A COMMUNICATION EFFICIENT FEDERATED KERNEL k-MEANS

Anonymous authors
Paper under double-blind review

ABSTRACT

A federated kernel k-means algorithm is developed in this paper. This algorithm resolves two challenging issues: 1) how to distributedly solve the kernel k-means problem under federated settings; 2) how to maintain communication efficiency in the algorithm. To tackle the first challenge, a distributed stochastic proximal gradient descent (DSPGD) algorithm is developed to determine an approximated solution to the kernel k-means problem. To tackle the second challenge, a communication efficient mechanism (CEM) is designed to reduce the communication cost. Besides, the federated kernel k-means provides two levels of privacy preservation. Theoretical analysis shows: 1) DSPGD with CEM converges with an $O(1/T)$ rate, where T is the number of iterations; 2) the communication cost of DSPGD with CEM is unrelated to the number of data samples; 3) the clustering loss of the federated kernel k-means can approach that of the centralized kernel k-means. The experimental results show that the federated kernel k-means achieves the highest clustering quality with the communication cost reduced by more than 60%.

1 INTRODUCTION

Conventionally, kernel k-means (Dhillon et al., 2004) is conducted in a centralized manner where training data are stored in one place, such as a cloud server. However, as a rapidly growing number of devices are connected to the Internet, the volume of generated data increase exponentially (Chiang & Zhang, 2016). Uploading all these data to the cloud server can lead to large cost of communication bandwidth. For example, a smartphone manufacturer usually needs to analyze usage patterns of its smartphones, purposing to optimize energy consumption performance of these smartphones. The usage patterns can be obtained by clustering users' energy consumption data via kernel k-means. However, if the number of users reaches the order of hundreds of millions, it may not be a cost-effective scheme to upload users' energy consumption data to the cloud server. Besides, exposing users' raw data to the cloud server can lead to data privacy issues. To resolve these issues, a promising approach is to develop a federated learning algorithm (McMahan et al., 2017; Yang et al., 2019) for kernel k-means, which achieves comparable performance as the centralized kernel k-means. In this algorithm, a local training process is conducted at each user's device, based on the local data only. The local computational results, rather than the local data, are then uploaded to the cloud server to accomplish the kernel k-means clustering.

However, it is nontrivial to design a federated learning algorithm for kernel k-means due to three challenging issues: 1) how to solve the problem of kernel k-means in a distributed manner without sending users' data to a central place; 2) how to maintain communication efficiency in the algorithm; 3) how to protect users' data privacy in the algorithm. In this paper, a federated kernel k-means algorithm (denoted as FK k-means) is developed to resolve the first two issues. Meanwhile, FK k-means can provide two levels of privacy preservation: 1) users' local data are not exposed to the cloud server; 2) the cloud server cannot recover users' local data from the local computational results via matrix operations. FK k-means can also incorporate the differential privacy mechanism (Dwork et al., 2006) to provide stronger privacy, which is subject to future work. To tackle the first issue, a distributed stochastic proximal gradient descent (DSPGD) algorithm is designed to determine the eigenpairs of the kernel matrix $\mathbf{K}$. From these eigenpairs, an approximated solution to the kernel k-means problem can then be obtained (Wang et al., 2019; Ding et al., 2005). Since $\mathbf{K}$ is not available at the cloud server, a random feature method (Rahimi & Recht, 2008; Kar & Karnick,

2012) is used to estimate $\mathbf{K}$ with a matrix $\boldsymbol{\xi} = \frac{1}{D}\mathbf{A}\mathbf{A}^\top$, where the rows of $\mathbf{A}$ are D-dimension random feature vectors and are distributed over users' devices. It has been proved that the optimal solution $\mathbf{Z}^*$ to a convex stochastic composite optimization (SCO) problem of $\boldsymbol{\xi}$ is a low rank approximation to $\mathbf{K}$ (Zhang et al., 2016). Thus, the SCO problem needs to be solved in a distributed manner to obtain $\mathbf{Z}^*$ and the corresponding eigenvectors and eigenvalues, which is accomplished by DSPGD. More specifically, the $(t+1)$-th iteration of $\mathbf{Z}^*$, i.e., $\mathbf{Z}_{t+1}$, is determined at the cloud server by performing the singular value shrinkage (SVS) operation (Cai et al., 2010) on a weighted sum of $\mathbf{Z}_t$ and $\boldsymbol{\xi}_t$. The SVS operation requires computation of eigenpairs of the weighted sum. However, $\boldsymbol{\xi}_t$ is not known at the cloud server. In this case, the eigenpairs can be determined via the distributed Lanczos algorithm (DLA) (Penna & Stańczak, 2014). As a result, the eigenpairs of the kernel matrix $\mathbf{K}$ are finally obtained by DSPGD in a distributed way.

However, DSPGD is not communication efficient, i.e., the second challenging issue remains unresolved. To this end, a communication efficient mechanism (CEM) is designed to enhance DSPGD. CEM utilizes two properties of the weighted sum of $\mathbf{Z}_t$ and $\boldsymbol{\xi}_t$: positive semidefinite (p.s.d.) and low rank. Based on the p.s.d. property, a matrix $\mathbf{W}_t$ is designed such that $\mathbf{W}_t\mathbf{W}_t^\top$ equals the weighted sum, where $\mathbf{W}_t$ is constructed using $\mathbf{A}_t$ and the previous local computational results at users' devices. The eigenvectors of the weighted sum are exactly the left singular vectors of $\mathbf{W}_t$. Due to the low rank property, the dimension of the right singular vectors of $\mathbf{W}_t$ is much smaller than that of the left singular vectors of $\mathbf{W}_t$. Moreover, a left singular vector can be constructed based on the corresponding right singular vector and $\mathbf{W}_t$. Thus, in CEM the cloud server computes the right singular vectors by applying DLA to $\mathbf{W}_t^\top\mathbf{W}_t$. Based on the right singular vectors, the corresponding left singular vectors are then constructed at users' devices.

The theoretical analysis shows that DSPGD with CEM converges to $\mathbf{Z}^*$ at an $O(1/T)$ rate, where T is the iteration number. The communication cost of DSPGD with CEM is linear to the dimension of the right singular vector times the number of users, which can be much smaller than the number of data samples. The clustering loss of FK k-means can approach that of the centralized kernel k-means. The experimental results show that, compared with the state-of-the-art schemes, FK k-means can reduce the communication cost by more than 60% when the highest average normalized mutual information (NMI) scores are considered

2 Related Work

2.1 Distributed Kernel k-means

Many algorithms have been developed to conduct the kernel k-means clustering in a distributed way. The kernel approximation method is a popular approach employed in these algorithms, such as the Nyström method (Chitta et al., 2011; 2014) and the random feature method (Chitta et al., 2012). A trimmed kernel k-means algorithm (Tsapanos et al., 2015) decreases the computational cost and the space complexity by significantly reducing the non-zero entries in K via a kernel matrix trimming algorithm. In (Elgohary et al., 2014) an approximate nearest centroid (APNC) embedding is developed to embed the data samples so that the clustering assignment step of kernel k-means can be parallel executed. A communication efficient kernel principle component analysis (PCA) algorithm (Balcan et al., 2016) along with distributed liner k-means can approximately solve the kernel k-means problem while maintaining the communication efficiency. However, these algorithms are designed with an assumption that they are executed at the cloud server where the constraint on transmissions of the raw data samples does not exist.

2.2 Federated Learning

Federated learning (McMahan et al., 2017) is a new machine learning framework aiming to protect users' data privacy and save the communication cost during the learning process. In the framework, a local model is updated at each user's device, and these local models instead of users' local data are then aggregated at the cloud server to generate a global model. The distributed optimization method in the framework is applicable to the models whose optimization problem can be decomposed into several independent subproblems, such as neural networks. and many algorithms (Konečnỳ et al., 2016; Yang et al., 2018; Yurochkin et al., 2019) are developed. However, it is non-trivial to decompose the optimization problem of kernel k-means under the federated learning framework. Some

algorithms (Liu et al., 2017; Caldas et al., 2018) improve federated multi-task learning (Smith et al., 2017) with kernel. However, these algorithms either employ explicit feature mapping (Liu et al., 2017) that can lead to impractical computational cost, or require to send the support vectors of users' local data (i.e., some local data samples) to the cloud server (Caldas et al., 2018), which can leak users' privacy information. Due to these limitations, these algorithms are not applicable to kernel k-means under the federated learning framework. Recently, a concept of clustered federated learning (Ghosh et al., 2020; Sattler et al., 2020; Mansour et al., 2020) is proposed, where the clients of federated learning are clustered according to cosine similarity between their gradient updates. However, their clustering problems are different from kernel k-means problem, and thus are not feasible for kernel k-means problem in the federated setting.

3 PRELIMINARY

Let $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathcal{X}$ be a set of N data samples. Given a feature mapping $\phi(\cdot) : \mathcal{X} \mapsto \mathcal{H}$ and the number of clusters k, the problem of kernel k-means whose objective is to find an optimal indicator matrix $\mathbf{Y}^*$ can be written as

$$\min_{\mathbf{Y}\in\{0,1\}^{n\times k}} \mathrm{Tr}(\mathbf{K}) - \mathrm{Tr}(\mathbf{L}^{\frac{1}{2}}\mathbf{Y}^\top\mathbf{K}\mathbf{Y}\mathbf{L}^{\frac{1}{2}}) \quad \text{s.t. } \mathbf{Y}\mathbf{1}_k = \mathbf{1}_n, \tag{1}$$

where $\mathbf{K}$ is the kernel matrix with each entry $K_{ij} = \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_j)$, $\mathbf{L}^{\frac{1}{2}} = \mathbf{Diag}([\frac{1}{\sqrt{N_1}}, \dots, \frac{1}{\sqrt{N_k}}])$ is a diagonal matrix, N_i is the number of samples in the i-th cluster, and $\mathbf{1}_k$ is a column vector with all the k items equal to 1. However, the discrete matrix $\mathbf{Y}$ renders the problem in (1) difficult to solve. An efficient approach is first decomposing $\mathbf{K}$ as $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ via eigenvalue decomposition (EVD), and applying linear k-means to the matrix $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$ to obtain a suboptimal indicator matrix $\tilde{\mathbf{Y}}$ (Wang et al., 2019; Chitta et al., 2012; Ding et al., 2005). To reduce the computational complexity, only the first s column vectors of $\mathbf{H}$ are selected as the input of linear k-means (Wang et al., 2019; Cohen et al., 2015). Based on this approach, FK k-means is developed in this paper.

4 FEDERATED KERNEL k-MEANS

4.1 DISTRIBUTED STOCHASTIC PROXIMAL GRADIENT DESCENT

To solve the problem of kernel k-means in a distributed manner, a distributed stochastic proximal gradient descent (DSPGD) algorithm is developed to obtain the top-s eigenvectors and eigenvalues of the kernel matrix $\mathbf{K}$. Since users' data cannot be gathered in a central place, $\mathbf{K}$ cannot be explicitly constructed. Thus, a random feature method (Rahimi & Recht, 2008; Kar & Karnick, 2012) is used to estimate $\mathbf{K}$. For popular shift-invariant kernels $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ with Fourier representation $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \int p(\mathbf{w})\exp(j\mathbf{w}^\top(\mathbf{x}_i - \mathbf{x}_j))\mathrm{d}\mathbf{w}$ where $p(\mathbf{w})$ is a probability density function, they can be estimated by the random Fourier features (Rahimi & Recht, 2008) as follows. By randomly drawing D independent samples $\{\mathbf{w}_1, \dots, \mathbf{w}_D\}$ from $p(\mathbf{w})$, a random feature vector $\mathbf{a}(\mathbf{x}_i)$ for a data sample $\mathbf{x}_i$ can be written as $\mathbf{a}(\mathbf{x}_i) = [\sqrt{2}\cos(\mathbf{w}_1^\top\mathbf{x}_i + b_1), \dots, \sqrt{2}\cos(\mathbf{w}_D^\top\mathbf{x}_i + b_D)]^\top$ where $\{b_1, \dots, b_D\}$ are independent random variables drawn from $[0, 2\pi)$ uniformly. As a result, an unbiased estimation of $\mathbf{K}$ can be written as $\boldsymbol{\xi} = \frac{1}{D}\mathbf{A}\mathbf{A}^\top$ where $\mathbf{A} = [\mathbf{a}(\mathbf{x}_1), \cdots, \mathbf{a}(\mathbf{x}_n)]^\top$. It has been proved that the optimal solution $\mathbf{Z}^*$ to the convex stochastic composite optimization (SCO) problem

$$\min_{\mathbf{Z}\in\mathbb{R}^{n\times n}} \frac{1}{2}\mathbb{E}[||\mathbf{Z} - \boldsymbol{\xi}||_F^2] + \lambda||\mathbf{Z}||_*$$

is a low rank approximation to $\mathbf{K}$ (Zhang et al., 2016). $\mathbf{Z}^*$ can be written as $\mathbf{Z}^* = \mathcal{D}_\lambda[\mathbf{K}] = \sum_{i:\lambda_i>\lambda}(\lambda_i - \lambda)\mathbf{u}_i\mathbf{u}_i^\top$ where $\mathcal{D}_\lambda$ is the singular value shrinkage (SVS) operator (Cai et al., 2010) with a threshold λ, and $(\mathbf{u}_i, \lambda_i)$ is the i-th eigenpair. As long as the rank of $\mathbf{Z}^*$ is larger than s, the top-s eigenpairs of $\mathbf{K}$ can be determined by performing EVD on $\mathbf{Z}^*$

