DISTRIBUTED PRINCIPAL COMPONENTS ANALYSIS IN SENSOR NETWORKS

Abiodun Aduroja, Ioannis D. Schizas* and Vasileios Maroulas†*

* Department of EE, Univ. of Texas at Arlington, 416 Yates Street, Arlington, TX 76010, USA

† Department of Math, Univ. of Tennessee at Knoxville, Knoxville, TN 37996, USA

ABSTRACT

Estimation of the principal eigenspace of a data covariance matrix is instrumental in applications such as data dimensionality reduction and denoising. In sensor networks the acquired data are spatially scattered which further calls for the development of distributed principal subspace estimation algorithms. Toward this end, the standard principal component analysis framework is reformulated as a separable constrained minimization problem which is solved by utilizing coordinate descent techniques combined with the alternating direction method of multipliers. Computationally simple local updating recursions are obtained that involve only single-hop inter-sensor communications and allow sensors to estimate the principal covariance eigenspace in a distributed fashion. Adaptive implementations are also considered that allow online information processing. Numerical tests demonstrate that the novel algorithm has the potential to achieve a considerably faster convergence rate and better steady-state estimation performance compared to existing alternatives.

Index Terms— Distributed processing, principal component analysis

1. INTRODUCTION

Data acquired across a network of sensors monitoring a field, often lie on a small dimensional subspace. The presence of a few sources in the sensed field leads to a low-rank data covariance matrix. Estimating the principal eigenspace of such covariances is essential for applications including data dimensionality reduction and sensor data denoising [1, 2]. Such tasks are tackled via principal component analysis (PCA) which focuses on recovering the principal covariance eigenvectors and projecting the data on the principal covariance eigenspace. The projection filters out noise and/or reduces the data dimensionality in a mean-square error (MSE) optimal way [1, 3].

Traditional PCA techniques have been developed assuming that all data are gathered at a central processing unit [1, 2, 4]. The distributed structure of sensor networks, where data are scattered across sensors, motivates the development of decentralized PCA techniques that are applicable even under single-hop communications between neighboring sensors. To this end, data aggregation techniques have been put forth in [5] to collect sensor data at a fusion center and perform PCA. A different setting is considered in [6], where training data are scattered across sensors and consensus-averaging techniques [7, 8] are employed such that each sensor estimates the data covariance matrix and performs eigenvalue decomposition to recover the principal eigenspace. A distributed scheme is developed in [9] under the assumption that the data covariance matrix has a decomposable structure. A more related in-network distributed PCA scheme was developed in [10], where each sensor estimates a specific part of the principal covariance eigenspace by combining gradient descent iterations [4] with vector and matrix consensus-averaging to diffuse information across the network.

Different from [5,6], we derive a distributed PCA algorithm that relies on in-network processing and does not require evaluation of the whole covariance matrix at a central location, or across all sensors; the latter task has a large communication and computational cost. Further, no special structure is imposed here on the data covariance as in [9]. Our approach entails expressing the standard PCA cost as a separable constrained minimization problem. This separable formulation is tackled in a distributed fashion by employing the alternating direction method of multipliers (ADMM) [11] combined with block coordinate descent iterations [12] (Sec. 3.1) which leads to a set of computationally simple local recursions. An adaptive implementation, that facilitates online information processing, is also developed (Sec. 4). As corroborated by numerical tests, the novel distributed PCA has the potential to achieve a considerably faster convergence rate while reaching a better steady-state performance compared to the consensus-based approach in [10]. This is important given the limited life-span of sensor networks.

2. PROBLEM STATEMENT AND PRELIMINARIES

Consider an ad hoc sensor network consisting of p sensors that monitor a field. Each sensor j is able to communicate only with its single-hop neighbors in $\mathcal{N}_j$, having cardinality $|\mathcal{N}_j|$. Assuming that inter-sensor links are symmetric, the sensor network is modeled as an undirected connected graph. Sensor j acquires scalar zero-mean measurements $\{x_\tau(j)\}_{j=1}^p$ at discrete time instances $\tau = 0, 1, 2, \ldots, t$. The scattered sensor measurements, stacked in $\mathbf{x}_\tau := [x_\tau(1), \ldots, x_\tau(p)]^T$, have covariance matrix $\mathbf{\Sigma}_x = \mathbb{E}[\mathbf{x}_\tau \mathbf{x}_\tau^T]$. Let $\mathbf{\Sigma}_x = \mathbf{U}_x \mathbf{\Lambda}_x \mathbf{U}_x^T$ denote the eigenvalue decomposition, where $\mathbf{U}_x \in \mathbb{R}^{p\times p}$ is the eigenvector matrix while the diagonal $\mathbf{\Lambda}_x$ contains the corresponding eigenvalues. It is of interest to estimate, using the sensor data $\{\mathbf{x}_\tau\}_{\tau=0}^t$, the r principal eigenvectors of $\mathbf{\Sigma}_x$ denoted as $\mathbf{U}_{x,r}$ which can be used for denoising, dimensionality reduction and so on.

