One-Shot Distributed Algorithm for PCA With RBF Kernels

Fan He, Kexin Lv, Jie Yang, and Xiaolin Huang, *Senior Member, IEEE*

***Abstract*—This letter proposes a one-shot algorithm for feature-distributed kernel PCA. Our algorithm is inspired by the dual relationship between sample-distributed and feature-distributed scenarios. This interesting relationship makes it possible to establish distributed kernel PCA for feature-distributed cases from ideas of distributed PCA in the sample-distributed scenario. In the theoretical part, we analyze the approximation error for both linear and RBF kernels. The result suggests that when eigenvalues decay fast, the proposed algorithm gives high-quality results with low communication cost. This result is also verified by numerical experiments, showing the effectiveness of our algorithm in practice.**

***Index Terms*—Distributed data, distributed learning, principal component analysis, one-shot algorithm, RBF kernels.**

I. Introduction

PRINCIPAL Component Analysis (PCA) is a fundamental technology in machine learning. For nonlinear tasks, PCA with given data $\mathbf{X} \in \mathcal{R}^{M\times T}$ could be formulated as follows,

$$\max_{\mathbf{w}^\top\mathbf{w}=\mathbf{I}} \|\mathbf{w}^\top\phi(\mathbf{X})\|_F^2. \tag{1}$$

Here $\phi(\cdot) : \mathcal{R}^M \to \mathcal{F}$ is an unknown non-linear mapping and $\|\cdot\|_F$ represents the Frobenius norm. Interestingly, the kernel trick could be implemented there, resulting in the kernel PCA (KPCA) [1]–[3]. Specifically, the optimal solution of (1) is $\mathbf{w} = \phi(\mathbf{X})\mathbf{V}$, where we use $\mathbf{V}$ for eigenvectors of the Gram kernel matrix $\mathbf{K} \triangleq \langle\phi(\mathbf{X}), \phi(\mathbf{X})\rangle$.

Nowadays, distributed algorithms of KPCA are in high demand. Generally, distributed data could be categorized into two regimes, namely, horizontally and vertically partitioned data [4], [5]. As shown in Fig. 1, we set features and samples as the horizontal and vertical axes. Then, when data are partitioned horizontally, agents contain part of samples with all features; when data are partitioned vertically, agents contain full samples but with only a part of features.

Manuscript received May 10, 2021; revised June 22, 2021; accepted June 22, 2021. Date of publication July 7, 2021; date of current version August 3, 2021. This work was supported in part by the National Natural Science Foundation of China under Grants 61977046, 61876107, and U1803261 and in part by Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dezhong Peng. *(Corresponding authors: Xiaolin Huang; Jie Yang.)*

The authors are with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the MOE Key Laboratory of System Control and Information Processing, Shanghai 200240, China (e-mail: hf-inspire@sjtu.edu.cn; kelen_lv@sjtu.edu.cn; jieyang@sjtu.edu.cn; xiaolinhuang@sjtu.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LSP.2021.3095017, provided by the authors.

Digital Object Identifier 10.1109/LSP.2021.3095017

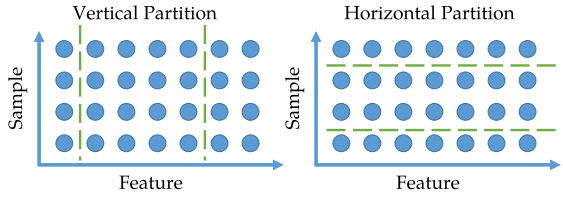

Fig. 1. Categorizations of data partition in distributed setting.

For PCA problems, there are massive researches focusing on both the horizontal [5]–[7] and the vertical regime [8]–[10]. However, although KPCA is very common in practice, e.g., in face recognition [11], [12] and process monitoring [13], [14], applicable distributed algorithms are not much. To the best of our knowledge, the existing studies on distributed KPCA are all for the horizontal regime, most of which require multi-communication rounds; see, e.g., [15]–[17].