However, the SCO problem needs to be solved in a distributed manner. To this end, a distributed stochastic proximal gradient descent (DSPGD) algorithm is developed. DSPGD inherits the update rule of stochastic proximal gradient descent (SPGD) (Zhang et al., 2016). More specifically, the $(t+1)$-th iteration of $\mathbf{Z}^*$, i.e., $\mathbf{Z}_{t+1}$ is determined at the cloud server via $\mathbf{Z}_{t+1} = \mathcal{D}_{\eta_t\lambda}[(1-\eta_t)\mathbf{Z}_t + \eta_t\boldsymbol{\xi}_t]$ where η_t is a step size of DSPGD, and $\mathcal{D}_{\eta_t\lambda}$ is the SVS operator with a threshold $\eta_t\lambda$. The SVS

operation can be completed by computing the eigenpairs of $(1-\eta_t)\mathbf{Z}_t + \eta_t \boldsymbol{\xi}_t$ (for rest of paper the weighted sum of $\mathbf{Z}_t$ and $\boldsymbol{\xi}_t$ is denoted as $\mathbf{R}_t$). However, it is not feasible to performing SVD over $\mathbf{R}_t$ since $\boldsymbol{\xi}_t$ is not known at the cloud server. Since $\mathbf{R}_t$ is a symmetric matrix, and $\boldsymbol{\xi}_t$ is distributed over users' devices, the eigenpairs of $\mathbf{R}_t$ can be determined via the distributed Lanczos algorithm (DLA) (Penna & Stańczak, 2014).

The major idea of DLA can be summarized as follows. Given a randomly initialized vector $\mathbf{c}_1$, a vector $\mathbf{g}_1 = \mathbf{R}\mathbf{c}_1$ is computed via distributed computation. The vector $\mathbf{g}_1$ is then transformed to a vector $\mathbf{c}_2$ that is orthogonal to $\mathbf{c}_1$ at the cloud server following the Lanczos iteration (the details about Lanczos iteration and DLA are provided in Appendix A). After the q-th iterations, an orthogonal matrix $\mathbf{C}_q = [\mathbf{c}_1, ..., \mathbf{c}_q]$ and a matrix $\mathbf{T}_q = \mathbf{C}_q^\top \mathbf{R}\mathbf{C}_q$ are determined at the cloud server, where $\mathbf{C}_q$ is an orthogonal basis for a Krylov subspace $\mathcal{K}_q(\mathbf{R}, \mathbf{c}_1) = \text{span}[\mathbf{c}_1, \mathbf{R}\mathbf{c}_1, ..., \mathbf{R}^{q-1}\mathbf{c}_1]$. Let $\mathbf{T}_q = \mathbf{P}_q \boldsymbol{\Sigma}_q \mathbf{P}_q^\top$ be the eigendecomposition of $\mathbf{T}_q$. It has been proved that the columns of $\mathbf{C}_q\mathbf{P}_q$ and the diagonal entries of $\boldsymbol{\Sigma}_q$ can converge to the eigenvectors and eigenvalues of $\mathbf{R}$, respectively (Demmel, 1997). As mentioned before, in the q-th iteration of DLA, $\mathbf{g}_q = \mathbf{R}\mathbf{c}_q$ is computed in a distributed manner as follows. The equation for $\mathbf{g}_q$ can be rewritten as

$$\mathbf{g}_q = (1-\eta_t)\mathbf{Z}_t\mathbf{c}_q + \eta_t \mathbf{A}_t\mathbf{A}_t^\top \mathbf{c}_q / D, \tag{2}$$

where $\mathbf{Z}_t\mathbf{c}_q$ can be computed at the cloud server, and $\mathbf{A}_t\mathbf{A}_t^\top\mathbf{c}_q$ needs to be computed in a distributed manner since $\mathbf{A}_t = [\mathbf{A}_t[1]^\top, \ldots, \mathbf{A}_t[M]^\top]^\top$ are distributed over M users' devices. To compute $\mathbf{A}_t\mathbf{A}_t^\top\mathbf{c}_q$, the vector $\mathbf{c}_q = [\mathbf{c}_q[1]^\top, \ldots, \mathbf{c}_q[M]^\top]^\top$ is first partitioned into M parts at the cloud server, and the m-th part $\mathbf{c}_q[m]$ is sent to the m-th user's device. A local vector $\mathbf{A}_t[m]^\top\mathbf{c}_q[m]$ is then computed at the m-th user's device. These local vectors from M users' devices are summed up at the cloud to obtain a vector $\mathbf{A}_t^\top\mathbf{c}_q$. $\mathbf{A}_t^\top\mathbf{c}_q$ is then broadcast to M users' devices, and a vector $\mathbf{A}_t[m]\mathbf{A}_t^\top\mathbf{c}_q$ is computed at the m-th user's device. The M vectors $\{\mathbf{A}_t[m]\mathbf{A}_t^\top\mathbf{c}_q, m = 1, ..., M\}$ are sent back to the cloud server where they are concatenated to form $\mathbf{A}_t\mathbf{A}_t^\top\mathbf{c}_q$. After sufficient iterations, the eigenpairs of $\mathbf{R}_t$, i.e., $\{(\tilde{\mathbf{u}}_{i,t}, \tilde{\lambda}_{i,t}), i = 1, ...s_t\}$ can be obtained at the cloud server, and thus $\mathbf{Z}_{t+1}$ can be determined via $\mathbf{Z}_{t+1} = \sum_{i:\tilde{\lambda}_{i,t} > \eta_t\lambda} (\tilde{\lambda}_{i,t} - \eta_t\lambda)\tilde{\mathbf{u}}_{i,t}\tilde{\mathbf{u}}_{i,t}^\top$. As a result, the approximate solutions to the eigenpairs of $\mathbf{K}$ are finally obtained via DSPGD.

However, DSPGD is not communication efficient. By defining the communication cost as the number of floating-point numbers uploaded from users' devices to the cloud server, the communication cost of DSPGD is related to the number of data samples N since the dimension of $\mathbf{A}_t\mathbf{A}_t^\top\mathbf{c}_q$ equals N. To this end, a communication efficient mechanism (CEM) is designed to reduce the communication cost.

4.2 Communication Efficient Mechanism

In CEM, two properties of $\mathbf{R}_t$, i.e., positive semidefinite (p.s.d.) and low rank, are utilized. Due to the p.s.d. property, a matrix $\mathbf{W}_t$ is designed such that $\mathbf{W}_t\mathbf{W}_t^\top = \mathbf{R}_t$. More specifically, let $\mathbf{Z}_t = \mathbf{U}_t\boldsymbol{\Lambda}_t\mathbf{U}_t^\top$ be the eigendecomposition of $\mathbf{Z}_t$ and $\mathbf{B}_t = \mathbf{U}_t\boldsymbol{\Lambda}_t^{\frac{1}{2}}$, then $\mathbf{W}_t$ can be written as $\mathbf{W}_t = [\sqrt{\frac{\eta_t}{D}}\mathbf{A}_t, \sqrt{1-\eta_t}\mathbf{B}_t]$. Note that the eigenvectors of $\mathbf{R}_t$ are exactly the left singular vectors of $\mathbf{W}_t$. A left singular vector of $\mathbf{W}_t$ can be constructed based on the corresponding right singular vector and $\mathbf{W}_t$. Moreover, the dimension of a right singular vector of $\mathbf{W}_t$ is $(r_t + D)$ where r_t is the rank of $\mathbf{Z}_t$ and D is the number of random features. Compared with N, i.e., the dimension of a left singular vector of $\mathbf{W}_t$, $(r_t + D)$ is usually much smaller due to the low rank property of $\mathbf{R}_t$. Thus, it is possible to reduce the communication cost of DSPGD by first computing the right singular vectors of $\mathbf{W}_t$, then constructing the left singular vectors based the right singular vectors and $\mathbf{W}_t$.

The details of CEM are as follows. Since the the right singular vectors of $\mathbf{W}_t$ are exactly the eigenvectors of $\mathbf{W}_t^\top\mathbf{W}_t$, the cloud server first determines these eigenvectors via DLA. Assume that the rows of $\mathbf{B}_t = [\mathbf{B}_t[1]^\top, \ldots, \mathbf{B}_t[M]^\top]^\top$ are distributed over M users' devices like $\mathbf{A}_t$. As a result, the rows of $\mathbf{W}_t = [\mathbf{W}_t[1]^\top, \ldots, \mathbf{W}_t[M]^\top]^\top$ are also distributed over M users' devices, and a submatrix of $\mathbf{W}_t$ can be constructed at the m-th user's device by $\mathbf{W}_t[m] = [\sqrt{\frac{\eta_t}{D}}\mathbf{A}_t[m], \sqrt{1-\eta_t}\mathbf{B}_t[m]]$.

Similar to (2), such a step in CEM can be written as

$$\mathbf{g}_q = \mathbf{W}_t^\top \mathbf{W}_t \mathbf{c}_q = \sum_{m=1}^{M} \mathbf{W}_t[m]^\top \mathbf{W}_t[m] \mathbf{c}_q,$$

where $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]\mathbf{c}_q$ can be determined at the m-th device and then uploaded to the cloud server. As all the eigenvalues larger than $\eta_t \lambda$ (e.g., the top-s_t eigenvalues) and the corresponding eigenvectors of $\mathbf{W}_t^\top \mathbf{W}_t$ are obtained at the cloud server, these eigenpairs are then broadcast to users' devices. It is proved that an eigenvector $\tilde{\mathbf{v}}_{i,t}$ of $\mathbf{W}_t^\top \mathbf{W}_t$ can be transformed into an eigenvector $\tilde{\mathbf{u}}_{i,t}$ of $\mathbf{W}_t \mathbf{W}_t^\top$ via $\tilde{\mathbf{u}}_{i,t} = \frac{1}{\sqrt{\tilde{\lambda}_{i,t}}} \mathbf{W}_t \tilde{\mathbf{v}}_{i,t}$, and $\mathbf{W}_t^\top \mathbf{W}_t$ has the same non-zero eigenvalues as $\mathbf{W}_t \mathbf{W}_t^\top$. More specifically, a vector $\tilde{\mathbf{u}}_{i,t}[m] = \frac{1}{\sqrt{\tilde{\lambda}_{i,t}}} \mathbf{W}_t[m] \tilde{\mathbf{v}}_{i,t}$ is determined at the m-th user's device, where $\tilde{\mathbf{u}}_{i,t}[m]$ is one part of the eigenvector $\tilde{\mathbf{u}}_{i,t}$, i.e., $\tilde{\mathbf{u}}_{i,t} = [\tilde{\mathbf{u}}_{i,t}[1]^\top, ..., \tilde{\mathbf{u}}_{i,t}[M]^\top]^\top$. According to the SVS operator $\mathcal{D}_{\eta_t \lambda}$, $\mathbf{Z}_{t+1}$ actually equals to $\sum_{i=1}^{s_t} (\tilde{\lambda}_{i,t} - \eta_t \lambda) \tilde{\mathbf{u}}_{i,t} \tilde{\mathbf{u}}_{i,t}^\top$.