The principal eigenspace $\mathbf{U}_{x,r}$ can be found, see e.g. [1], as the minimizer of

$$\hat{\mathbf{C}} = \arg\min (t+1)^{-1} \textstyle\sum_{\tau=0}^t \|\mathbf{x}_\tau - \mathbf{C}^T\mathbf{C}\mathbf{x}_\tau\|_2^2, \qquad (1)$$

with respect to (wrt) $\mathbf{C} \in \mathbb{R}^{r\times p}$. It turns out that the r principal eigenvectors of the sample-average covariance estimate $\hat{\mathbf{\Sigma}}_{x,t} = (t+1)^{-1}\sum_{\tau=0}^t \mathbf{x}_\tau\mathbf{x}_\tau^T$, namely $\hat{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}$, form a minimizer for (1). After setting $\mathbf{y}_\tau = \mathbf{C}\mathbf{x}_\tau$, we use (1) to arrive at

$$(\check{\mathbf{C}}, \{\check{\mathbf{y}}_\tau\}_{\tau=0}^t) = \arg\min_{\mathbf{C},\mathbf{y}_\tau} (t+1)^{-1}\textstyle\sum_{\tau=0}^t \|\mathbf{x}_\tau - \mathbf{C}^T\mathbf{y}_\tau|_2^2, \quad (2)$$

which will be the starting point to create a separable PCA cost that is amenable to distributed minimization. Applying first-order optimality conditions in (2) yields that $\check{\mathbf{y}}_\tau = (\check{\mathbf{C}}\check{\mathbf{C}}^T)^{-1}\check{\mathbf{C}}\mathbf{x}_\tau$ which

Work in this paper is supported by the NSF grant CCF 1218079 and UTA.

is different from $\check{\mathbf{y}}_\tau = \check{\mathbf{C}}\mathbf{x}_\tau$. Nonetheless, both (2) and (1) attain the same minimum value when $\check{\mathbf{C}}$ in (2) is selected such that $\check{\mathbf{C}}^T(\check{\mathbf{C}}\check{\mathbf{C}}^T)^{-1}\check{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}\hat{\mathbf{U}}_{x,r}^T$. Now let $\check{\mathbf{C}} = \mathbf{U}_c\mathbf{S}_c\mathbf{V}_c^T$ denote the singular value decomposition of $\check{\mathbf{C}}$, where $\mathbf{U}_c \in \mathbb{R}^{r\times r}$ and $\mathbf{V}_c \in \mathbb{R}^{p\times p}$ contain the left and right singular vectors of $\check{\mathbf{C}}$, while diagonal matrix $\mathbf{S}_c$ contains the singular values. The last two equations indicate that r left singular vectors of $\check{\mathbf{C}}$ corresponding to the largest singular values, say $\mathbf{U}_{c,r}$, satisfy $\mathbf{U}_{c,r} = \hat{\mathbf{U}}_{x,r}\mathbf{W}$ where $\mathbf{W}$ is an arbitrary $r \times r$ unitary matrix. Thus, $\mathbf{U}_{x,r}$ can be estimated from $\check{\mathbf{C}}$ in (2) up to a unitary matrix ambiguity.

