second discretized column vector to obtain a PCA result.

[0046] In various embodiments, there is a processor 130 for processing information from the first client 101 and the second client 111.

[0047] In various embodiments, the processor 130 may receive the first discretized column vector and the first noise and/or the first discretized column vector with the first noise from the first client and the second discretized column vector and the second noise and/or the second discretized column vector with the second noise from the second client. The processor 130 may use the first discretized column vector and the first noise and/or the first discretized column vector with the first noise and the second discretized column vector and the second noise and/or the second discretized column vector with the second noise to obtain the PCA result.

[0048] In various embodiments, the PCA result is a combination of the first discretized column vector with the first noise and the second discretized column vector with the second noise.

[0049] In various embodiments, the first client and the second client are configured to send the PCA result to a server for further processing using a secret sharing protocol.

[0050] In various embodiments, the secret sharing protocol is a BGW protocol.

[0051] In various embodiments, the BGW protocol is used to compute the first discretized column vector with the first noise and the second discretized column vector with the second noise into the PCA result without the first client knowing the second discretized column vector and without the second client knowing the first discretized column.

[0052] In various embodiments, the server is configured to reduce the PCA result into a covariance matrix to reduce the size of data.

[0053] In various embodiments, the first client is configured to introduce the first noise and the second client is configured to introduce the second noise using a predetermined noise parameter.

[0054] In various embodiments, the first noise and the second noise are Skellam noises.

[0055] In various embodiments, PCA is a dimensionality reduction method that is used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Thus, PCA has the ability to reduce data size of information such as the first column vector 102 and the second column vector 112.

[0056] Advantageously, PCA aims at reducing the number of attributes of the dataset D while capturing its covariance by performing a linear transformation. Covariance measures the total variation of two random variables from their expected values. Using covariance, we can only gauge the direction of the relationship (whether the variables tend to move in tandem or show an inverse relationship).

[0057] In various embodiments, for a dataset (matrix) $D \in \mathbb{R}^{m \times n}$, m represents the number of records and n represents the number of attributes in D . In various embodiments, a rank- k subspace (e.g., a matrix $A \in \mathbb{R}^{n \times k}$), with the goal of maximizing the variance in DA , which is equivalent to finding the top k eigenvectors of the covariance matrix $C = D^T D$.

[0058] The covariance matrix C is computed as:

$$C[i, j] = \sum_{l=1}^m (D[l, i] \cdot D[l, j]), \quad (1)$$

for $i, j \in [n]$. Given the optimal rank- k subspace, V_k (e.g., the k -dimensional principal singular subspace of C), the error of a rank- k subspace A on describing matrix D is defined as:

$$err = \|DV_k\|_F^2 - \|DA\|_F^2, \quad (2)$$

where $\| \cdot \|_F$ denotes the Frobenius norm of a matrix. The error reaches the minimum (0) when the columns of A match the principal rank- k right singular subspace of D .

[0059] In various embodiments, differential privacy is a mathematical framework for ensuring the privacy of individuals in datasets. It can provide a strong guarantee of privacy by allowing data to be analyzed without revealing sensitive information about any client in the dataset.

[0060] In various embodiments, differential privacy (DP) quantifies the indistinguishability of the output distributions for a mechanism on neighbouring datasets, where D and D' are called neighbouring datasets if they differ by one record (written as $D \sim D'$).

[0061] In various embodiments, there may be three types of DP (different definitions).

[0062] In various embodiments, for definition 1 ((ϵ, δ))- A randomized mechanism $M: D \rightarrow Y$ satisfies (ϵ, δ) differential privacy (DP) if for any $\mathcal{O} \subseteq \mathcal{Y}$ it holds that:

$$\sup_{D \sim D'} \Pr[M(D) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[M(D') \in \mathcal{O}] + \delta. \quad (3)$$

[0063] Here smaller ϵ or δ imply higher difficulty in distinguishing the distributions, and hence stronger privacy guarantees. Given a mechanism (function) of interest, a canonical approach to make it differentially private is to perturb its outcome by injecting random noise, the scale of which is calibrated to the sensitivity of the mechanism.

[0064] In various embodiments, for definition 2, let $\| \cdot \|$ represent a norm measure of interest (e.g. L_2 -norm), then the sensitivity $S(M)$ of a function $M: D \rightarrow \mathbb{R}_n$, denoted as $S(M)$, is

$$S(M) = \max_{D \sim D'} \|M(D) - M(D')\|.$$

defined as

[0065] In various embodiments, for definition 3, a randomized mechanism M satisfies (α, τ) Renyi differential privacy (RDP) for some $\alpha \in (0, 1) \cup (1, \infty)$ if

$$\sup_{D \sim D'} D_\alpha(M(D) \| M(D')) \leq \tau,$$

where $D_\alpha(P \| Q)$ denotes the Renyi divergence of P from Q that are defined over the same domain.