The assumption that the rows of $\mathbf{B}_t$ are distributed over M users' devices is confirmed as follows. Since $\mathbf{Z}_{t+1} = \mathbf{B}_{t+1}\mathbf{B}_{t+1}^\top$ is the eigendecomposition of $\mathbf{Z}_{t+1}$, the matrix $\mathbf{B}_{t+1}$ can be represented as $\mathbf{B}_{t+1} = [\sqrt{\tilde{\lambda}_{1,t} - \eta_t \lambda} \tilde{\mathbf{u}}_{1,t}, ..., \sqrt{\tilde{\lambda}_{s_t,t} - \eta_t \lambda} \tilde{\mathbf{u}}_{s_t,t}]$. Note that the vectors $\{\tilde{\mathbf{u}}_{i,t}[m], i = 1, ..., s_t\}$ are determined at the m-th user's device. Based on these vectors and the eigenvalues $\{\tilde{\lambda}_{i,t}, i = 1, ..., s_t\}$, the submatrix $\mathbf{B}_{t+1}[m]$ can also be determined at the m-th user's device via $\mathbf{B}_{t+1}[m] = [\sqrt{\tilde{\lambda}_{1,t} - \eta_t \lambda} \tilde{\mathbf{u}}_{1,t}[m], ..., \sqrt{\tilde{\lambda}_{i-1,t} - \eta_t \lambda} \tilde{\mathbf{u}}_{i-1,t}[m]]$. Thus, the assumption on the rows of $\mathbf{B}_t$ is confirmed to be true.

As DSPGD with CEM converges, an approximation $\hat{\mathbf{H}} = [\hat{\mathbf{H}}[1]^\top, \ldots, \hat{\mathbf{H}}[M]^\top]^\top$ of $\mathbf{H}$ is determined, where $\mathbf{K} = \mathbf{H}\mathbf{H}^\top$, and $\hat{\mathbf{H}}[m]^\top$ is held by the m-th user's device. A distributed linear k-means algorithm (Balcan et al., 2013) is then applied to the rows of $\hat{\mathbf{H}}$ to obtain the clustering result. DSPGD with CEM and the distributed linear k-means algorithm constitute FK k-means, and the pseudo code of the overall algorithm is provided in Appendix B.

5 Theoretical Analysis

5.1 Convergence Analysis for DSPDG

The convergence rate of DSPGD with CEM is derived in Theorem 1.

Theorem 1. *Assume $||\boldsymbol{\xi}||_F \leq L$. By setting $\eta_t = 2/t$, the following upper bound of $||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2$ holds with a probability at least $1 - \delta$*

$$||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2 \leq \frac{8}{T} \left[\lambda L \max_{t \in [T]} \sqrt{r_t} + L^2 \left(8 + 6 \log \frac{\lceil 2 \log_2 T \rceil}{\delta} \right) \right] = O(1/T), \tag{3}$$

where r_t is the rank of $\mathbf{Z}_t$.

The result in Theorem 1 indicates that DSPGD with CEM converges to $\mathbf{Z}^*$ at an $O(1/T)$ rate that is the same with that of SPGD developed in (Zhang et al., 2016). The proof of Theorem 1 is provided in Appendix C.

5.2 Communication Cost Analysis for Communication Efficient Mechanism

In Theorem 2, the communication cost of DSPGD with CEM and the communication cost of DSPGD without CEM are both analyzed.

Theorem 2. *For DSPGD with CEM, in the t-th iteration, its communication cost is linear to $r_t + D$ where r_t is the rank of $\mathbf{Z}_t$ and D is the number of random features. Besides, the ratio of the communication cost of DSPGD without CEM to that of DSPGD with CEM is $\frac{(N+MD)Q_0}{M(r_t+D)Q_1}$ in the t-th iteration, where N is the number of data samples; M is the number of users; Q_0 and Q_1 are the number of Lanczos iterations for DSPGD without CEM and DSPGD with CEM, respectively.*

By Theorem 2, the communication cost of DSPGD with CEM is unrelated to the number of data samples N, and the communication cost reduced by CEM can be revealed by the ratio. The values of Q_0 and Q_1 are affected by the selection of initial vector $\mathbf{c}_1$. However, empirically the values of Q_0 and Q_1 are at the same order no matter which initial vectors are chosen. The dominant factor of the ratio is still $\frac{N+MD}{M(r_t+D)}$. Since $\mathbf{Z}_t$ is used to approach a low rank approximation of $\mathbf{K}$, empirically its rank r_t has an upper bound. In our experiments, the value of r_t is at the same order of s, i.e. the number of eigenvectors of $\mathbf{K}$ to be determined by DSPGD. Usually, the number of data samples at a user's device is much larger than s so that it is easy to satisfied that $N > Mr_t$, and CEM can definitely reduce the communication cost for DSPGD in these cases. The proof of Theorem 2 and the empirical results for r_t are given in Appendix D.

5.3 Clustering Loss Analysis for Federated Kernel k-means

Before the analysis, a γ-approximate algorithm of linear k-means is first defined as follows.

Definition 1. *A linear k-means algorithm is applied to a matrix $\mathbf{H}$ with n row, where an indicator matrix $\tilde{\mathbf{Y}}$ is obtained. This algorithm is called a γ-approximate algorithm if, for any matrix $\mathbf{H}$, $f(\tilde{\mathbf{Y}};\mathbf{H}) \leq \gamma \min_{\mathbf{Y}} f(\mathbf{Y};\mathbf{H})$ where f is the objective function of linear k-means.*

A upper bound of clustering loss is then derived in Theorem 3 for FK k-means.

Theorem 3. *The objective function of kernel k-means in (1) is denoted as f_K. Assume $\mathbf{H}_t$ is the output of DSPGD with CEM after T iterations, and a γ-approximate algorithm is applied to the first s columns of $\mathbf{H}_t$ to obtain an indicator matrix $\widetilde{\mathbf{Y}}_T$. For $\widetilde{\mathbf{Y}}_T$, the following inequality holds with a probability at least $1-\delta(T)$, i.e., $f_K(\widetilde{\mathbf{Y}}_T) \leq \gamma(1+\varepsilon+\frac{k}{s})\min_{\mathbf{Y}} f_K(\mathbf{Y})$, where $\varepsilon = O(\sqrt{\frac{s}{T}})$.*

Note that as T increases, $\delta(T)$ decreases, and the error ratio $\gamma(1+\varepsilon+\frac{k}{s})$ can approach $\gamma(1+\frac{k}{s})$ that is the error ratio of the scalable kernel k-means conducted at the cloud server (Wang et al., 2019). The proof of Theorem 3 is given in Appendix E.

5.4 Privacy Analysis for Federated Kernel k-means

FK k-means can provide two levels of privacy preservation: 1) one user's local data are not exposed to the cloud server and other users; 2) the cloud server cannot recover users' local data from the collected local computational results via matrix operation. The first level is a basic requirement of federated learning. The second level can be proved by Theorem 4.

Theorem 4. *Based on the collected local computational results, the cloud server can at most recover the matrices $\{\mathbf{W}_t[m]^\top \mathbf{W}_t[m], m = 1, ..., M\}$ via matrix operations. Moreover, recovering the random feature matrix $\mathbf{A}_t$ from such matrices is an ill-posed problem with infinite solutions.*

By Theorem 4, the cloud server cannot recover the random feature matrices from the local computational results. Without such random feature matrices, it is infeasible for the cloud server to recover users' local data via matrix operations. More explanation and the proof of Theorem 4 are provided in Appendix F. Moreover, FK k-means can incorporate the differential privacy mechanism (Dwork et al., 2006; Su et al., 2016) or random perturbation (Lin, 2016) to provide higher level of privacy preservation, which is subject to future work.

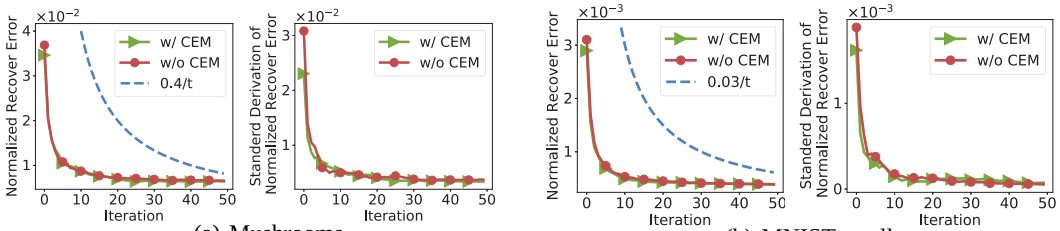
6 Experiments

6.1 Experimental Setting

Four types of existing schemes are considered in the experiments: centralized kernel k-means (Zha et al., 2001) (denoted as CK k-means), scalable kernel k-means (Wang et al., 2019) (denoted as SK k-means), distributed kernel k-means with random feature (Chitta et al., 2012) (denoted as RFK k-means), and communication efficient distributed kernel PCA (Balcan et al., 2016) (denoted as CE PCA). CK k-means and SK k-means are regarded as the centralized methods, and the rest methods are considered as the distributed methods. One constraint on these distributed methods is that users' raw data cannot be uploaded to the cloud server. Besides, Gaussian kernel is used in each algorithm. Four datasets are selected for performance evaluation: Three public datasets (Mushrooms,

Table 1: Datasets statistics and hyperparameter settings for FK k-means

Dataset	#samples (N)	#features (d)	#clusters (k)	#random features (D)	#eigenvectors (s)
Mushrooms	$8,124$	112	2	15	2
MNIST-small	$20,000$	780	10	200	10
MNIST	$60,000$	780	10	200	12
Covtype	$581,012$	54	7	30	10
Smartphone	$177,029$	12	4	20	6



(a) Mushrooms (b) MNIST-small

Figure 1: The convergence curves of the two versions of DSPGD and standard deviation curves of the normalized recover error on the Mushrooms dataset and the MNIST-small dataset.

MNIST, and Covtype) from the LIBSVM dataset repository and one dataset (Smartphone) provided by a company. In addition, $20,000$ data samples are randomly selected from the dataset MNIST to construct a dataset MNIST-small that is used to validate the convergence of DSPGD. The statistical information of these datasets is given in Table 1. The description of the Smartphone dataset and the existing methods are included in Appendix G.

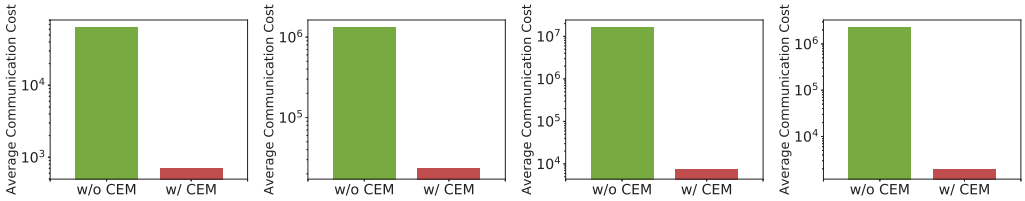
The hyperparameters of FK k-means are determined as follows. The kernel parameter γ is computed based on the average interpoint distance in the given dataset (Wang et al., 2019): $\gamma = \frac{N^2}{2\sum_{i=1}^{N}\sum_{j=1}^{N}\|\mathbf{x}_i-\mathbf{x}_j\|_2^2}$. The threshold parameter λ of DSPGD is set to the $(k+2)$-th eigenvalue obtained in the first iteration of DSPGD. The configuration of the number of random features D and the parameter s in the top-s eigenvectors for each dataset is provided in Table 1 (the hyperparameter configuration of the existing methods and a discussion on the configuration of D are provided in Appendix G). In the experiments, $M = 5$ worker processes and one coordinator process are generated to simulate users' devices and the cloud server, respectively. The worker processes communicate with the coordinator process via the message passing interface (MPI) in a synchronized manner. All the experiments are executed in a server with one i7-6850k CPU and 32 GB RAM.