3. DISTRIBUTED PCA

In order to develop a distributed (D-) PCA algorithm we will rewrite the centralized PCA cost in (2) in a separable way and then employ the alternating direction method of multipliers (ADMM) [11, 13, 14] combined with block coordinate descent techniques, see e.g., [12], to split the optimization problem into smaller subtasks that can be implemented in parallel across sensors. Towards this end, the cost in (2) can be rewritten as

$$J(\mathbf{C}, \{\mathbf{y}_\tau\}_{\tau=0}^t) = (t+1)^{-1}\sum_{j=1}^p\sum_{\tau=0}^t(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_\tau)^2, \quad (3)$$

where $\mathbf{C}_{:j}$ denotes the jth column of $\mathbf{C}$ for $j = 1, \ldots, p$. Sensor j is responsible for forming updates for $\mathbf{C}_{:j}$ that estimates the jth row of $\mathbf{U}_{x,r}$. Since summands in (3) are coupled through the vectors $\mathbf{y}_\tau$, separate minimization of $\sum_{\tau=0}^t(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_\tau)^2$ at sensor j will not return the minimizer of (2). A separable PCA formulation that is equivalent to the centralized minimization problem in (2) is obtained by introducing the auxiliary vectors $\mathbf{y}_{\tau,j}$ for each sensor j and impose the consensus constraint $\mathbf{y}_{\tau,1} = \mathbf{y}_{\tau,2} = \ldots = \mathbf{y}_{\tau,p}$. The following *separable* constrained optimization problem is obtained

$$(\check{\mathbf{C}}, \{\check{\mathbf{y}}_{\tau,j}\}_{j=1,\tau=0}^{p,t}) = \arg\min(t+1)^{-1}\sum_{j=1}^p\sum_{\tau=0}^t(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_{\tau,j})^2,$$
$$\text{s. to } \mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,j'}, \quad j' \in \mathcal{N}_j \text{ and } \tau = 0, \ldots, t \quad (4)$$

Since the network is connected it follows readily that (4), (3) and (2) are equivalent in the sense that $\{\check{\mathbf{y}}_{\tau,j} = \check{\mathbf{y}}_\tau\}_{j=1}^J$.

3.1. Algorithmic Construction

In order to solve (4) in a distributed fashion we employ ADMM which will allow sensor j to obtain iteratively estimates for the jth row of $\mathbf{U}_x$ via $\mathbf{C}_{:j}$. To facilitate utilization of ADMM, consider the auxiliary variables $\mathbf{z}_{\tau,j}^{j'}$ for $j' \in \mathcal{N}_j$ and $\tau = 0, \ldots, t$. Then, substitute the constraints in (4) with the equivalent ones

$$\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j}^{j'} \text{ and } \mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j'}^{j} \text{ for } j' \in \mathcal{N}_j \text{ and } j \neq j'. \quad (5)$$

The variables $\mathbf{z}_{\tau,j}^{j'}$ are just used to derive the local recursions run across sensors to find $\check{\mathbf{C}}_{:j}$, and eventually these variables are eliminated. Next, let $\mathbf{v}_{\tau,j}^{j'}$ and $\mathbf{w}_{\tau,j}^{j'}$ denote the multipliers associated with the constraints $\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j}^{j'}$ and $\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j'}^{j}$ respectively. ADMM exploits the decomposable structure of the augmented Lagrangian function [12, Ch. 3] which for (4) is written as

$$\mathcal{L}[\mathbf{C}, \{\mathbf{y}_{\tau,j}\}_{\tau=0,j=1}^{t,p}, \mathbf{v}, \mathbf{w}] = (t+1)^{-1}\sum_{j=1}^p\sum_{\tau=0}^t(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_{\tau,j})^2$$
$$+\sum_{j=1}^p\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^t\left[(\mathbf{v}_{\tau,j}^{j'})^T(\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j}^{j'}) + (\mathbf{w}_{\tau,j}^{j'})^T(\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j'}^{j})\right]$$
$$+0.5c\sum_{j=1}^p\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^t\left[\|\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j}^{j'}\|_2^2 + \|\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j'}^{j}\|_2^2\right], \quad (6)$$

where c is a positive penalty coefficient, while $\mathbf{v}$ and $\mathbf{w}$ contain the multipliers $\mathbf{v}_{\tau,j}^{j'}$ and $\mathbf{w}_{\tau,j}^{j'}$, respectively for $\tau = 0, \ldots, t$, $j' \in \mathcal{N}_j$ and $j = 1, \ldots, p$. In order to tackle (4) we first employ a block coordinate descent where we first minimize wrt $\mathbf{C}$ while treating the $\mathbf{y}_{\tau,j}$'s fixed and vice versa. ADMM will be utilized when minimizing (4) wrt to $\mathbf{y}_{\tau,j}$, while respecting the consensus constraints in (5). When minimizing wrt $\mathbf{C}$ assuming fixed $\mathbf{y}_{\tau,j}$ one iteration will be enough since $\check{\mathbf{C}}$ will be found in closed form. However, application of ADMM to determine $\mathbf{y}_{\tau,j}$ for a fixed $\mathbf{C}$, while respecting the equality constraints in (4), will involve multiple iterations denoted as K. Multiple recursions (ideally $K \to \infty$) in ADMM will be needed to enforce the consensus requirement across the $\mathbf{y}_{\tau,j}$ variables [14].