In this letter, we aim to develop an one-shot distributed KPCA algorithm for vertically partitioned data. One-shot schemes mean to aggregate only once, which are thus communication-efficient and have been successfully applied in many distributed tasks, e.g., [18]. The main obstacle in vertical regime is that we only locally know a part of features. Then it seems that one cannot estimate the global kernel matrix without heavy communication. To handle this problem, we fully investigate the kernel trick that can transfer the primal optimization variables corresponding to features to dual variables corresponding to samples. With this idea, we can link KPCA in the vertical regime to PCA in the horizontal regime and then establish a distributed KPCA (DKPCA). The proposed DKPCA locally calculates eigenvectors corresponding to the first D eigenvalues of local kernel matrices and then sent them to a fusion center, where they are aggregated to produce the global estimator. Note that existing distributed PCA cannot be directly applied to KPCA in the horizontal regime. While the proposed DKPCA is applicable to both linear and RBF kernels with only one communication round, such that non-linear features can be efficiently extracted from complex data now. Theoretical discussions will show that the approximation error is related to the D-th eigenvalue of local matrices. When eigenvalues decay fast, DKPCA has very high-quality results, which is also confirmed by numerical experiments.

II. Distributed Kernel PCA

A. Preliminaries and Notations

We use regular letters for scalars, capital letters in bold for matrices, and lowercase letters in bold for vectors. We consider a distributed setting, where data are partitioned vertically and

Algorithm 1 One-Shot Distributed Algorithm for Kernel PCA in the Vertical Partition Regime. (DKPCA)

1: On local agents, calculate the local kernel matrix $\mathbf{K}^{(j)}$. Solve the eigenvalue problem on $\mathbf{K}^{(j)}$ to obtain $\mathbf{V}_D^{(j)}$ and $\boldsymbol{\lambda}_D^{(j)}$ and sent them to the fusion center.
2: On the fusion center, calculate $\hat{\mathbf{K}}$ to approximate $\mathbf{K}$.
- For linear kernel, use (2).
- For RBF kernel, use (3).
3: Compute the leading D eigenvectors $\hat{\mathbf{V}} \in \mathcal{R}^{T\times D}$ of the approximate matrix $\hat{\mathbf{K}}$.
4: **Return** $\hat{\mathbf{V}}$.

stored distributedly in J local agents. Such vertical regime are common in practice, e.g., in wireless sensor networks [19], [20] and ranking or evaluation systems [21], [22]. Specifically, each agent j acquires zero-mean data vectors $\mathbf{x}^{(j)} = \{\mathbf{x}_i^{(j)}\}_{i=1}^T \in \mathcal{R}^{M_j\times T}$, which are i.i.d. at time $i = 1, 2, \ldots, T$. M_j is the feature dimension of $\mathbf{x}^{(j)}$ and we have $\sum_{j=1}^J M_j = M$. Let $\mathbf{X} = [(\mathbf{x}^{(1)})^\top (\mathbf{x}^{(2)})^\top \cdots (\mathbf{x}^{(J)})^\top]^\top \in \mathcal{R}^{M\times T}$ denote all data collected by agents, which are not stored together but given for convenience.

Here we briefly review the KPCA problem, i.e., problem (1), where PCA is executed in a Reproducing Kernel Hilbert Space (RKHS) introduced by a reproducing kernel $\kappa(\cdot,\cdot)$. The goal of KPCA is to diagonalize the covariance matrix $\mathbf{C} = \sum_i \phi(\mathbf{x}_i) \otimes \phi(\mathbf{x}_i)^*$. However, it is hard to do eigendecomposition on $\mathbf{C}$ since ϕ is implicit. Then KPCA turns to solve the dual eigenproblem, where the eigendecomposition is performed on the Gram kernel matrix $\mathbf{K} \in \mathcal{R}^{T\times T}$, i.e.,

$$\mathbf{K}_{p,q} = \kappa(\mathbf{x}_p, \mathbf{x}_q) = \langle \phi(\mathbf{x}_p), \phi(\mathbf{x}_q)\rangle, p, q = 1, \ldots, T,$$
$$\lambda_d \mathbf{V}_d = \mathbf{K}\mathbf{V}_d, d = 1, \ldots, T,$$

where $\mathbf{V}_d \in \mathcal{R}^{T\times 1}$ is the eigenvector of $\mathbf{K}$ corresponding to the d-th largest eigenvalue λ_d. Then the d-th eigenvector of $\mathbf{C}$ can be rewritten as $\mathbf{w}_d = \phi(\mathbf{X})\mathbf{V}_d$. Such kernel trick sidesteps the problem of computing unknown $\phi(\cdot)$ and moreover, it makes the distributed computation for vertically partitioned data more convenient because:

- The covariance $\mathbf{C}$ is not separable and generally the approximation by local covariance matrices is not accurate [23], e.g., $\mathbf{C} \neq \sum_j \mathbf{C}^{(j)} = \sum_j \sum_i \phi(\mathbf{x}_i^{(j)}) \otimes \phi(\mathbf{x}_i^{(j)})^*$.
- $\mathbf{K}$ itself (linear kernels) or its main calculation part (RBF kernels) is separable, e.g., a linear kernel $\mathbf{K} = \sum_j \mathbf{K}^{(j)}$.

B. Distributed Algorithm for Kernel PCA

We first link the existing distributed algorithm for PCA in horizontal regime [5] to that for KPCA in vertical regime and then extend it to RBF kernels. The proposed algorithm could produce a good estimation of the global optimum in one communication round with privacy protection.

In the horizontal regime, the key property is the consistency between the sum of local covariance matrices and the global covariance matrix, i.e., $\mathbf{C} = \sum_j \mathbf{C}_j$, which results in benefits for both algorithm design and theory analysis. Considering linear KPCA in vertical regime is a dual problem of linear PCA in horizontal regime, we can easily transfer the algorithm in [5] to that of KPCA. Mathematically, for linear kernels, $\phi(\mathbf{x}) = a\mathbf{x}, \forall a \in \mathcal{R}$. We assume $a = 1$ without loss of generality and then it holds that $\mathbf{K} = \mathbf{X}^\top\mathbf{X} = \sum_{j=1}^J (\mathbf{x}^{(j)})^\top \mathbf{x}^{(j)} = \sum_j \mathbf{K}^{(j)}$. Thus, the estimator $\hat{\mathbf{K}}$ is calculated as follows.

$$\hat{\mathbf{K}} = \sum\nolimits_j \hat{\mathbf{K}}^{(j)} = \sum\nolimits_j \mathbf{V}_D^{(j)} \boldsymbol{\lambda}_D^{(j)} (\mathbf{V}_D^{(j)})^\top. \tag{2}$$

However, in horizontal regime, the algorithm in [5] cannot be extended to non-linear cases, limiting its application in practice. Instead, in vertical regime, we can further extend this idea to RBF kernels by noting the following property.

$$\begin{aligned}\mathbf{K}(p,q) &= \kappa(\mathbf{x}_p, \mathbf{x}_q) = \exp\left(-\frac{\|\mathbf{x}_p - \mathbf{x}_q\|_2^2}{2\sigma^2}\right)\\ &= \exp\left(-\frac{\sum_{j=1}^J (\mathbf{x}_p^{(j)} - \mathbf{x}_q^{(j)})^2}{2\sigma^2}\right)\\ &= \exp\left(\frac{-(\mathbf{x}_p^{(1)} - \mathbf{x}_q^{(1)})^2}{2\sigma^2}\right)\cdots\exp\left(\frac{-(\mathbf{x}_p^{(J)} - \mathbf{x}_q^{(J)})^2}{2\sigma^2}\right)\\ &= \mathbf{K}^{(1)}(p,q)\cdot\mathbf{K}^{(2)}(p,q)\cdot\cdots\cdot\mathbf{K}^{(J)}(p,q),\end{aligned}$$

where σ is the kernel width. Using $\circ$ to denote the Hadamard (element-wise) product operator, we have $\mathbf{K} = \mathbf{K}^{(1)} \circ \mathbf{K}^{(2)} \circ \cdots \circ \mathbf{K}^{(J)}$. Therefore, once the eigenvectors of each local kernel matrix are obtained, the whole kernel matrix $\mathbf{K}$ could be approximated as the following way,

$$\begin{aligned}\hat{\mathbf{K}} &= \hat{\mathbf{K}}^{(1)} \circ \cdots \circ \hat{\mathbf{K}}^{(J)}\\ &= \left(\mathbf{V}_D^{(1)}\boldsymbol{\lambda}_D^{(1)}(\mathbf{V}_D^{(1)})^\top\right) \circ \cdots \circ \left(\mathbf{V}_D^{(J)}\boldsymbol{\lambda}_D^{(J)}(\mathbf{V}_D^{(J)})^\top\right).\end{aligned} \tag{3}$$