[0066] In various embodiments, secret sharing is a class of methods for distributing a secret among a group of parties in a way such that: 1) no single party is able to learn any non-trivial information about the secret; 2) when a sufficiently large number of parties combine their information, the secret is reconstructed.

[0067] Lagrange interpolation theorem, states that N points on a polynomial uniquely define a polynomial of degrees lower than N . With this in mind, the secret holder divides the secret into N shares by computing N points on a polynomial of degree $N - 1$, whose constant term is exactly the original secret, and the other $N - 1$ coefficients are randomly selected by the secret holder and kept privately. As a result, any subset of less than N parties cannot learn anything non-trivial about the secret. On the other hand, N parties can reconstruct the secret by polynomial interpolation.

[0068] In various embodiments, a type of secret sharing may be a BGW (Ben-Or, Goldwasser and Widgerson) protocol.

[0069] In various embodiments, in BGW protocol, the outcome of a polynomial function $f(s_1, \dots, s_N)$, where the function is known to all parties, without revealing the private input to others. The three-step execution of BGW is as follow:

[0070] 1. First, each client distributes their private input as a secret share to other clients, using the Shamir's algorithm. Note that no client can infer any other client's private input, which is enforced by the security property of Shamir's algorithm.

[0071] 2. Next, each client simulates the computation of function f using a digital circuit while keeping the value of each secret as secret shared by all parties. Similar to the first step, the value of each secret cannot be inferred by any party.

[0072] 3. Finally, all of the clients reconstruct the true outcome of f , using their secret shares.

[0073] The BGW protocol achieves a notion of information-theoretic security, e.g., every client can learn absolutely no more information about other clients' private inputs than what can be learned from the output itself, i.e. $f(s_1, \dots, s_N)$. Note that this guarantee of BGW is different from DP since an adversary can still extract information of the private inputs from the sensitive outcome of f .

[0074] In various embodiments, noise is introduced to the first vertical column and the second vertical column for additional privacy. The noise can be a Gaussian noise or a Skellam noise. Skellam noise is noise which uses skellam distribution.

[0075] A random variable Z obeys a symmetric Skellam distribution with respect to a parameter μ if it is the difference between two independent Poisson random variables of the same parameter μ . By linearity of Poisson distributions, the sum of two independent Skellam noise sampled from $Sk(\mu_1)$ and $Sk(\mu_2)$ has distribution $Sk(\mu_1 + \mu_2)$. $Sk(\mu)$ as the ensemble of

n independent random variables sampled from $\text{Sk}(\mu)$. The following divergence guarantee for perturbing the outcome of an integer-valued function with noises sampled from $\text{Sk}(\mu)$.

[0076] For any integer $\alpha > 1$, and any n -dimensional integer-valued function F with bounded output range, it holds that:

$$\begin{aligned} D_\alpha(F(D) + \text{Sk}^n(\mu) \| F(D') + \text{Sk}^n(\mu)) &\leq \tau, \\ \text{for } \Delta_2 = \|F(D) - F(D')\|_2, \Delta_1 = \|F(D) - F(D')\|_1, \text{ and } \tau \text{ computed as} \\ \tau &= \frac{\alpha \Delta_2^2}{4\mu} + \min_{\alpha > 1} \left(\frac{(2\alpha - 1)\Delta_2^2 + 6\Delta_1}{16\mu^2}, \frac{3\Delta_1}{4\mu} \right). \end{aligned} \quad (4)$$

[0077] In various embodiments, for principal component analysis in vertical federated learning, let $D \in \mathbb{R}^{m \times n}$ represent the sensitive data where each row $D[i, :] \in \mathbb{R}^{1 \times n}$ represents the i -th ($i \in [m]$) record vector consisting of n real-valued attributes, and each column $D[:, j] \in \mathbb{R}^{m \times 1}$ represents the vector of attribute $j \in [n]$ from all records.