Now let $\kappa = 0, 1, \ldots,$ denote the index for a coordinate descent cycle and $k = 1, \ldots, K$ indicate the consensus iteration index within a coordinate cycle. Then $\kappa \cdot K + k$ enumerates the total number of consensus iterations from the beginning after k consensus iterations have been completed during the κth coordinate descent cycle. One coordinate descent cycle will entail one iteration per sensor to update $\mathbf{C}_{:j}$'s, and K consensus iterations associated with ADMM. Specifically, let $\mathbf{y}_{\tau,j}(\kappa K + K)$ and $\mathbf{z}_{\tau,j}^{j'}(\kappa K + K)$ indicate the most recent updates for $\mathbf{y}_{\tau,j}$ and $\mathbf{z}_{\tau,j}^{j'}$ respectively, after K consensus iterations have been completed during coordinate cycle κ. Minimization of (4) wrt to $\mathbf{C}_{:j}$ during coordinate descent cycle $\kappa + 1$, while treating $\mathbf{y}_{\tau,j}$ as fixed and equal to $\mathbf{y}_{\tau,j}(\kappa K + K)$, gives that the jth column of $\mathbf{C}$ can be updated at sensor j as

$$\mathbf{C}_{:j}(\kappa+1) = \left[\sum_{\tau=0}^t \mathbf{y}_{\tau,j}((\kappa+1)K)(\mathbf{y}_{\tau,j}((\kappa+1)K))^T\right]^{-1}$$
$$\times \sum_{\tau=0}^t \mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j). \quad (7)$$

To make the notation more compact let $\mathbf{y}_{\tau,j}^\kappa(k) := \mathbf{y}_{\tau,j}(\kappa K + k)$, $\mathbf{v}_{\tau,j}^{\kappa,j'}(k) = \mathbf{v}_{\tau,j}^{j'}(\kappa K + k)$, $\mathbf{w}_{\tau,j}^{\kappa,j'}(k) = \mathbf{w}_{\tau,j}^{j'}(\kappa K + k)$ and $\mathbf{z}_{\tau,j}^{\kappa,j'}(k) = \mathbf{z}_{\tau,j}^{j'}(\kappa K + k)$. Then, updates $\mathbf{y}_{\tau,j}^{\kappa+1}(k)$ will be formed at sensor j for $k = 1, \ldots, K$ by employing the ADMM. The first step in ADMM, during coordinate descent cycle $\kappa + 1$, updates the Lagrange multipliers using the gradient ascent iterations

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + c[\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)], \quad (8)$$
$$\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k-1) + c[\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)], \quad (9)$$

for $j' \in \mathcal{N}_j$, $k = 1, \ldots, K$ and $\tau = 0, \ldots, t$. Note that $\mathbf{v}_{\tau,j}^{\kappa+1,j'}(0) = \mathbf{v}_{\tau,j}^{\kappa,j'}(K)$, thus coordinate cycle $\kappa + 1$ starts using the most up-to-date $\mathbf{v}$'s from cycle κ. The same holds for the $\mathbf{y}$'s, $\mathbf{w}$'s and $\mathbf{z}$'s. The second step entails minimization of (6) wrt $\mathbf{y}_{\tau,j}$ while treating the rest optimization variables as fixed to their most up-to-date value. It follows that (details omitted due to space limitations) that for $j = 1, \ldots, p$

$$\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = \left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + c|\mathcal{N}_j|\mathbf{I}\right]^{-1}$$
$$\times\left[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k))\right.$$
$$\left.+c\sum_{j'\in\mathcal{N}_j}(\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{z}_{\tau,j'}^{\kappa+1,j}(k))\right]. \quad (10)$$