6.2 Experimental Results

The convergence of DSPGD is validated over two datasets, Mushrooms and MNIST-small, whose low rank matrices $\hat{\mathbf{K}} = \sum_{i=1}^{s}\lambda_i\mathbf{u}_i\mathbf{u}_i^\top$ can be computed by performing SVD on their kernel matrices $\mathbf{K}$. A normalized recover error $\frac{\|\mathbf{K}_t-\hat{\mathbf{K}}\|_F^2}{N^2}$ (Zhang et al., 2016) is recorded for each iteration t of DSPGD, where $\mathbf{K}_t = \sum_{i=1}^{s}(\tilde{\lambda}_{i,t} + (1-\eta_t)\lambda)\tilde{\mathbf{u}}_{i,t}\tilde{\mathbf{u}}_{i,t}^\top$ is the estimation of $\hat{\mathbf{K}}$ at iteration t. In the left subfigure of Figure 1(a) and that of Figure 1(b), the convergence curves of DSPGD are lower than the curve of $0.4/t$ and the curve of $0.03/t$, respectively, which verifies DSPGD converges at an $O(1/t)$ rate. In Figure 1(a) and 1(b), the curves of two version of DSPGD nearly overlap, indicating that CEM has little impact on the convergence of DSPGD.

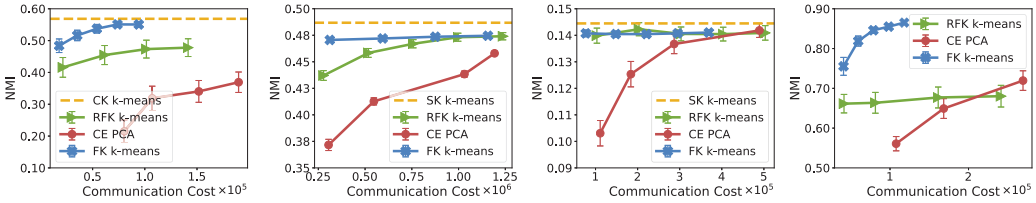
The average communication cost per iteration of DSPGD with CEM and that of DSPGD without CEM are compared in Figure 2 to evaluate the effectiveness of CEM. The log-scale is used for the y-axis of each subfigure, and the unit of the y-axis is the number of the floating-point numbers. According to the results in the four subfigures, it is shown that CEM can reduce communication cost of DSPGD by more than 95%.

In order to evaluate the clustering quality and the communication cost of FK k-means, curves of average normalized mutual information (NMI) (Strehl & Ghosh, 2002) versus average communication cost are plotted in Figure 3 for FK k-means and the existing schemes. The error bars in



(a) Mushroom (b) MNIST (c) Covtype (d) Smartphone

Figure 2: The average communication cost per iteration of the two versions of DSPGD.



(a) Mushrooms (b) MNIST (c) Covtype (d) Smartphone

Figure 3: The NMI score versus the average communication cost of DSPGD and the existing methods on the four datasets.

Figure 3 are the 95% confidence interval of the average NMI scores. For the three public datasets, FK k-means can achieve comparable average NMI scores to that of the centralized methods (CK k-means for Mushroom dataset and SK k-means for MNIST dataset and covtype dataset). For a centralized method, its communication cost equals to the volume of a dataset, which is too large to be shown. Thus, in Figure 3(a), 3(b), and 3(c) the dash line only represents the average NMI scores rather than the relationship of average NMI scores versus the corresponding communication cost. In Figure 3(a) and 3(b), given a fixed communication cost, FK k-means can achieve the highest average NMI scores among the three distributed algorithms. Besides, in 3(b), FK k-means nearly achieves the upper bound of the average NMI score with a low communication cost. To reach such a NMI score, FK k-means reduces the communication cost by more than 60% compared with RFK k-means. In Figure 3(c), FK k-means has a similar performance to that of RFK k-means. Compared with CE PCA, FK k-means reduces the communication cost by more than 60% when the highest average NMI score is considered. For the Smartphone dataset, since it has no labels, the cluster quality of a given clustering algorithm is evaluated by measuring the similarity between the clustering results of the algorithm and that of SK k-means. To this end, the clustering results of SK k-means are used as the labels to compute NMI scores. It is shown in Figure 3(d) that FK k-means has a much higher upper bound for the average NMI score (close to 0.9) than that of RFK k-means and CE PCA.

7 Conclusion

In this paper, FK k-means was developed. In the algorithm, a distributed stochastic proximal gradient descent approach was first designed to determine the eigenpairs of the kernel matrix in a distributed manner. A communication efficient mechanism was then designed to reduce the communication cost. Besides communication efficient, FK k-means can also provide two levels of privacy preservation. In theoretical analysis, DSPGD with CEM was proved to converge at an $O(1/T)$ rate. The communication cost of DSPGD with CEM is unrelated to the number of data samples. The clustering loss of FK k-means can approach that of the centralized kernel k-means. The effectiveness of the FK k-means was validated by experiments on several real-world datasets. In the future work, an interesting topic is to incorporate differential privacy into FK k-means to provide higher level of privacy preservation.

REFERENCES

Maria Florina Balcan, Yingyu Liang, Le Song, David Woodruff, and Bo Xie. Communication efficient distributed kernel principal component analysis. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 725–734, 2016.

Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general topologies. In *Advances in Neural Information Processing Systems 26*, pp. 1995–2003, 2013.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

Sebastian Caldas, Virginia Smith, and Ameet Talwalkar. Federated kernelized multi-task learning. In *SysML Conference 2018*, 2018.

Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, 2016.

Radha Chitta, Rong Jin, Timothy C Havens, and Anil K Jain. Approximate kernel k-means: Solution to large scale kernel clustering. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 895–903, 2011.

Radha Chitta, Rong Jin, and Anil K Jain. Efficient kernel clustering using random fourier features. In *2012 IEEE 12th International Conference on Data Mining*, pp. 161–170, 2012.

Radha Chitta, Rong Jin, Timothy C Havens, and Anil K Jain. Scalable kernel clustering: Approximate kernel k-means. *arXiv preprint arXiv:1402.3849*, 2014.

Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pp. 163–172, 2015.

James W Demmel. *Applied numerical linear algebra*. SIAM, 1997.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 551–556, 2004.

Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pp. 606–610, 2005.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284, 2006.

Ahmed Elgohary, Ahmed K Farahat, Mohamed S Kamel, and Fakhri Karray. Embed and conquer: Scalable embeddings for kernel k-means on mapreduce. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 425–433, 2014.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020.

Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, pp. 583–591, 2012.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

Keng-Pei Lin. Privacy-preserving kernel k-means clustering outsourcing with random transformation. *Knowledge and Information Systems*, 49(3):885–908, 2016.

Sulin Liu, Sinno Jialin Pan, and Qirong Ho. Distributed multi-task relationship learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 937–946, 2017.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Federico Penna and Sławomir Stańczak. Decentralized eigenvalue algorithms for distributed signal detection in wireless networks. *IEEE Transactions on Signal Processing*, 63(2):427–440, 2014.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pp. 1177–1184, 2008.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30*, pp. 4424–4434, 2017.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the 6th ACM Conference on Data and Application Security and Privacy*, pp. 26–37, 2016.

Nikolaos Tsapanos, Anastasios Tefas, Nikolaos Nikolaidis, and Ioannis Pitas. A distributed framework for trimmed kernel k-means clustering. *Pattern recognition*, 48(8):2685–2698, 2015.

Shusen Wang, Alex Gittens, and Michael W Mahoney. Scalable kernel k-means clustering with Nyström approximation: relative-error bounds. *The Journal of Machine Learning Research*, 20 (1):431–479, 2019.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 7252–7261, 2019.

Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems 14*, pp. 1057–1064, 2001.

Lijun Zhang, Tianbao Yang, Jinfeng Yi, Rong Jin, and Zhi-Hua Zhou. Stochastic optimization for kernel PCA. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 2316–2322, 2016.

Algorithm 1 Lanczos Algorithm

1: **Input:** An symmetric matrix $\mathbf{R}$, an initial vector $\mathbf{c}_1$
2: **Output:** An approximation $\mathbf{P}_Q$ to the eigenvectors of $\mathbf{R}$, and an approximation $\boldsymbol{\sigma} = [\sigma_1, ..., \sigma_Q]$ to the eigenvalues of $\mathbf{R}$
3: Initialize $\beta_0 = 0$ and $\mathbf{c}_0 = \mathbf{0}$
4: **for** $q = 1, 2, \ldots, Q$ **do**
5: $\quad \mathbf{g} = \mathbf{R}\mathbf{c}_q$
6: $\quad \alpha_q = \mathbf{c}_q^\top \mathbf{g}$
7: $\quad \mathbf{g} = \mathbf{g} - \alpha_q \mathbf{c}_q - \beta_{q-1}\mathbf{c}_{q-1}$
8: $\quad \beta_q = ||\mathbf{g}||_2$
9: $\quad$ **if** $\beta_q = 0$ **then**
10: $\quad\quad$ **break**
11: $\quad$ **end if**
12: $\quad \mathbf{c}_{q+1} = \mathbf{g}/\beta_q$
13: $\quad$ Construct a symmetric tridiagonal matrix $\mathbf{T}_Q$
14: $\quad$ Perform EVD on $\mathbf{T}_Q$ to obtain its eigenvectors $\mathbf{P}_Q$ and its eigenvalues $\boldsymbol{\sigma} = [\sigma_1, ..., \sigma_Q]$
15: $\quad$ Compute $\mathbf{C}_Q\mathbf{P}_Q$
16: **end for**

A Details about distributed Lanczos algorithm

To find the eigenpairs of a symmetric matrix $\mathbf{R}$, the Lanczos algorithm (LA) (Lanczos, 1950) first build a Krylov subspace $\mathcal{K}_q(\mathbf{R}, \mathbf{c}_1) = \text{span}[\mathbf{c}_1, \mathbf{R}\mathbf{c}_1, ..., \mathbf{R}^{q-1}\mathbf{c}_1]$ where $\mathbf{c}_1$ is an initial vector, and then it employs the Rayliegh-Ritz procedure to construct the best approximate eigenpairs for $\mathbf{R}$ in the Krylov subspace. In the first step, LA constructs an orthogonal basis of the Krylov subspace following the procedure of line 5 to line 12 in Algorithm 1. Meanwhile, a symmetric tridiagonal matrix $\mathbf{T}_Q = \mathbf{C}_Q^\top \mathbf{R} \mathbf{C}_Q$ can be explicitly constructed with $\boldsymbol{\alpha}_Q$ and $\boldsymbol{\beta}_Q$ via

$$\mathbf{T}_Q = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{Q-1} \\ & & \beta_{Q-1} & \alpha_Q \end{bmatrix}.$$

Based on $\mathbf{T}_Q$, the Rayliegh-Ritz procedure can be utilized to approximate the eigenpairs of $\mathbf{R}$. Let $\mathbf{T}_Q = \mathbf{P}_Q \boldsymbol{\Sigma}_Q \mathbf{P}_Q^\top$ be the eigendecomposition of $\mathbf{T}_Q$. It has been proved that the columns of $\mathbf{C}_Q\mathbf{P}_Q$ and the diagonal entries of $\boldsymbol{\Sigma}_Q$ are the optimal approximation to the eigenvectors and eigenvalues of $\mathbf{R}$, respectively (Demmel, 1997). Thus, in the Rayliegh-Ritz procedure, $\mathbf{P}_Q$ and $\boldsymbol{\Sigma}_Q$ are determined by performing EVD on $\mathbf{T}_Q$, and then $\mathbf{C}_Q\mathbf{P}_Q$ are computed as the approximation to the eigenvectors of $\mathbf{R}$. As the number of iteration Q increases, the columns of $\mathbf{C}_Q\mathbf{P}_Q$ and the diagonal entries of $\boldsymbol{\Sigma}_Q$ can converge to the eigenvectors and eigenvalues of $\mathbf{R}$, respectively (Demmel, 1997).