[0078] For vertical federated learning (VFL), where each of the clients possesses a collection of attributes (e.g., a subset of columns in D). More specifically, if there are N

[0079] clients, where client q possesses a portion D_q of D ($q \in [N]$). Here D_q consists of a collection of attributes A_q such that $D_q = \{D[:, j] : j \in A_q\}$. Besides, it is assumed that the collections of columns (resp. attributes) $\{D_q : q = 1, \dots, N\}$ (resp. $\{A_q : q = 1, \dots, N\}$) are

mutually exclusive. $\sum_{q=1}^N |A_q| = n$. Given an integer k (usually $k \ll n$), an untrusted server aims to (approximately) learn a rank k subspace $\tilde{V}_k \in \mathbb{R}^{n \times k}$ that preserves most of the variance in the original dataset D , namely, maximizing $\|D\tilde{V}_k\|_F^2$. The privacy-preserving mechanism disclosed is able to find such $\tilde{V}_k$, while preserving differential privacy regarding any individual record such as $D[i, :]$ and any dataset portion of a client such as D_q .

[0080] In various embodiments, there are three entities involved in the threat model under vertical federated learning: an individual record $D[i, :]$ ($i \in [m]$), a client q possessing the dataset portion D_q ($q \in [N]$), and the untrusted server who wants to learn about dataset D .

[0081] First, it is assumed that every individual trusts every client, but not the server. This means that client q has full access to the corresponding private portion D_q containing the sensitive information of the individual records (including the information for identifying an individual) which is required to be protected against the server. The validity of such an

assumption can be interpreted by an example. For example, a scenario of online shopping, given the same group of customers, each online retailer which is considered as a client, possesses the purchasing or browsing histories of each individual customer, which are considered as attributes. However, other than these trusted clients, an individual may not want any other entity, including the server, to know their own private record (including the information for identifying an individual), which also means that is considered private for the server. The above discussion leads to the following DP definition.

[0082] In Definition 4, $((\epsilon, \delta)$ -record-server DP). a randomized mechanism M satisfies (ϵ, δ) record server DP if for any set of output $\mathcal{O} \subseteq \text{Range}(\mathcal{M}_{\text{server}})$, it holds that:

$$\sup_{D \sim D'} \Pr[\mathcal{M}_{\text{server}}(D) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}_{\text{server}}(D') \in \mathcal{O}] + \delta,$$

[0083] Here “ $\sim$ ” represents the neighboring relation of two datasets (resp. matrices) that one can be obtained from the other by adding/removing one record (resp. row).

[0084] Using the online retailer example, a client (e.g., online retailer), also has the incentive

[0085] to prevent their competitors to learn about their private data partition (e.g., customers’ purchasing or browsing histories), since otherwise, a competitor may send personalized advertisements to the client’s customers, which could cause damage to the client’s business, leading to the following DP definition.

[0086] In Definition 5, a randomized mechanism M satisfies (ϵ, δ) -partition

[0087] client DP if for every client q and the private partition D_q and any set of output $\mathcal{O} \subseteq \text{Range}(\mathcal{M}_{\text{client } q})$, it holds that

$$\sup_{D_q^c \sim D_q'^c} \Pr[\mathcal{M}_{\text{client } q}(D_q^c \circ D_q) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}_{\text{client } q}(D_q'^c \circ D_q) \in \mathcal{O}] + \delta,$$

[0088] Here “ $\sim$ ” represents the neighboring relation that one set can be obtained from the other by replacing one row. For any $q = 1, \dots, N$, we use $\overline{D_q^c}$ to represent the complement matrix that takes D_q out of the complete matrix D . We write $D_q^c \circ D_q$ to represent the reverse operation, that is, constructing a matrix by concatenating D_q and $\overline{D_q^c}$ following the same order of indices as of D .

[0089] Note that the neighboring relation in partition-client DP is defined on the action of replacing one row, instead of the action of adding/removing one row in record-server DP. This is because every client already has full access to their private data portion, including the information needed to identify an individual. Hence, it is assumed that the records are aligned in the clients' partitions and that m is considered non-private for the clients, meaning that the only information that a client is curious about is the value of the attributes in other clients' partitions. Every party (either a server or a client) strictly follows the execution of M while trying to infer the private data in other clients' partitions.

[0090] In various embodiments, each client first independently perturb their local data (e.g., a submatrix) with random Gaussian noise. Next, all the clients reveal their perturbed local data to one of the clients (without loss of generality, say client 1), who then reconstructs the whole perturbed matrix; computes the covariance matrix; and sends top k eigenvectors of the covariance matrix to the server. The privacy guarantee of this baseline follows from the DP guarantee of additive Gaussian noise and that post-processing (computing the eigenvectors) preserves DP.