Finally, we compute the first D eigenvectors of $\hat{\mathbf{K}}$, denoted as $\hat{\mathbf{V}}$, and the projection matrix $\hat{\mathbf{W}} = \sum_i \hat{\mathbf{V}}_i \phi(\mathbf{x}_i)$. Notice that for this calculation, $\mathbf{W}$ is unknown but $\langle \hat{\mathbf{W}}, \phi(\mathbf{y})\rangle$ can be calculated by kernel trick in a distributed system because $\mathbf{x}^\top\mathbf{y}$ and $\|\mathbf{x} - \mathbf{y}\|_F^2$ is separable along features. The overall algorithm is summarized in Algorithm 1.

Approximation Analysis: Here we present the approximation analysis for DKPCA in both linear and RBF cases. Specifically, we study the $\sin\Theta$ distance between the eigenspaces spanned by $\mathbf{V}$ and $\hat{\mathbf{V}}$, where $\mathbf{V}$ are the eigenvectors of the global gram kernel matrix $\mathbf{K}$, and $\hat{\mathbf{V}}$ is the estimator calculated by DKPCA. $\sin\Theta$ distance is well-defined and is widely used for measuring the distance between two linear spaces [5], [24]. Let $\alpha_1, \ldots, \alpha_D$ be the singular values of $\mathbf{V}^\top\hat{\mathbf{V}}$ and define $\sin\Theta(\mathbf{V}, \hat{\mathbf{V}})$ as follows.

$$\begin{aligned}\Theta(\mathbf{V}, \hat{\mathbf{V}}) &= \mathrm{diag}\{\cos^{-1}(\alpha_1), \ldots, \cos^{-1}(\alpha_d)\}\\ &\triangleq \mathrm{diag}\{\theta_1, \ldots, \theta_d\}\\ \sin\Theta(\mathbf{V}, \hat{\mathbf{V}}) &= \mathrm{diag}\{\sin(\theta_1), \ldots, \sin(\theta_d))\}.\end{aligned}$$

Then we have the following theorem for DKPCA.

Theorem 1: Let $\mathbf{V} \in \mathcal{R}^{T\times D}$ be the leading D eigenvectors of the global kernel matrix $\mathbf{K} \in \mathcal{R}^{T\times T}$ that is derived by a kernel function κ, and $\hat{\mathbf{V}}$ be its approximation computed by DKPCA.

TABLE I
COMMUNICATION AND COMPUTATION COMPLEXITY OF THREE KINDS OF ALGORITHMS FOR PCA

	COMM.	COMP.
DPCA [5]	$\mathcal{O}(DM)$	$\mathcal{O}((T+DJ)M^2)$
DKPCA (Alg. 1)	$\mathcal{O}(DT)$	$\mathcal{O}((M+DJ)T^2)$
KPCA (EVD-based,[25])	$\mathcal{O}(TM/J)$	$\mathcal{O}(DT^2+MT^2)$

If κ is a linear kernel, then $\mathbf{V}, \hat{\mathbf{V}}$ satisfy

$$\|\sin\Theta(\mathbf{V},\hat{\mathbf{V}})\|_F \leq \frac{J\sqrt{T-D}\max_j(\lambda_{D+1}^{(j)})}{\lambda_D(\mathbf{K})-\lambda_{D+1}(\mathbf{K})}.$$

If κ is a RBF kernel, then $\mathbf{V}, \hat{\mathbf{V}}$ satisfy

$$\|\sin\Theta(\mathbf{V},\hat{\mathbf{V}})\|_F \leq \frac{J\sqrt{T}\max_j(\lambda_{D+1}^{(j)})}{\lambda_D(\mathbf{K})-\lambda_{D+1}(\mathbf{K})}.$$

See supplemental materials for the proof of Theorem 1. Theorem 1 indicates that the approximation error is related with J, T, D, and λ. If λ decay fast, which is common for RBF kernels, then DKPCA will have very high-quality results.