As for the distributed Lanczos algorithm (DLA), only the step of line 5 in Algorithm 1 is conducted in a distributed manner, and other steps are conducted at the cloud server. In our problem, if

$$\mathbf{R} = (1 - \eta_t)\mathbf{Z}_t + \eta_t \boldsymbol{\xi}_t = (1 - \eta_t)\mathbf{Z}_t + \frac{\eta_t}{D}\mathbf{A}_t\mathbf{A}_t^\top,$$

where $\mathbf{Z}_t$ and $\mathbf{c}_q$ are known at the cloud server, and $\mathbf{A}_t = [\mathbf{A}_t[1]^\top, \ldots, \mathbf{A}_t[M]^\top]^\top$ are distributed over M users' devices, then $(1 - \eta_t)\mathbf{Z}_t\mathbf{c}_q$ is computed at the cloud server and $\eta_t \mathbf{A}_t\mathbf{A}_t^\top \mathbf{c}_q/D$ is computed in a distributed manner as follows. The vector $\mathbf{c}_q = [\mathbf{c}_q[1]^\top, \ldots, \mathbf{c}_q[M]^\top]^\top$ is first partitioned into M parts at the cloud server, and the m-th part $\mathbf{c}_q[m]$ is sent to the m-th user's device. A local vector $\mathbf{A}_t[m]^\top \mathbf{c}_q[m]$ is then computed at the m-th user's device. These local vectors from M users' devices are summed up at the cloud to obtain a vector $\mathbf{A}_t^\top \mathbf{c}_q$. $\mathbf{A}_t^\top \mathbf{c}_q$ is then broadcast to M users' devices, and a vector $\mathbf{A}_t[m]\mathbf{A}_t^\top \mathbf{c}_q$ is computed at the m-th user's device. These M vectors are sent back to the cloud server where they are concatenated to form $\mathbf{A}_t\mathbf{A}_t^\top \mathbf{c}_q$. If

$$\mathbf{R} = \mathbf{W}_t^\top \mathbf{W}_t = \sum_{m=1}^{M} \mathbf{W}_t[m]^\top \mathbf{W}_t[m],$$

where $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]$ can be computed at the m-th user's device, then each user' devices first determines $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]\mathbf{c}_q$ locally, and these M vectors are then uploaded to the cloud server where they are summed up to form $\mathbf{W}_t^\top \mathbf{W}_t \mathbf{c}_q$.

In the t-th iteration of DSPGD, DLA is used to compute the eigenvalues larger than $\eta_t \lambda$ of the $\mathbf{R}_t$. Thus, in practice, the convergence criterion of DLA is that all the approximated eigenvalues larger than $\eta_t \lambda$ converge, rather than that the number of iteration reaches its maximal value Q.

One issue of LA and DLA in practice is that they can only be conducted in floating point arithmetic, which can destroy the orthogonality of the columns in $\mathbf{C}_q$, and further affect the convergence of DSPGD. To this end, a full reorthogonalization method (Demmel, 1997) is utilized to guarantee that $\mathbf{C}_q$ is an orthogonal matrix with a high probability. The key idea of this method to generate a new vector $\mathbf{c}_q$ from a subspace that is orthogonal to all the previous vectors $\{\mathbf{c}_1, ..., \mathbf{c}_{q-1}\}$, which can be accomplished by replacing line 7 in Algorithm 1 with

$$\mathbf{g} = \mathbf{g} - \sum_{i=1}^{q-1} \mathbf{g}^\top \mathbf{c}_i \mathbf{c}_i. \tag{4}$$

The operation in (4) can be called multiple times in one iteration of LA to increase the probability that $\mathbf{C}_q$ is an orthogonal matrix. In the implementation of federated kernel k-means, such operation is called twice in each iteration of DLA. Note that the full reorthogonalization only requires more flops at the cloud server, which does not affect the algorithm complexity at users' devices.

B Pseudo Code for Federated Kernel k-Means

Algorithm 2 Federated Kernel k-Means Algorithm

Input: The threshold parameter λ, the number of eigenvectors s used for the clustering, local datasets $\mathcal{L}_m, m = 1, ..., M$, the number of random features D, the maximal number of iterations T
Output: the clustering result for each data sample
Initialize $\mathbf{B}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
    Initialize $\eta_t = 1/t$, $q = 0$ ▷ t, q are global variables
    for $m = 1, 2, ..., M$ **do** ▷ Conducted at M users' devices in parallel
        Compute $\mathbf{A}_t[m]$ by applying a random feature method to $\mathcal{L}_m$
        Construct $\mathbf{W}_t[m] = [\sqrt{\frac{\eta_t}{D}}\mathbf{A}_t[m], \sqrt{1-\eta_t}\mathbf{B}_t[m]]$
    end for
    Call DLA to determine the eigenvalues $\{\tilde{\lambda}_{i,t}, i = 1, ..., s_t\}$ larger than $\eta_t \lambda$ and the corresponding eigenvectors $\{\tilde{\mathbf{v}}_{1,t}, ..., \tilde{\mathbf{v}}_{s_t,t}\}$ of $\mathbf{W}_t^\top \mathbf{W}_t$ ▷ s_t denotes the number of obtained eigenvectors
    Broadcast these s_t eigenvectors and eigenvalues from the cloud server to M users' devices
    for $m = 1, 2, ..., M$ **do** ▷ Conducted at M users' devices in parallel
        Receive the eigenvectors and eigenvalues to determine $\{\tilde{\mathbf{u}}_{1,t}[m], ..., \tilde{\mathbf{u}}_{s_t,t}[m]\}$ via $\tilde{\mathbf{u}}_{i,t}[m] = \frac{1}{\sqrt{\tilde{\lambda}_{i,t}}}\mathbf{W}_t[m]\tilde{\mathbf{v}}_{i,t}$
        Compute $\mathbf{B}_{t+1}[m] = [\sqrt{\tilde{\lambda}_{1,t} - \eta_t\lambda}\tilde{\mathbf{u}}_{1,t}[m], ..., \sqrt{\tilde{\lambda}_{s_t,t} - \eta_t\lambda}\tilde{\mathbf{u}}_{s_t,t}[m]]$
    end for
end for
for $m = 1, 2, ..., M$ **do** ▷ Conducted at M users' devices in parallel
    Select s vectors $\{\tilde{\mathbf{u}}_{1,T}[m], ..., \tilde{\mathbf{u}}_{s,T}[m]\}$ to determine a matrix

$$\mathbf{H}[m] = [\sqrt{\tilde{\lambda}_{1,T} + (1-\eta_T)\lambda}\tilde{\mathbf{u}}_{1,T}[m], ..., \sqrt{\tilde{\lambda}_{s,T} + (1-\eta_T)\lambda}\tilde{\mathbf{u}}_{s,T}[m]]$$

end for
Apply a distributed linear k-means algorithm over the rows of $\mathbf{H} = [\mathbf{H}[1]^\top, ..., \mathbf{H}[M]^\top]^\top$ to obtain an indicator matrix $\widetilde{\mathbf{Y}}$

The pseudo code for federated kernel k-means (including CEM) is shown in Algorithm 2. For the simplicity of the pseudo code, the threshold parameter λ is directly set as an input parameter. In practice, λ is set to the $(k+2)$-th eigenvalue obtained in the first iteration of DSPGD. The distributed linear k-means algorithm used in Algorithm 2 is proposed in (Balcan et al., 2013). This algorithm first constructs a coreset of the given dataset and computes a weight for each samples in the coreset in a distributed manner, and then conduct a weighted linear k-means over coreset at the cloud server to determine the centroid for each cluster. These centroids are finally used to generate the clustering result for all the data samples.

C Proof of Theorem 1

This proof partially follows the proof of Theorem 1 in Zhang et al. (2016). The difference is that the t-th iteration of $\mathbf{Z}^*$ obtained by DSPGD, i.e., $\mathbf{Z}_t$, may not equal $\mathbf{Z}_t^* = \mathcal{D}_{\eta_t\lambda}[(1-\eta_t)\mathbf{Z}_t + \eta_t\boldsymbol{\xi}_t]$. The gap between $\mathbf{Z}_t$ and $\mathbf{Z}_t^*$ is caused by that the distributed Lanczos algorithm (DLA) only approximates the eigenpairs of a target matrix. Thus, in this proof, it is assumed that

$$||\mathbf{Z}_t - \mathbf{Z}_t^*||_F^2 \leq \epsilon, \quad \forall t, \tag{5}$$

when DLA reaches its convergence criterion, where $\epsilon \ll 1$.

Before the proof, we first define

$$\begin{aligned} F(\mathbf{Z}) =& \frac{1}{2}\mathbb{E}[||\mathbf{Z} - \boldsymbol{\xi}||_F^2], \\ f_t(\mathbf{Z}) =& \frac{1}{2}||\mathbf{Z} - \boldsymbol{\xi}_t||_F^2. \end{aligned}$$

For a μ-strongly convex function $l(\mathbf{Z})$, if $l(\mathbf{Z}_1) \geq l(\mathbf{Z}_2)$, then

$$l(\mathbf{Z}_1) - l(\mathbf{Z}_2) \geq \frac{\mu}{2}||\mathbf{Z}_1 - \mathbf{Z}_2||_F^2. \tag{6}$$

In the $t+1$-th iteration of DSPGD, the goal is to determine the optimal solution $\mathbf{Z}_{t+1}^*$ to the following optimization problem

$$\min_{\mathbf{Z}\in\mathbb{R}^{n\times n}} \frac{1}{2}||\mathbf{Z} - \mathbf{Z}_t||_F^2 + \eta_t\langle \mathbf{Z} - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \eta\lambda||\mathbf{Z}||_*. \tag{7}$$

By DLA, an approximate solution $\mathbf{Z}_{t+1}$ that satisfies (5) can be obtained. The following lemma is a key step in this proof.

Lemma 1. *Before the convergence of DSPGD, the following inequality holds, i.e.,*

$$\begin{aligned} &\frac{1}{2}||\mathbf{Z}_{t+1} - \mathbf{Z}_t||_F^2 + \eta_t\langle \mathbf{Z}_{t+1} - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \eta\lambda||\mathbf{Z}_{t+1}||_* \\ \leq& \frac{1}{2}||\mathbf{Z}^* - \mathbf{Z}_t||_F^2 + \eta_t\langle \mathbf{Z}^* - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \eta\lambda||\mathbf{Z}^*||_*. \end{aligned} \tag{8}$$

Proof. The objective function in (7) can be rewritten as

$$\begin{aligned} &\frac{1}{2}||\mathbf{Z} - \mathbf{Z}_t||_F^2 + \eta_t\langle \mathbf{Z} - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \eta\lambda||\mathbf{Z}||_* \\ =& \frac{1}{2}||\mathbf{Z} - \mathbf{Z}_t||_F^2 + \eta_t\langle \mathbf{Z} - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \frac{\eta_t^2}{2}||\nabla f_t(\mathbf{Z}_t)||_F^2 - \frac{\eta_t^2}{2}||\nabla f_t(\mathbf{Z}_t)||_F^2 + \eta\lambda||\mathbf{Z}||_* \\ =& \frac{1}{2}||\mathbf{Z} - [(1-\eta_t)\mathbf{Z}_t + \eta_t\boldsymbol{\xi}]||_F^2 + \eta\lambda||\mathbf{Z}||_* - \frac{\eta_t^2}{2}||\nabla f_t(\mathbf{Z}_t)||_F^2. \end{aligned}$$

Since $\frac{\eta_t^2}{2}||\nabla f_t(\mathbf{Z}_t)||_F^2$ is a constant, we can only consider

$$l(\mathbf{Z}) = \frac{1}{2}||\mathbf{Z} - [(1-\eta_t)\mathbf{Z}_t + \eta_t\boldsymbol{\xi}]||_F^2 + \eta\lambda||\mathbf{Z}||_*$$

in the following part of the proof. Now we first assume that $l(\mathbf{Z}^*) \leq l(\mathbf{Z}_{t+1})$, then we have

$$l(\mathbf{Z}_{t+1}) - l(\mathbf{Z}_{t+1}^*) \geq l(\mathbf{Z}^*) - l(\mathbf{Z}_{t+1}^*) \geq \frac{\mu}{2}||\mathbf{Z}_{t+1}^* - \mathbf{Z}^*||_F^2. \tag{9}$$

Moreover, $l(\mathbf{Z}_{t+1}) - l(\mathbf{Z}^*_{t+1})$ can be expanded as

$$\begin{aligned}
&l(\mathbf{Z}_{t+1}) - l(\mathbf{Z}^*_{t+1}) \\
=&\frac{1}{2}||\mathbf{Z}_{t+1} - \mathbf{R}_t||_F^2 + \eta\lambda||\mathbf{Z}_{t+1}||_* - \frac{1}{2}||\mathbf{Z}^*_{t+1} - \mathbf{R}_t||_F^2 + \eta\lambda||\mathbf{Z}^*_{t+1}||_* \\
=&\frac{1}{2}(||\mathbf{Z}_{t+1} - \mathbf{R}_t||_F - ||\mathbf{Z}^*_{t+1} - \mathbf{R}_t||_F)(||\mathbf{Z}_{t+1} - \mathbf{R}_t||_F + ||\mathbf{Z}^*_{t+1} - \mathbf{R}||_F) \\
&+ \eta_t\lambda(||\mathbf{Z}_{t+1}||_* - ||\mathbf{Z}^*_{t+1}||_*) \\
\leq&\frac{1}{2}||\mathbf{Z}_{t+1} - \mathbf{Z}^*_{t+1}||_F(||\mathbf{Z}_{t+1} - \mathbf{Z}^*_{t+1}||_F + 2||\mathbf{Z}^*_{t+1} - \mathbf{R}||_F) + \eta_t\lambda||\mathbf{Z}_{t+1} - \mathbf{Z}^*_{t+1}||_*
\end{aligned} \tag{10}$$