[0091] In various embodiments, the SSS-PCA, is built upon the classic DP algorithm for PCA and injects independent noises to the upper diagonal of the covariance matrix $C = DTD$

$$C[i, j] = \sum_{l=1}^m (D[l, i] \cdot D[l, j]),$$

defined as follows: , for $i, j \in [n]$. After this, an adversary could compute a k -dimensional singular subspace of the perturbed covariance matrix $\tilde{C}$ using standard non-DP algorithms. The privacy guarantee follows from the fact that $\tilde{C}$ is already differentially private and post-processing preserves DP.

[0092] First, the exact $C[i, j]$ is computed from two columns $D[:, i]$ and $D[:, j]$, which could be stored in different clients. In such cases, the computation for $C[i, j]$ cannot be done by one client alone, which requires use of secret sharing to perform this computation without directly revealing the private inputs (e.g., columns of D). However, secret sharing, does not prevent a curious party from inferring the private inputs from the computation outcome, thus injection of noise to the inputs preserve differential privacy.

[0093] The SSS-PCA is shown in Algorithm 1 below.

[0094] It is needed to compute the entry of the covariance matrix at the j -th column of the i -th row while preserving privacy. The first case is when both columns $D[:, i]$ and $D[:, j]$

(including the case when $i = j$). In such cases, client p who possesses the column(s) discretizes the column vector(s) using Algorithm 2, and obtains $\overline{D}[:, i]$ (and $\overline{D}[:, j]$) (Line 5 in Algorithm 1), where the overline represents the discretized version. Next, the client samples a Skellam noise z_p from $\text{Sk}(2\mu)$ (Line 6) and computes:

$$R[i, j] = \sum_{l=1}^m \overline{D}[l, i] \cdot \overline{D}[l, j] + z_p, \quad (5)$$

Algorithm 1: SSS-PCA

Input: Private matrix $D \in \mathbb{R}^{m \times n}$; number of clients N , discretization parameter γ , noise parameter μ .

- 1 The server initializes $\tilde{C} \in \mathbb{R}^{n \times n}$ as a zero matrix.
- 2 **for** $i = 1 \dots n$ **do**
- 3 **for** $j = i \dots n$ **do**
- 4 **if** Columns $D[:, i]$ and $D[:, j]$ are possessed by the same client p **then**
- 5 Client p discretizes the column vector(s) with parameter γ using Algorithm 2, and obtains $\overline{D}[:, i]$ (and $\overline{D}[:, j]$).
- 6 Client p samples Skellam noise z_p from $\text{Sk}(2\mu)$.
- 7 Client p computes and reports the result of $R[i, j]$ (defined as in Eq. (5)) to the server.
- 8 **else**
- 9 Client p (resp. client q) who possesses column $D[:, i]$ (resp. column $D[:, j]$) discretizes the column vector with parameter γ using Algorithm 2, and obtains $\overline{D}[:, i]$ (resp. $\overline{D}[:, j]$).
- 10 Client p (resp. client q) samples Skellam noise z_p (resp. z_q) from $\text{Sk}(\mu)$.
- 11 Clients p and q use the BGW protocol to compute and report the results of $R[i, j]$ (defined as in Eq. (6)) to the server.
- 12 The server assigns $\tilde{C}[i, j] = \tilde{C}[j, i] = \frac{1}{\gamma^2} \cdot R[i, j]$.

Output: $\tilde{V}_k$, the k -dimensional principal singular subspace (i.e., top k eigenvectors) of $\tilde{C}$.

Algorithm 2: Discretization Procedure

Input: Vector to discretize $x \in \mathbb{R}^n$, discretization parameter γ

- 1 Initialize $v \in \mathbb{R}^n$ as a zero vector.
- 2 $u \leftarrow \gamma \cdot x$.
- 3 **for each dimension** $j = 1 \dots n$ **do**
- 4 Flip a coin with heads probability $u(j) - \lfloor u(j) \rfloor$.
- 5 **if** Heads **then**
- 6 $v(j) \leftarrow \lfloor u(j) \rfloor + 1$.
- 7 **else**
- 8 $v(j) \leftarrow \lfloor u(j) \rfloor$.

Output: Integer-valued vector $u \in \mathbb{Z}^n$.