Remark 1: Theorem 1 relies on the large gap between $\lambda_D(\mathbf{K})$ and $\lambda_{D+1}(\mathbf{K})$ due to the use of the Davis-Kahan Theorem [24]. Thus the error is unbounded if the gap is nearly zero. Note that such a gap is an inherent attribute of data that cannot be controlled. Hence, when applying DKPCA in practice, we must carefully choose D such that the gap is large, i.e., we should avoid dealing with the repeated eigenvalues.

Communication and Computation cost: Algorithm 1 is quite efficient with only one communication round. To analyze quantitatively, we restrict our discussion to an evenly and homogeneously distributed situation, i.e., the local feature dimension is $\mathcal{O}(M/J)$ and there is no statistical difference among agents. The discussion on other cases is similar but is more complicated in form. We summarize communication and computation complexity of Algorithm 1 (DKPCA), distributed PCA (DPCA) [5] and central KPCA in Table I. Here the computational complexity of eigenvalue decomposition (EVD) is considered to be $O(DT^2+MT^2)$ [25], where the leading D eigenvectors are obtained from data $\mathbf{X}\in\mathcal{R}^{M\times T}$. Compared with central algorithms, where additional communication and fusion are required, DKPCA sacrifices computation efficiency for communication efficiency. To pursue high communication efficiency, we prefer a small D, e.g., when D is much smaller than M/J, Then, DKPCA will have significant advantages over central algorithms on communication cost and the computation cost is $\mathcal{O}(MT^2)$, the same as central algorithms.

Self-adaptive strategy for data maldistribution: Hereinbefore, we simply set equal $D^{(j)}$ for all agents in DKPCA, which, however, may not work well for the case of data maldistribution. Theorem 1 shows that for given J, T, D[1], the error is related to δ and $\max_j\lambda_{D+1}^{(j)}$. To reduce the approximate error, small $\lambda_{D+1}^{(j)}$ is preferred. Thus, local agents that have larger eigenvalues (which means they hold more information) need to send more eigenvectors. On the other side, in some agents, the eigenvalues decay fast so that the D-th one is almost 0. Then sending D eigenvectors is redundant.

[1]In fact, J, T, δ are the inherent attribute of data that we cannot change. We will show the different performance of DKPCA for data with different J, T in section III.

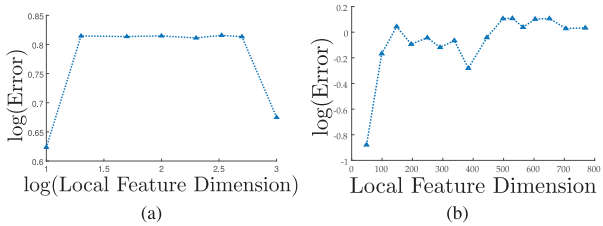

Fig. 2. The log of the mean error with respect to the number of local agents. (a) Simulation data and linear kernels with $M=1000, D=10$ and $T=400$. (b) Real data and RBF kernels ($\sigma=\frac{\sqrt{M}}{3}$) with $M=8545, D=10$ and $T=537$.

Therefore, we consider the following self-adaptive strategy for the selection of $D^{(j)}$ such that each agent can decide the number of the eigenvectors sent to the center according to their local eigenvalues $\boldsymbol{\lambda}^{(j)}$:

$$D^{(j)} = \min\{\min_D\{\lambda_{D+1}(\mathbf{K}^{(j)})\leq\epsilon\}, T\}, \tag{4}$$

where ϵ is a positive threshold value. Note that the decays of $\boldsymbol{\lambda}_i$ are generally assumed to be polynomial or exponential [26], i.e. $\lambda_i=\mathcal{O}(i^{-\delta})$ or $\lambda_i=\mathcal{O}(e^{-\rho i})$. Then the D given by (4) is $\Omega((\frac{c}{\epsilon})^{\frac{1}{\delta}})$ or $\Omega(-\frac{1}{\rho}\log(\frac{\epsilon}{c}))$, where c is some constant.