It is well known that given a matrix $\mathbf{M}$ the following inequality holds for its nuclear norm and its Frobenius norm, i.e., $||\mathbf{M}||_*^2 \leq \mathrm{rank}(\mathbf{M})||\mathbf{M}||_F^2$. By this inequality, we have

$$||\mathbf{Z}_{t+1} - \mathbf{Z}^*_{t+1}||_* \leq \sqrt{r}||\mathbf{Z}_{t+1} - \mathbf{Z}^*_{t+1}||_F \leq \sqrt{r\epsilon}, \tag{11}$$

where r is the rank of $(\mathbf{Z}_{t+1} - \mathbf{Z}^*_{t+1})$. Substitute (5) and (11) into (10), we have

$$l(\mathbf{Z}_{t+1}) - l(\mathbf{Z}^*_{t+1}) \leq \frac{1}{2}\epsilon^2 + \epsilon||\mathbf{Z}^*_{t+1} - \mathbf{R}_t||_F + \eta_t\lambda\sqrt{r\epsilon}.$$

Since $||\mathbf{Z}^*_{t+1} - \mathbf{R}_t||_F$ is a constant, this upper bound of $l(\mathbf{Z}_{t+1}) - l(\mathbf{Z}^*_{t+1})$ can become arbitrarily small if ϵ is arbitrarily small. Hence, according to (9), $||\mathbf{Z}^*_{t+1} - \mathbf{Z}^*||_F^2$ can also be arbitrarily small. However, this contradicts that $||\mathbf{Z}^*_{t+1} - \mathbf{Z}^*||_F^2$ cannot become arbitrarily small before the convergence of DSPGD. Therefore, before the convergence of DSPGD, $l(\mathbf{Z}^*) \geq l(\mathbf{Z}_{t+1})$ is satisfied. □

The rest part then follows the proof of Theorem 1 in Zhang et al. (2016). Based on Lemma 1 and the property of strongly convex function in (6), the update rule of SPDG implies

$$\begin{aligned}
&\frac{1}{2}||\mathbf{Z}_{t+1} - \mathbf{Z}_t||_F^2 + \eta_t\langle\mathbf{Z}_{t+1} - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \eta\lambda||\mathbf{Z}_{t+1}||_* \\
\leq&\frac{1}{2}||\mathbf{Z}^* - \mathbf{Z}_t||_F^2 + \eta_t\langle\mathbf{Z}^* - \mathbf{Z}_t, \nabla f_t(\mathbf{Z}_t)\rangle + \eta_t\lambda||\mathbf{Z}^*||_* - \frac{1}{2}||\mathbf{Z}^* - \mathbf{Z}_{t+1}||_F^2.
\end{aligned} \tag{12}$$

Since $F(\mathbf{Z})$ is 1-strongly convex, it can be shown that

$$\begin{aligned}
&\frac{1}{2}||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 \\
\leq&F(\mathbf{Z}_t) + \lambda||\mathbf{Z}_t||_* - F(\mathbf{Z}^*) - \lambda||\mathbf{Z}^*||_* \\
\leq&\langle\mathbf{Z}_t - \mathbf{Z}^*, \nabla F(\mathbf{Z}_t)\rangle - \frac{1}{2}||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + \lambda||\mathbf{Z}_t||_* - \lambda||\mathbf{Z}^*||_* \\
=&\langle\mathbf{Z}_t - \mathbf{Z}^*, \nabla f_t(\mathbf{Z}_t)\rangle - \lambda||\mathbf{Z}^*||_* - \frac{1}{2\eta_t}||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 \\
&+ \lambda||\mathbf{Z}_t||_* - \frac{1}{2}||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + \frac{1}{2\eta_t}||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + \langle\nabla F(\mathbf{Z}_t) - \nabla f_t(\mathbf{Z}_t), \mathbf{Z}_t - \mathbf{Z}^*\rangle \\
\overset{(12)}{\leq}&\langle\mathbf{Z}_t - \mathbf{Z}_{t+1}, \nabla f_t(\mathbf{Z}_t)\rangle - \lambda||\mathbf{Z}_{t+1}||_* - \frac{1}{2\eta_t}||\mathbf{Z}_{t+1} - \mathbf{Z}_t||_F^2 - \frac{1}{2\eta_t}||\mathbf{Z}^* - \mathbf{Z}_{t+1}||_F^2 \\
&+ \lambda||\mathbf{Z}_t||_* + \frac{1}{2}\left(\frac{1}{\eta_t} - 1\right)||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + \langle\nabla F(\mathbf{Z}_t) - \nabla f_t(\mathbf{Z}_t), \mathbf{Z}_t - \mathbf{Z}^*\rangle \\
\leq&\max_{\mathbf{W}}\left(\langle\mathbf{W}, \nabla f_t(\mathbf{Z}_t)\rangle - \frac{1}{2\eta_t}||\mathbf{W}||_F^2\right) - \frac{1}{2\eta_t}||\mathbf{Z}_{t+1} - \mathbf{Z}^*||_F^2 \\
&+ \lambda||\mathbf{Z}_t||_* - \lambda||\mathbf{Z}_{t+1}||_* + \frac{1}{2}\left(\frac{1}{\eta_t} - 1\right)||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + \langle\nabla F(\mathbf{Z}_t) - \nabla f_t(\mathbf{Z}_t), \mathbf{Z}_t - \mathbf{Z}^*\rangle \\
=&\frac{\eta_t}{2}||\nabla f_t(\mathbf{Z}_t)||_F^2 - \frac{1}{2\eta_t}||\mathbf{Z}_{t+1} - \mathbf{Z}^*||_F^2 \\
&+ \lambda||\mathbf{Z}_t||_* - \lambda||\mathbf{Z}_{t+1}||_* + \frac{1}{2}\left(\frac{1}{\eta_t} - 1\right)||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + \langle\nabla F(\mathbf{Z}_t) - \nabla f_t(\mathbf{Z}_t), \mathbf{Z}_t - \mathbf{Z}^*\rangle,
\end{aligned} \tag{13}$$

where the third inquality holds based on the inequality in (12).

By substituting $\delta_t = \langle \boldsymbol{\xi}_t - \mathbf{K}, \mathbf{Z}_t - \mathbf{Z}^* \rangle$ and $C^2 = \max_{t\in[T]} ||\mathbf{Z}_t - \boldsymbol{\xi}_t||_F^2$ into (13),

$$||\mathbf{Z}_{t+1} - \mathbf{Z}^*||_F^2 \leq \eta_t^2 C^2 + 2\eta_t \delta_t + 2\lambda\eta_t \left(||\mathbf{Z}_t||_* - ||\mathbf{Z}_{t+1}||_*\right) + (1 - 2\eta_t) ||\mathbf{Z}_t - \mathbf{Z}^*||_F^2. \quad (14)$$

The inequality in (14) is the same as the result of Lemma 1 in Zhang et al. (2016). Thus, the following lemmas [1] from Zhang et al. (2016) can be directly utilized to derive a probability bound for $||\mathbf{Z}_{t+1} - \mathbf{Z}^*||_F^2$.

Lemma 2 (Lemma 2 in Zhang et al. (2016))**.** *Define $\gamma = \max_{t\in[T]} ||\mathbf{Z}_t||_*$. By setting $\eta_t = \frac{2}{t}$, an upper bound of $||\mathbf{Z}_{t+1} - \mathbf{Z}^*||_F^2$ can be written as*

$$||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2 \leq \frac{4(C^2 + \lambda\gamma)}{T} + \frac{2}{T(T-1)} \left[2\sum_{t=2}^{T} (t-1)\delta_t - \sum_{t=2}^{T} (t-1)||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 \right].$$

The upper bound of $\sum_{t=2}^{T} (t-1)\delta_t$ in Lemma 2 is then provided in Lemma 3.

Lemma 3 (Lemma 3 in Zhang et al. (2016))**.** *Assume $||\boldsymbol{\xi}_t - \mathbf{K}||_F \leq G$, and $||\mathbf{Z}_t - \mathbf{Z}^*||_F \leq H$, $\forall t > 2$. With a probability at least $1 - \delta$, $\sum_{t=2}^{T} (t-1)\delta_t$ is upper bounded by*

$$\sum_{t=2}^{T} (t-1)\delta_t \leq \frac{1}{2} \sum_{t=2}^{T} (t-1)||\mathbf{Z}_t - \mathbf{Z}^*||_F^2 + 2G^2\tau(T-1) + \frac{2}{3}GH(T-1)\tau + GH(T-1),$$

where $\tau = \log \frac{\lceil 2\log_2 T \rceil}{\delta}$.

Based on Lemma 2 and Lemma 3, the following upper bound of $||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2$ holds with a probability at least $1 - \delta$

$$, ||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2 \leq \frac{4}{T}\left(C^2 + \lambda\gamma + 2G^2\tau + \frac{2}{3}GH\tau + GH \right). \quad (15)$$

In Lemma 4, the upper bounds for C, γ, G, and H are provided.

Lemma 4 (Lemma 4 in Zhang et al. (2016))**.** *Assume $||\boldsymbol{\xi}||_F \leq L$. By setting $\eta_t = \frac{2}{t}$, it can be obtained that*

$$C^2 \leq 10L^2,\ \gamma \leq 2L \max_{t\in[T]} \sqrt{r_t},\ G = 2L,\ and\ H = 3L,$$

where r_t is the rank of $\mathbf{Z}_t$.

By substituting the upper bounds in Lemma 4 into (15), the upper bound of $||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2$ can be rewritten as

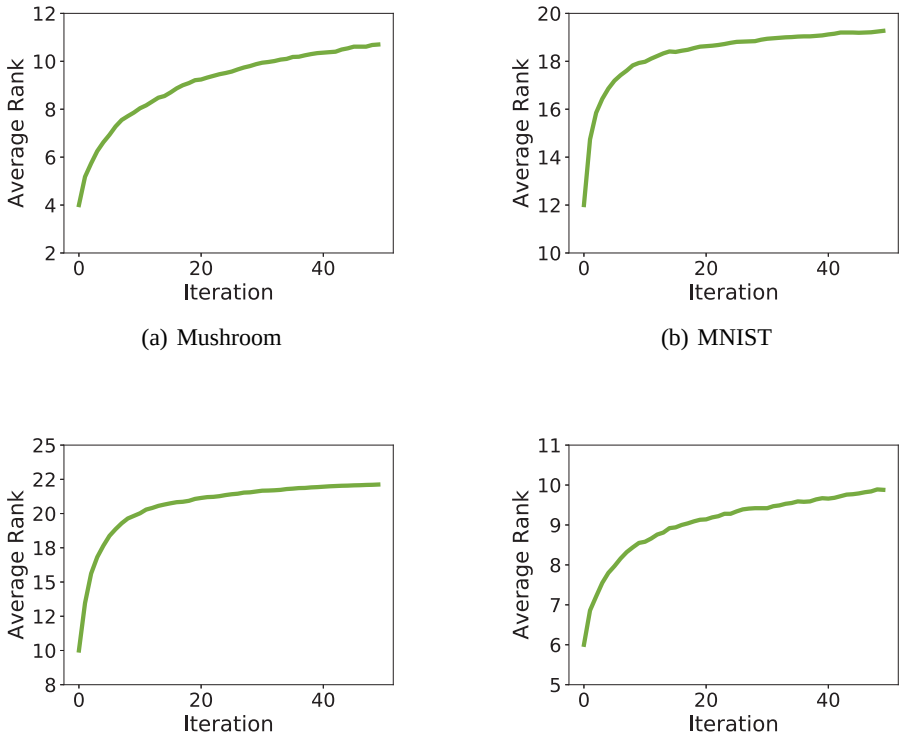
$$\begin{aligned} ||\mathbf{Z}_{T+1} - \mathbf{Z}^*||_F^2 \leq & \frac{8}{T}\left[\lambda L \max_{t\in[T]} \sqrt{r_t} + L^2 \left(8 + 6\log \frac{\lceil 2\log_2 T \rceil}{\delta} \right) \right] \\ = & O(1/T). \end{aligned}$$

D Proof of Theorem 2 and Empirical Results

For DSPGD with CEM, in the t-the iteration, the m-th user's device only needs to upload one vector in each iteration of DLA, i.e., $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]\mathbf{c}_q$. Since $\mathbf{W}_t[m] = [\sqrt{\frac{\eta_t}{D}}\mathbf{A}_t[m], \sqrt{1-\eta_t}\mathbf{B}_t[m]]$, the dimension of $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]\mathbf{c}_q$ equals $D + r_t$, where r_t is the rank of $\mathbf{Z}_t$ and D is the number of random feature. Moreover, DLA requires several Lanczos iterations to approach the eigenpairs of $\mathbf{W}_t^\top \mathbf{W}_t$. Thus, the communication cost of DSPGD with CEM is linear to $D + r_t$.