[0095] The client reports the outcome to the server (Line 7). The other case is when columns $D[:, i]$ and $D[:, j]$ are possessed by different clients, say client p and client q . In such cases, client p (resp. client q) who possesses the column vector $D[:, i]$ (resp. vector $D[:, j]$) privately and independently discretizes the vector, obtaining the integer-valued column vector $\overline{D}[:, i]$ (resp. vector $\overline{D}[:, j]$) (Line 9). Next, the two clients privately and independently generate

two Skellam noises z_p and z_q from $\text{Sk}(\mu)$ (Line 10). After that, the two clients use the BGW protocol to compute:

$$R[i, j] = \sum_{l=1}^m \overline{D}[l, i] \cdot \overline{D}[l, j] + z_p + z_q, \quad (6)$$

[0096]

[0097] The two clients reports the outcome to the server (Line 11). After the computation is done for all upper diagonal entries, the server obtains the perturbed covariance matrix $C^\sim$ by downscaling the outcome (Line 12), and computes its top k eigenvectors on his own.

[0098] In various embodiments, the noise injection for $R[i, j]$ in Eq. (6) has to be contributed by two clients (both z_p and z_q). Otherwise, the single client (or any other party) who perform the noise injection can infer private information about the underlying D from the exact outcome.

[0099] SSS-PCA satisfies DP, regarding both a record and a client.

[00100] Lemma 2. Given discretization parameter γ and noise parameter μ , Algorithm 1 satisfies $(\alpha, \tau_{\text{server}})$ record-server RDP with:

$$\tau_{\text{server}} = \frac{\alpha \Delta_2^2}{8\mu} + \min \left(\frac{(2\alpha - 1)\Delta_2^2 + 6\Delta_1}{64\mu^2}, \frac{3\Delta_1}{8\mu} \right), \quad (7)$$

[00101]

[00102] Algorithm 1 also satisfies $(\alpha, \tau_{\text{client}})$ -partition-client RDP with

$$\tau_{\text{client}} = \frac{\alpha \Delta_2^2}{\mu} + \min \left(\frac{(2\alpha - 1)\Delta_2^2 + 3\Delta_1}{4\mu^2}, \frac{3\Delta_1}{2\mu} \right), \quad (8)$$

where $\Delta_2 = \sqrt{\gamma^2 + d}$ and $\Delta_1 = \min(\gamma^2 + d, \sqrt{d}\sqrt{\gamma^2 + d})$.

[00103] Lemma 3. Let V_k be the rank- k subspace of the original matrix D and let $\tilde{V}_k$ be the principal rank- k subspace of matrix $C^\sim$ (computed with additive Skellam noise), then with high probability

$$\|\tilde{D}\tilde{V}_k\|_F^2 \geq \|DV_k\|_F^2 - O(k\sqrt{n}). \quad (9)$$

[00104] Lemma 3 quantifies how well $\tilde{V}_k$ captures the variance of D . To prove Lemma 3, it suffices to quantify how well R captures the variance of γD . The problem is reduced to bounding the largest eigenvalue (spectral norm) of the matrix $R - \gamma^2 DTD$ in the worst case, which can be decomposed as $E_d + E_p$, where E_d is a symmetric random matrix whose entries

represent the randomness due to discretization (recall Line 4 in Algorithm 1) and E_p is a symmetric random matrix whose entries represent the randomness due to DP (e.g., each entry is sampled from $\text{Sk}(2\mu)$). The entries (in the upper diagonal) of E_d are independent random variates that are uniformly bounded with finite variance and the entries (in the upper diagonal) of E_p are sub-exponential, which means that the largest eigenvalues of E_d and E_p are of order $O(\sqrt{n})$ with high probability $1 - o(1/n)$.

[00105] FIG. 2 shows an exemplary method for implementing differentially private principal component analysis (PCA) for vertically partitioned data according to an embodiment.

[00106] As shown in FIG. 2, there may be a method 200 for for implementing differentially private principal component analysis (PCA) for vertically partitioned data. In the method 200, a first step 202 may include discretizing a first column vector from a first client to obtain a first discretized column vector. A second step 204 may include discretizing a second column vector from a second client to obtain a second discretized column vector. A third step 206 may include introducing a first noise to the first discretized column vector using the first client and introducing a second noise to the second discretized column vector using the second client to obtain a PCA result.

[00107] It will be understood that the above operations described above relating to FIG. 2 are not limited to this particular order. Any suitable, modified order of operations may be used.

[00108] FIG. 3 shows charts showing performance of PCA under various conditions according to an embodiment. FIG. 4 shows charts showing performance results under various conditions according to an embodiment.