III. EXPERIMENTS

The performance of DKPCA is evaluated here by three experiments, where both simulation and real data are used for linear cases and non-linear cases, respectively. Details of simulation data generation are in supplemental materials. Real data used in our experiments include the gene expression of different drugs and toxicants on rats [27], which is collected on cRNA microarray chips and is available at the NIH GEO, under accession number GSE2187. The size of total data is 8565×537, corresponding to four categories: fibrates (107 samples), statins (93 samples), azoles (156 samples), and toxicants (181 samples). The features are removed if more than 10% of the samples have their values missing. The rest missing values are filled with mean values.

The leading D eigenvectors $\mathbf{V}_{gt}\in\mathcal{R}^{D\times T}$ calculated by performing the SVD algorithm on the whole underlying kernel matrix are regarded as the ground truth. We use $\sin\Theta$ distance to evaluate the error of the estimator $\hat{\mathbf{V}}\in\mathcal{R}^{D\times T}$, i.e., $\text{Error}=D-\|\mathbf{V}_{gt}^\top\hat{\mathbf{V}}\|_F^2$. All the simulations are repeated 50 times and are done with Matlab R2016b in Core i5-7300HQ 2.50 GHz 8 GB RAM. The codes of DKPCA and experiments are available in https://github.com/hefansjtu/DKPCA.

Relationship between Estimation Error and the Number of Local Agents: In this experiment, we vary the number of local agents to see how it affects the estimation error. The result is reported in Fig. 2, where (a) linear kernels with $M=1000$ and $T=400$ and (b) RBF kernels with $\sigma=\frac{\sqrt{M}}{3}, M=8545$ and $T=537$ are considered. The data are uniformly partitioned and the target is to estimate the first $D=10$ eigenvectors of the global kernel matrix $\mathbf{K}$.

Fig. 2 shows that overall the estimation error is small. For different numbers of local agents, the estimation error is similar except the extreme cases: there is only one agent or each agent transmits all the data. This phenomenon can be explained by Theorem 1, which indicates that the error bound has a positive correlation to the number of local agents J and the D-th local eigenvalues. It should be noted that the information in local

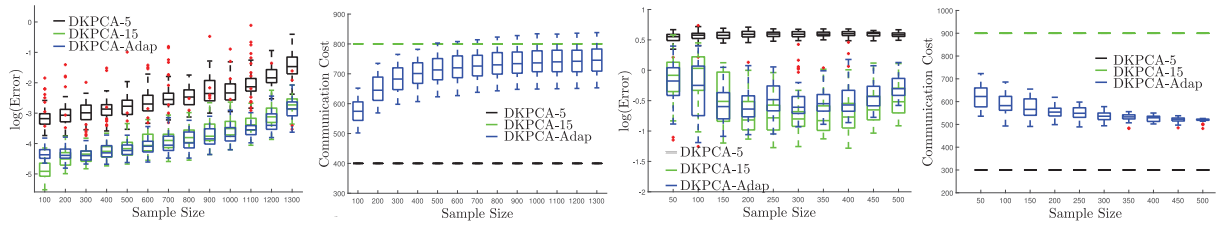

Fig. 3. The log of the mean error and the total number of local eigenvectors that transmitted to the center with respect to the number of samples. (a-b) Simulation data and the linear kernel are used, where $M = 1000,\ D = 10$ and $J = 10$. (c-d) Real data and the RBF kernel are used with $\sigma = \frac{\sqrt{M}}{3}$, $M = 8545,\ D = 10$ and $J = 60$.