To compute the ratio, we first derive the communication cost for DSPGD without CEM. in the t-the iteration, the m-th user's device needs to upload two vectors in each iteration of DLA: $\mathbf{A}_t[m]^\top \mathbf{c}_q[m]$

[1] These lemmas can be found in the supplementary material of Zhang et al. (2016) that can be downloaded from `https://cs.nju.edu.cn/zlj/pdf/AAAI-2016-Zhang-S.pdf`



(a) Mushroom (b) MNIST

(c) Covtype (d) Smartphone

Figure 4: The rank of $\mathbf{Z}_t$ versus the number of iteration t for the four real-world dataset

and $\mathbf{A}_t[m]\mathbf{A}_t^\top \mathbf{c}_q$, where the dimension of $\mathbf{A}_t^\top \mathbf{c}_q$ equals D. For the concatenation of all M vectors $\{\mathbf{A}_t[m]\mathbf{A}_t^\top \mathbf{c}_q, m = 1, ..., M\}$, its dimension equals the number of data samples N. Thus, the communication cost of DSPGD without CEM is linear to $N + MD$.

Given the number of Lanczos iterations Q_0 for DSPGD without CEM and the number of Lanczos iterations Q_1 for DSPGD with CEM, the ratio can be determined by $\frac{(N+MD)Q_0}{M(r_t+D)Q_1}$.

For $\mathbf{R}_t$ and $\mathbf{W}_t^\top \mathbf{W}_t$, they have the same non-zero eigenvalues. Based on the property, it can be proved that the convergence rate of DLA when DLA is applied to $\mathbf{R}_t$ is the same with that of DLA when DLA is applied to $\mathbf{W}_t^\top \mathbf{W}_t$. While the values of Q_0 an Q_1 are not only affected by the convergence rate of DLA, but also affected by the selection of the initial vector. However, empirically, the values of Q_0 an Q_1 are at the same order, and the dominant factor of the ratio is still $\frac{N+MD}{M(r_t+D)}$.

Empirically, the figures of the average value of r_t versus the number of iterations t for the four real-world dataset are shown in Figure 4. The results show that the rank r_t tends to converge as the value of t increases. Besides, the upper bound of r_t is a constant factor larger than the number of eigenvectors s in Table 1, and such upper bound is much smaller than the number of users' local data samples, which can explain the dramatical reduction on the communication cost in Figure 2.

E Proof of Theorem 3

Define $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, and $\mathbf{P} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$. The low-rank approximation of $\mathbf{K}$ with rank s is denoted as $\mathbf{K}_s = \mathbf{U}\mathbf{\Lambda}_s\mathbf{U}^\top$, and $\mathbf{P}_s = \mathbf{U}\mathbf{\Lambda}_s^{\frac{1}{2}}$ where the diagonal of $\mathbf{\Lambda}_s$ contains the s largest eigenvalues of $\mathbf{K}$

while its rest diagonal entries are all zero. The output of DSPGD at iteration t is an estimation of $\mathbf{K}_s$, denoted as $\widetilde{\mathbf{K}}_t$, and $\widetilde{\mathbf{K}}_t = \widetilde{\mathbf{P}}_t \widetilde{\mathbf{P}}_t^\top$.

The following two lemmas will be used in the proof of Theorem 4.

Lemma 5. *Given $\widetilde{\mathbf{K}}_t$, the following inequality holds with a probability at least $1-\delta$ for any rank k projection matrix $\mathbf{\Pi} \in \mathbb{R}^{n\times n}$,*

$$\mathrm{Tr}(\mathbf{\Pi}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t)) \leq O(\sqrt{\frac{s}{t}})$$

Proof. Since $\mathbf{\Pi}$ is a rank-k projection matrix, it is obvious that

$$\mathrm{Tr}(\mathbf{\Pi}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t)) \leq ||\mathbf{K}_s - \widetilde{\mathbf{K}}_t||_*.$$

For a rank-s matrix $\mathbf{A}$, the following inequality holds for its Nuclear norm and its Frobenius norm

$$||\mathbf{A}||_*^2 \leq s||\mathbf{A}||_F^2.$$

Hence,

$$||\mathbf{K}_s - \widetilde{\mathbf{K}}_t||_* \leq \sqrt{s}||\mathbf{K}_s - \widetilde{\mathbf{K}}_t||_F.$$

By Theorem 3, $\mathbf{Z}_t$ converges to $\mathbf{Z}^*$ at an $O(1/t)$ rate. Note $\mathbf{Z}^*$ has the same eigenvectors as $\mathbf{K}_s$. Thus, $\widetilde{\mathbf{K}}_t$ constructed based on $\mathbf{Z}_t$ also converges to $\mathbf{K}_s$ at an $O(1/t)$ rate with a probability at least $1-\delta$, i.e., $||\mathbf{K}_s - \widetilde{\mathbf{K}}_t||_F^2$ has an upper bound as

$$||\mathbf{K}_s - \widetilde{\mathbf{K}}_t||_F^2 \leq O(1/t).$$

Hence, the following inequality holds with a probability at least $1-\delta$

$$\begin{aligned}\mathrm{Tr}(\mathbf{\Pi}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t)) &\leq \sqrt{s}||\mathbf{K}_s - \widetilde{\mathbf{K}}_t||_F \\ &\leq O(\sqrt{\frac{s}{t}}).\end{aligned}$$

□

Lemma 6. *Fix an error parameter $\varepsilon \in (0,1)$. For any rank k projection matrix $\mathbf{\Pi} \in \mathbb{R}^{n\times n}$,*

$$\mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K} - \widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right) \leq (\varepsilon + \frac{k}{s})||\mathbf{P} - \mathbf{\Pi}\mathbf{P}||_F^2.$$

Proof. It holds that

$$\begin{aligned}\mathrm{Tr}\left((\mathbf{I}_n - \mathbf{\Pi})(\mathbf{K} - \widetilde{\mathbf{K}}_t)\right) &= \mathrm{Tr}(\mathbf{K} - \widetilde{\mathbf{K}}_t) - \mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K} - \widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right) \\ &= ||\mathbf{K} - \mathbf{K}_s||_* + \mathrm{Tr}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t) - \mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K} - \widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right).\end{aligned}$$

Thus, $\mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K} - \widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right)$ can be rewritten as

$$\mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K} - \widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right) = ||\mathbf{K} - \mathbf{K}_s||_* + \mathrm{Tr}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t) - \mathrm{Tr}\left((\mathbf{I}_n - \mathbf{\Pi})(\mathbf{K} - \widetilde{\mathbf{K}}_t)(\mathbf{I}_n - \mathbf{\Pi})\right).$$

It follows that

$$\begin{aligned}&\mathrm{Tr}\left((\mathbf{I}_n - \mathbf{\Pi})(\mathbf{K} - \widetilde{\mathbf{K}}_t)(\mathbf{I}_n - \mathbf{\Pi})\right) \\ =&\mathrm{Tr}\left((\mathbf{I}_n - \mathbf{\Pi})(\mathbf{K} - \mathbf{K}_s)(\mathbf{I}_n - \mathbf{\Pi})\right) + \mathrm{Tr}\left((\mathbf{I}_n - \mathbf{\Pi})(\mathbf{K}_s - \widetilde{\mathbf{K}}_t)(\mathbf{I}_n - \mathbf{\Pi})\right) \\ =&\mathrm{Tr}\left((\mathbf{I}_n - \mathbf{\Pi})(\mathbf{K} - \mathbf{K}_s)(\mathbf{I}_n - \mathbf{\Pi})\right) + \mathrm{Tr}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t) - \mathrm{Tr}(\mathbf{\Pi}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t)) \\ \geq&||\mathbf{P} - \mathbf{P}_{s+k}||_F^2 + \mathrm{Tr}(\mathbf{K}_s - \widetilde{\mathbf{K}}_t) - O(\sqrt{\frac{s}{t}}),\end{aligned}$$

where the last inequality comes from Lemma 5. Thus,

$$\begin{aligned}
\mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K}-\widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right) &\leq ||\mathbf{K}-\mathbf{K}_s||_* - ||\mathbf{P}-\mathbf{P}_{s+k}||_F^2 + O(\sqrt{\frac{s}{t}}) \\
&= ||\mathbf{P}-\mathbf{P}_s||_F^2 - ||\mathbf{P}-\mathbf{P}_{s+k}||_F^2 + O(\sqrt{\frac{s}{t}}) \\
&= \sum_{i=s+1}^{n}\sigma_i^2(\mathbf{P}) - \sum_{i=s+k+1}^{n}\sigma_i^2(\mathbf{P}) + O(\sqrt{\frac{s}{t}}) \\
&= \sum_{i=s+1}^{s+k}\sigma_i^2(\mathbf{P}) + O(\sqrt{\frac{s}{t}}) \\
&\leq \frac{k}{s}\sum_{i=k+1}^{s+k}\sigma_i^2(\mathbf{P}) + O(\sqrt{\frac{s}{t}}).
\end{aligned}$$

Since $O(\sqrt{\frac{s}{t}})$ can be arbitrarily small, it can be rewritten as $O(\sqrt{\frac{s}{t}}) = \varepsilon||\mathbf{P}-\mathbf{P}_k||_F^2$. Besides, $\sum_{i=k+1}^{s+k}\sigma_i^2(\mathbf{P}) \leq \sum_{i=k+1}^{n}\sigma_i^2(\mathbf{P}) = ||\mathbf{P}-\mathbf{P}_k||_F^2$. Hence, it can be obtained that

$$\mathrm{Tr}\left(\mathbf{\Pi}(\mathbf{K}-\widetilde{\mathbf{K}}_t)\mathbf{\Pi}\right) \leq (\varepsilon + \frac{k}{s})||\mathbf{P}-\mathbf{P}_k||_F^2.$$

□

It can be obtained that

$$\begin{aligned}
||(\mathbf{I}_n-\mathbf{\Pi})\mathbf{P}||_F^2 - ||(\mathbf{I}_n-\mathbf{\Pi})\widetilde{\mathbf{P}}_t||_F^2 &= \mathrm{Tr}((\mathbf{I}_n-\mathbf{\Pi})\mathbf{P}\mathbf{P}^\top) - \mathrm{Tr}((\mathbf{I}_n-\mathbf{\Pi})\widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top) \\
&= \mathrm{Tr}(\mathbf{P}\mathbf{P}^\top - \widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top) - \mathrm{Tr}(\mathbf{\Pi}(\mathbf{P}\mathbf{P}^\top - \widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top)\mathbf{\Pi}).
\end{aligned}$$

Let $\alpha = \mathrm{Tr}(\mathbf{P}\mathbf{P}^\top - \widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top)$, and then the above equation can be rewritten as

$$||(\mathbf{I}_n-\mathbf{\Pi})\mathbf{P}||_F^2 + \mathrm{Tr}(\mathbf{\Pi}(\mathbf{P}\mathbf{P}^\top - \widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top)\mathbf{\Pi}) = \alpha + ||(\mathbf{I}_n-\mathbf{\Pi})\widetilde{\mathbf{P}}_t||_F^2.$$