[00109] The solution is compared with existing solutions on KDDCUP dataset ($m = 195666$ and $n = 817$) and the ACSIncome dataset of California ($m = 145585$ and $n = 117$) and shown in FIGS. 3 and 4.

[00110] The average utility over 20 independent runs in FIG. 3, from which it is clear that the solution always outperforms the baseline solution and is close to the strong centralized-DP baseline under various parameter settings, confirming the efficacy of SSS-PCA. In particular, for KDDCUP, SSS-PCA performs almost the same as the strong centralized-DP baseline. For ACSIncome, the difference between SSS-PCA and the centralized baseline is only marginal when $\gamma \geq 6$, and note that even when $\gamma = 4$, SSS-PCA still outperforms the distributed baseline by a large margin.

[00111] An illustration for $k = 2$ on KDDCUP is shown in FIG. 4, where both the centralized-DP algorithm and the solution obtain high-quality 2-dimensional subspaces for the input dataset, as the normal and not normal points are separated with a large margin when projecting onto the subspaces.

[00112] The methods described herein may be performed and the various processing or computation units and the devices and computing entities described herein may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor. A "circuit" may also be software being implemented or executed by a processor, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which are described herein may also be understood as a "circuit" in accordance with an alternative embodiment.

[00113] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A computer system for implementing differentially private principal component analysis (PCA) for vertically partitioned data comprising:

a first client possessing a first column vector;

a second client possessing a second column vector;

wherein the first client is configured to discretize the first column vector to obtain a first discretized column vector and the second client is configured to discretize the second column vector to obtain a second discretized column vector;

wherein the first client to configured to introduce a first noise to the first discretized column vector and the second client to configured to introduce a second noise to the second discretized column vector to obtain a PCA result.

2. The computer system of claim 1, wherein the PCA result is generated based on a combination of the first discretized column vector with the first noise and the second discretized column vector with the second noise.

3. The computer system of claims 1-2, wherein at least one of the first client and the second client are configured to send the PCA result to a server for further processing using a secret sharing protocol.

4. The computer system of claim 3, wherein the secret sharing protocol is a BGW protocol.

5. The computer system of claim 4, wherein the BGW protocol is used to compute the first discretized column vector with the first noise and the second discretized column vector

with the second noise into the PCA result without the first client knowing the second discretized column vector and without the second client knowing the first discretized column.

6. The computer system of claims 3-5, wherein the server is configured to reduce the PCA result into a covariance matrix to reduce the size of data.

7. The computer system of claims 1-6, wherein the first client is configured to introduce the first noise and the second client is configured to introduce the second noise using a predetermined noise parameter.

8. The computer system of claims 1-7, wherein the first noise and the second noise are Skellam noises.

9. The computer system of claims 1-8, wherein the first client is configured to discretize the first column vector and the second client is configured to discretize the second column vector using a predetermined discretization algorithm to reduce the size of the first column vector and the second column vector.

10. A computer implemented method for implementing differentially private principal component analysis (PCA) for vertically partitioned data comprising:

discretizing a first column vector from a first client to obtain a first discretized column vector;

discretizing a second column vector from a second client to obtain a second discretized column vector;

introducing a first noise to the first discretized column vector using the first client and introducing a second noise to the second discretized column vector using the second client to obtain a PCA result.

11. The computer implemented method of claim 10, wherein the PCA result is generated based on a combination of the first discretized column vector with the first noise and the second discretized column vector with the second noise.
12. The computer implemented method of claims 10-11, wherein at least one of the first client and the second client are configured to send the PCA result to a server for further processing using a secret sharing protocol.
13. The computer implemented method of claim 12, wherein the secret sharing protocol is a BGW protocol.
14. The computer implemented method of claim 13, wherein the BGW protocol is used to compute the first discretized column vector with the first noise and the second discretized column vector with the second noise into the PCA result without the first client knowing the second discretized column vector and without the second client knowing the first discretized column.
15. The computer implemented method of claims 12-14, wherein the server is configured to reduce the PCA result into a covariance matrix to reduce the size of data.
16. The computer implemented method of claims 10-15, wherein the first client is configured to introduce the first noise and the second client to configured to introduce the second noise using a predetermined noise parameter.
17. The computer implemented method of claims 10-16, wherein the first noise and the second noise are Skellam noises.

18. The computer implemented method of claims 10-17, wherein the first client is configured to discretize the first column vector and the second client is configured to discretize the second column vector using a predetermined discretization algorithm to reduce the size of the first column vector and the second column vector.

19. A computer program element comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 10 to 17.

20. A computer-readable medium comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 10 to 17.