TABLE II
CLASSIFY ERROR RATE ON GSE2187 DATASET USING L-SVM WITH DKPCA, KPCA AND PCA

Number of Features		1	5	10	20	50	100	200
Toxicant vs Fibrate	DKPCA	0.3675±0.0383	0.0323±0.0218	0.0164±0.0128	0.0150±0.0118	0.0148±0.0111	0.0145±0.0110	0.0148±0.0115
	KPCA	0.3675±0.0383	0.0259±0.0151	0.0173±0.0124	0.0148±0.0111	0.0150±0.0111	0.0145±0.0110	0.0148±0.0115
	PCA	0.5048±0.0932	0.2009±0.1121	0.0352±0.0517	0.0259±0.0551	0.0277±0.0702	0.0264±0.0587	0.0243±0.0562
Toxicant vs Azole	DKPCA	0.4790±0.0410	0.3966±0.0305	0.2778±0.0361	0.1907±0.0376	0.1416±0.0331	0.1099±0.0283	0.0969±0.0259
	KPCA	0.4889±0.0350	0.3912±0.0341	0.2778±0.0379	0.1844±0.0341	0.1391±0.0325	0.1102±0.0283	0.0969±0.0259
	PCA	0.5019±0.0616	0.5124±0.0415	0.4655±0.0620	0.3781±0.0782	0.2207±0.1309	0.1982±0.1502	0.2134±0.1671
Toxicant vs Others	DKPCA	0.3341±0.0148	0.3326±0.0202	0.2477±0.0209	0.2004±0.0231	0.1499±0.0197	0.1428±0.0169	0.1410±0.0172
	KPCA	0.3341±0.0148	0.3284±0.0267	0.2420±0.0229	0.1973±0.0207	0.1488±0.0190	0.1430±0.0176	0.1410±0.0172
	PCA	0.4578±0.1367	0.4928±0.0992	0.4129±0.0763	0.3526±0.0888	0.2045±0.1324	0.1953±0.1613	0.2077±0.1584

decreases when J increases, which further leads to the decay of local eigenvalues. Thus, even when the number of local agents increases, the accuracy could be maintained at a high level.

Self-adaptive Strategy for Data Maldistribution: The error of DKPCA and DKPCA with self-adaptive strategy (DKPCA-Adap) are reported in Fig. 3 with linear and RBF kernels. The target is to recover the first $D = 10$ eigenvectors in data maldistribution cases, where the local feature dimensions are randomly generated. The result of DKPCA with fixed $D^{(j)} = C$ is denoted as DKPCA-C. ϵ in DKPCA-Adap is set as $0.04\lambda_1$ for linear cases and $0.0005\lambda_1$ for non-linear cases, where λ_1 denotes the largest eigenvalue in local. Different T are also considered to see how the accuracy changes.

Fig. 3 demonstrates the relationship between the sample size and (a) the estimation error (b) the total communication cost $\sum_j D^{(j)}$. Intuitively, the performance of DKPCA-5 is worst because it requires the least communication, and thus DKPCA-15 is the best. The DKPCA-Adap achieves similar accuracy as the DKPCA-15 but requires much less communication, showing the effectiveness of the self-adaptive strategy.

Interestingly, the tendency of communication cost w.r.t. T are different in Fig. 3, which increases in linear cases but decreases in RBF cases. The main reason is the rank of data. The simulation data is low-rank and thus adding new samples brings little new information. Therefore, the increase of sample size forces all the eigenvalues to grow uniformly, leading to the increase of $D^{(j)}$ and thus the high communication cost. But the high-dimension real data is small-size, adding new data causes the increase of principal eigenvalue (more than other eigenvalues), leading to the decrease of the communication cost. In conclusion, this strategy is more suitable for small distributed datasets with high dimensions.

Comparison between distributed and full sample KPCA in real classification tasks: The aim of PCA is to keep useful information during data projection. Here we consider 3 two-class classification tasks to show that DKPCA can preserve similar information as KPCA. That is, the same post-learning algorithm can achieve similar performance on the two projected data produced by DKPCA and KPCA. We first map data into a low-dimension feature space by DKPCA, the central kernel algorithm (KPCA) with RBF kernels with $\sigma = 50$, and the central linear algorithm (PCA), for which the corresponding eigenproblems are all solved by SVD. The feature dimension of the projected data changes from 1 to 200. The projected data are then sent to a linear support vector machine (L-SVM). We randomly choose 200 data as the training set and use the rest for the test.

The average classification error and its standard deviation over 50 trials are reported in Table II. As the feature dimension of the projected data increases, the classification error rates of all methods decrease. KPCA is based on full data and is expected to be better than the proposed DKPCA. But from Table II, one could observe that the difference is slight. In KPCA and DKPCA, the RBF kernel is applied and thus achieves less classification error than PCA, although the PCA is conducted on full data. Notice that before the proposed DKPCA, no distributed PCA algorithm can apply nonlinear kernels in vertical regime.

IV. CONCLUSION

This letter introduced a one-shot privacy-preserving algorithm, DKPCA, for distributed kernel PCA in vertical regime. The main technique is the kernel trick, by which the vertical regime and horizontal regime are linked up and nonlinear dimension reduction could be implemented with RBF kernels. We presented the approximation analysis and experiments result of DKPCA, which coincides with the theoretical result and demonstrate that DKPCA is effective to extract non-linear features and is communication efficient.