After sufficient iterations, both α and $\mathrm{Tr}(\mathbf{\Pi}(\mathbf{P}\mathbf{P}^\top - \widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top)\mathbf{\Pi})$ are non-negative with a high probability. Thus, by Lemma 6 it holds that

$$\begin{aligned}
||(\mathbf{I}_n-\mathbf{\Pi})\mathbf{P}||_F^2 &\leq \alpha + ||(\mathbf{I}_n-\mathbf{\Pi})\widetilde{\mathbf{P}}_t||_F^2 \\
&= ||(\mathbf{I}_n-\mathbf{\Pi})\mathbf{P}||_F^2 + \mathrm{Tr}(\mathbf{\Pi}(\mathbf{P}\mathbf{P}^\top - \widetilde{\mathbf{P}}_t\widetilde{\mathbf{P}}_t^\top)\mathbf{\Pi}) \\
&\leq (1+\varepsilon+\frac{k}{s})||(\mathbf{I}_n-\mathbf{\Pi})\mathbf{P}||_F^2.
\end{aligned} \tag{16}$$

Based on (16), Theorem 4 can be proved as follows. Let $\mathbf{\Pi} = \widetilde{\mathbf{Y}}_t\widetilde{\mathbf{L}}_t\widetilde{\mathbf{Y}}_t^\top$, where $\widetilde{\mathbf{Y}}_t$ is the indicator matrix obtained by applying a γ-approximate algorithm to $\widetilde{\mathbf{P}}_t$, then

$$\begin{aligned}
||(\mathbf{I}_n - \widetilde{\mathbf{Y}}_t\widetilde{\mathbf{L}}_t\widetilde{\mathbf{Y}}_t^\top)\mathbf{P}||_F^2 &\leq \alpha + ||(\mathbf{I}_n - \widetilde{\mathbf{Y}}_t\widetilde{\mathbf{L}}_t\widetilde{\mathbf{Y}}_t^\top)\widetilde{\mathbf{P}}_t||_F^2 \\
&\leq \alpha + \gamma||(\mathbf{I}_n - \widetilde{\mathbf{Y}}_t^*\widetilde{\mathbf{L}}_t^*\widetilde{\mathbf{Y}}_t^{*\top})\widetilde{\mathbf{P}}_t||_F^2,
\end{aligned}$$

where $\widetilde{\mathbf{Y}}_t^*$ is the optimal indicator matrix for the linear k-means problem on $\widetilde{\mathbf{P}}_t$. Since $\gamma > 1$, it follows that

$$\begin{aligned}
\alpha + \gamma||(\mathbf{I}_n - \widetilde{\mathbf{Y}}_t^*\widetilde{\mathbf{L}}_t^*\widetilde{\mathbf{Y}}_t^{*\top})\widetilde{\mathbf{P}}_t||_F^2 &\leq \alpha + \gamma||(\mathbf{I}_n - \mathbf{Y}^*\mathbf{L}^*\mathbf{Y}^{*\top})\widetilde{\mathbf{P}}_t||_F^2 \\
&\leq \gamma(1+\varepsilon+\frac{k}{s})||(\mathbf{I}_n - \mathbf{Y}^*\mathbf{L}^*\mathbf{Y}^{*\top})\mathbf{P}||_F^2.
\end{aligned}$$

Thus,

$$||(\mathbf{I}_n - \widetilde{\mathbf{Y}}_t\widetilde{\mathbf{L}}_t\widetilde{\mathbf{Y}}_t^\top)\mathbf{P}||_F^2 \leq \gamma(1+\varepsilon+\frac{k}{s})||(\mathbf{I}_n - \mathbf{Y}^*\mathbf{L}^*\mathbf{Y}^{*\top})\mathbf{P}||_F^2,$$

which is equivalent to $f_K(\widetilde{\mathbf{Y}}_t) \leq \gamma(1+\varepsilon+\frac{k}{s})\min_{\mathbf{Y}} f_K(\mathbf{Y})$.

F PRIVACY PRESERVATION PROPERTY OF FEDERATED KERNEL k-MEANS

F.1 RECOVER USERS' DATA FROM RANDOM FEATURE MATRICES

A random feature for a data sample $\mathbf{x}_i$ has the form $\cos(\boldsymbol{\omega}^\top \mathbf{x}_i + b)$ where the $\boldsymbol{\omega}$ and b are determined by the cloud server. Since the value of $\boldsymbol{\omega}^\top \mathbf{x}_i + b$ cannot be arbitrarily large, the number of its possible values is limited. If enough such random features are collected, the cloud server can determine the value of $\boldsymbol{\omega}^\top \mathbf{x}_i + b$ for each random feature, and then recover $\mathbf{x}_i$ by solving a system of linear equations.

F.2 PROOF OF THEOREM 4

We then prove that the cloud server can at most recover the matrices $\{\mathbf{W}_t[m]^\top \mathbf{W}_t[m], m = 1, ..., M\}$ (only the multiplication $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]$ not the matrix $\mathbf{W}_t[m]$) from the local computational results (e.g., $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]\mathbf{c}_q$). The eigenpairs $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]$ are determined via the distributed Lanczos algorithm. Since

$$\mathbf{W}_t[m]^\top \mathbf{W}_t[m] = \begin{bmatrix} \frac{\eta_t}{D}\mathbf{A}_t[m]^\top \mathbf{A}_t[m] & \sqrt{\frac{\eta_t(1-\eta_t)}{D}}\mathbf{A}_t[m]^\top \mathbf{B}_t[m] \\ \sqrt{\frac{\eta_t(1-\eta_t)}{D}}\mathbf{B}_t[m]^\top \mathbf{A}_t[m] & (1-\eta_t)\mathbf{B}_t[m]^\top \mathbf{B}_t[m] \end{bmatrix},$$

$\mathbf{A}_t[m]^\top \mathbf{A}_t[m]$ can be recovered from $\mathbf{W}_t[m]^\top \mathbf{W}_t[m]$.

For a matrix $\mathbf{A}_t[m] \in \mathbb{R}^{n_m \times D}$ ($n_m < D$), a matrix $\mathbf{A}' \in \mathbb{R}^{n_m \times D}$ can be constructed via

$$\mathbf{A}' = \mathbf{U}_o \mathbf{A}_t[m],$$

where $\mathbf{U}_o \in \mathbb{R}^{n_m \times n_m}$ is an arbitrary orthogonal matrix with $\mathbf{U}_o^\top \mathbf{U}_o = \mathbf{I}_n$. By this construction, it can be derived that

$$\mathbf{A}'^\top \mathbf{A}' = \mathbf{A}_t[m]^\top \mathbf{U}_o^\top \mathbf{U}_o \mathbf{A}_t[m] = \mathbf{A}_t[m]^\top \mathbf{I}_n \mathbf{A}_t[m] = \mathbf{A}_t[m]^\top \mathbf{A}_t[m].$$

Since there exist infinite matrices $\mathbf{U}_o$ satisfying $\mathbf{U}_o^\top \mathbf{U}_o = \mathbf{I}_n$, the problem $\mathbf{A}_t[m]^\top \mathbf{A}_t[m] = \mathbf{A}'^\top \mathbf{A}'$ has infinite solutions. Hence, recovering the random feature matrix $\mathbf{A}_t[m]$ from $\mathbf{A}_t[m]^\top \mathbf{A}_t[m]$ is an ill-posed problem with infinite solutions.

Since by employing CEM, the cloud cannot recover the random feature matrices via matrix operations, according to Section F.1, it is infeasible for the cloud server to recover users' data by solving a system of linear equations.

G ADDITIONAL EXPERIMENTAL SETTINGS

The three public datasets (Mushrooms, MNIST, Covtype)[2] are selected from the LIBSVM dataset repository. The Smartphone dataset is provided by a company. The smartphone dataset contains the power consumption data of one app on users' smartphones. Its twelve features represent the power consumption on twelve hardware components. The clustering task for this dataset is to find the distinct usage patterns of the app based on the power consumption data. For the concern of privacy, the Smartphone dataset will not be disclosed.

The description of the four existing methods used in the experiments are listed as follows.

1. Centralized kernel k-means Zha et al. (2001) (denoted as CK k-means): directly perform truncated SVD on the kernel matrix $\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ to obtain a matrix that consists of the first s column vectors of $\mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$, and then apply linear k-means to this matrix;
2. Scalable kernel k-means Wang et al. (2019) (denoted as SK k-means): utilize the Nyström method to approximate the kernel matrix $\mathbf{K}$, and conduct kernel k-means over the approximated kernel matrix;

[2]These datasets can be downloaded from
`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

3. Distributed kernel k-means with random feature (Chitta et al., 2012) (denoted as RFK k-means): first transform the raw data samples to the corresponding random vector via the random Fourier feature method (Rahimi & Recht, 2008) and then utilize a distributed linear k-means to find the clusters in space of these random features;
4. Communication efficient distributed kernel PCA Balcan et al. (2016) (denoted as CE PCA): first conduct dimension reduction on the raw data samples through the communication efficient kernel PCA that integrates subspace embedding and adaptive sampling techniques to perform approximated kernel PCA in a distributed manner, and then apply a distributed linear k-means algorithm to the data samples after the dimension reduction.

For three distributed algorithms (FK k-means, RFK k-means, and CE PCA), the distributed linear k-means algorithm developed in (Balcan et al., 2013) is utilized to obtain the clustering results. Thus, the number of data samples C in the coreset should be assigned.

For FK k-means, the maximal iteration number T is selected from $[10, 20, 30, 40, 50]$. In the experiments of the Mushrooms dataset, C is set to 1000. In the experiment of the MNIST dataset, C is set to 1000. In the experiment of the Covtype dataset, C is set to 4000. In the experiment of the Smartphone dataset, C is set to 4000.

RFK k-means has two hyperparameters: the kernel parameter γ, and the number of random features D. The hyperparameter configuration for RFK k-means is set as follows. The value of γ is the same as that of FK k-means. In the experiments of the Mushrooms dataset, $D = 200$, and C is selected from $[100, 300, 500, 700]$. In the experiment of the MNIST dataset, $D = 800$, and C is selected from $[100, 300, 500, 700, 900]$. In the experiment of the Covtype dataset, $D = 100$, and C is selected from $[1000, 2000, 3000, 4000, 5000]$. In the experiment of the Smartphone dataset, $D = 50$, and C is selected from $[500, 1000, 2000, 3000]$.

CE PCA has six hyperparameters: the kernel parameter γ, the number of principle components d after PCA, the number of random features D, the subspace embedding dimension for the feature expansion d_s, The subspace embedding dimension for the data points d_p, and the number of representative points p. The hyperparameter configuration for CE PCA is set as follows. The value of γ is the same as that of FK k-means In the experiments over different datasets, some hyperparameters of these two methods are not changed. For CE PCA, $d_s = 50$, $d_p = 250$, and $p = 500$. Besides, the number of principle components d in CE PCA is the same as the number of eigenvectors s in FK k-means. To obtain different communication cost and normalized mutual information scores, the value of D is set to different values for both methods. In the experiment over the Mushrooms dataset, D is selected from $[20, 50, 100, 200]$, and C is set to 1000. In the experiment over the MNIST dataset, D is selected from $[100, 200, 400, 800]$, C is set to 1000. In the experiment over the Covtype dataset, D are selected from $[20, 50, 100, 200]$, and C is set to 4000. In the experiment over the Smartphone dataset, D is selected from $[20, 50, 100]$, C is set to 4000.

The discussion of the configuration of D is as follows. For RFK k-means and CE PCA, D is usually set to large values (more than 100). While for federated kernel k-means, D can be set to relatively small values (less than 50). The reason is as follows. RFK k-means (and CE PCA) only employs random feature once to estimate the kernel matrix. Thus, it requires a large number of random features to obtain an estimation of the kernel matrix with low approximation error, and furthermore a high NMI score. In contrast, federated kernel k-means is an iterative algorithm where random features are employed in each iteration to reduce the gap between the estimation and the kernel matrix. Hence, the number of random feature is not necessary to be set to a large value in each iteration.

The communication cost of the three algorithms are determined as follows. For FK k-means, the communication cost is the communication cost of DSPGD with CEM plus the that of the distributed linear k-means. For RFK k-means, its communication cost equals the communication cost of the distributed linear k-means. For CE PCA, its communication cost is the communication cost of performing distributed PCA plus the communication cost of the distributed linear k-means. The communication cost the distributed linear k-means equals the number of data samples in the coreset times the dimension of a data sample. In both FK k-means and CE PCA, the dimension of a data sample equals the number of eigenvectors s. In RFK k-means, the dimension of a data sample equals the number of random features D since the raw data cannot be exposed to the cloud server, and only the random feature vectors can be uploaded to the cloud server.