REFERENCES

[1] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 536–542.

[2] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[3] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Proc. Int. Conf. Artif. Neural Netw*. 1997, pp. 583–588.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[5] J. Fan, D. Wang, K. Wang, and Z. Zhu, "Distributed estimation of principal eigenspaces," *Ann. Statist.*, vol. 47, no. 6, 2019, Art. no. 3009.

[6] J. Ge, Z. Wang, M. Wang, and H. Liu, "Minimax-optimal privacy-preserving sparse PCA in distributed systems," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1589–1598.

[7] D. Garber, O. Shamir, and N. Srebro, "Communication-efficient algorithms for distributed stochastic principal component analysis," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1203–1212.

[8] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Proc. 42nd Asilomar Conf. Signals, Syst. Comput.*, 2008, pp. 1722–1726.

[9] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 725–738, Aug. 2011.

[10] I. D. Schizas and A. Aduroja, "A distributed framework for dimensionality reduction and denoising," *IEEE Trans. Signal Process.*, vol. 63, no. 23, pp. 6379–6394, Dec. 2015.

[11] K. I. Kim, K. Jung, and H. J. Kim, "Face recognition using kernel principal component analysis," *IEEE Signal Process. Lett.*, vol. 9, no. 2, pp. 40–42, Feb. 2002.

[12] C. Liu, "Gabor-based kernel PCA with fractional power polynomial models for face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 572–581, May 2004.

[13] S. W. Choi and I. Lee, "Nonlinear dynamic process monitoring based on dynamic kernel PCA," *Chem. Eng. Sci.*, vol. 59, no. 24, pp. 5897–5908, 2004.

[14] X. Liu, U. Kruger, T. Littler, L. Xie, and S. Wang, "Moving window kernel PCA for adaptive monitoring of nonlinear processes," *Chemometrics Intell. Lab. Syst.*, vol. 96, no. 2, pp. 132–143, 2009.

[15] R. Rosipal and M. Girolami, "An expectation-maximization approach to nonlinear component analysis," *Neural Comput.*, vol. 13, no. 3, pp. 505–510, 2001.

[16] W. Zheng, C. Zou, and L. Zhao, "An improved algorithm for kernel principal component analysis," *Neural Process. Lett.*, vol. 22, no. 1, pp. 49–56, 2005.

[17] M. F. Balcan, Y. Liang, L. Song, D. Woodruff, and B. Xie, "Communication efficient distributed kernel principal component analysis," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 725–734.

[18] E. Dobriban and Y. Sheng, "Wonder: Weighted one-shot distributed ridge regression in high dimensions," *J. Mach. Learn. Res.*, vol. 21, no. 66, pp. 1–52, 2020.

[19] O. Ghorbel, M. W. Jmal, M. Abid, and H. Snoussi, "Distributed and efficient one-class outliers detection classifier in wireless sensors networks," in *Proc. Int. Conf. Wired/Wireless Internet Commun*., 2015, pp. 259–273.

[20] Y.-A. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[21] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson, "Distributed clustering using collective principal component analysis," *Knowl. Inf. Syst.*, vol. 3, no. 4, pp. 422–448, 2001.

[22] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[23] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Signal Process.*, vol. 104, pp. 120–135, 2014.

[24] Y. Yu, T. Wang, and R. J. Samworth, "A useful variant of the Davis-Kahan theorem for statisticians," *Biometrika*, vol. 102, no. 2, pp. 315–323, 2015.

[25] A. Sharma and K. K. Paliwal, "Fast principal component analysis using fixed-point algorithm," *Pattern Recognit. Lett.*, vol. 28, no. 10, pp. 1151–1155, 2007.

[26] F. R. Bach, "Sharp analysis of low-rank kernel matrix approximations," in *Proc. Conf. Learn. Theory*, 2013, pp. 185–209.

[27] G. Natsoulis *et al.*, "Classification of a large microarray data set: Algorithm comparison and analysis of drug signatures," *Genome Res.*, vol. 15, no. 5, pp. 724–736, 2005.