The third step in ADMM involves forming the updates $\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)$. This is done by minimizing (6) wrt $\mathbf{z}_{\tau,j}^{\kappa+1,j'}$, while fixing the other variables to their most up-to-date values. Then, it follows

$$\begin{aligned}\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1) =& 0.5[\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k+1)] \\ &+ 0.5c^{-1}[\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k)], \quad (11)\end{aligned}$$

where $j = 1, \ldots, p$ and $j' \in \mathcal{N}_j$. Substituting (11) into the two recursions in (8), it follows that if the Lagrange multipliers are initialized such that $\mathbf{v}_{\tau,j}^{0,j'}(0) = -\mathbf{w}_{\tau,j'}^{0,j}(0)$, then $\mathbf{v}_{\tau,j}^{\kappa,j'}(k) = -\mathbf{w}_{\tau,j'}^{\kappa,j}(k)$ for all τ, κ and k, while

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + 0.5c(\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{y}_{\tau,j'}^{\kappa+1}(k)), \quad (12)$$

for $j' \in \mathcal{N}_j$. Thus, sensor j only has to keep track of $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j'\in\mathcal{N}_j}$ since $\mathbf{w}_{\tau,j}^{\kappa,j'}(k) = -\mathbf{v}_{\tau,j'}^{\kappa,j}(k)$ becomes redundant. Then, using (11) and $\mathbf{v}_{\tau,j}^{\kappa,j'}(k) = -\mathbf{w}_{\tau,j'}^{\kappa,j}(k)$ recursion (10) becomes

$$\begin{aligned}\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) =& \left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times \Big[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) - \mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)) \\ &+c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{\tau,j}^{\kappa+1}(k) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k))\Big], \quad (13)\end{aligned}$$

where the $\mathbf{z}_{\tau,j}^{\kappa,j'}(k)$ variables have been eliminated. Using the convergence claims in [14] turns out that if an infinite number of consensus iterations ($K \to \infty$) is applied during coordinate cycle $\kappa+1$ then consensus is reached in the sense that $\lim_{k\to\infty}\mathbf{y}_{\tau,j}^{\kappa+1}(k) = (\mathbf{C}(\kappa+1)\mathbf{C}(\kappa+1)^T)^{-1}\mathbf{C}(\kappa+1)\mathbf{x}_\tau$ (the principal component vectors) for all sensors $j = 1, \ldots, p$, while $\mathbf{C}(\kappa+1) := [\mathbf{C}_{:1}(\kappa+1)\ldots\mathbf{C}_{:p}(\kappa+1)]$.

Recursions (7), (12) and (13) constitute a *batch* D-PCA approach, whereby sensor j keeps track of i) $\mathbf{C}_{:j}(k+1) \in \mathbb{R}^{r\times 1}$, that estimates the jth row of $\mathbf{U}_{x,r}$; ii) the multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j'\in\mathcal{N}_j}$; and iii) the principal components in $\mathbf{y}_{\tau,j}^{\kappa}(k+1)$ for $\tau = 0, \ldots, t$. In a batch setting, sensors first gather $t+1$ measurements (t is fixed), namely $\mathbf{x}_0, \ldots, \mathbf{x}_t$, and then employ the D-PCA algorithm. During coordinate descent cycle $\kappa+1$ sensor j first forms $\mathbf{C}_{:j}(\kappa+1)$ via (7), using its local principal components vector updates $\mathbf{y}_{\tau,j}^{\kappa}(K)$ for $\tau = 0, \ldots, t$. Then, sensor j runs K consensus iterations by carrying out (12) and (13). Specifically, during consensus iteration $k+1$ and coordinate cycle $\kappa+1$, it receives from its neighbors $j' \in \mathcal{N}_j$ the $r \times 1$ vectors $\mathbf{y}_{\tau,j'}^{\kappa+1}(k)$ and updates its multipliers $\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k)$ via (12). Then, sensor j receives $\{\mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)\}_{j'\in\mathcal{N}_j}$ which are used along with $\{\mathbf{y}_{\tau,j'}^{\kappa+1}(k)\}_{j'\in\mathcal{N}_j}$ to form $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ via (13). For an increasing number of consensus iterations $K \to \infty$ and coordinate descent cycles ($\kappa \to \infty$), $\mathbf{C}(\kappa)$ and $\mathbf{y}_{\tau,j}(\kappa)$ converge at least to a stationary point of the PCA cost in (3). This can be established using the convergence properties of block coordinate descent and ADMM in [12] and [11, 14], respectively.

3.2. Adaptive D-PCA

The batch D-PCA algorithm derived earlier is ideal for settings where estimation performance is more important than real-time information processing. Further, it should be pointed out that there are $(t+1)K$ consensus iterations per coordinate cycle that every sensor has to carry out for updating $\{\mathbf{y}_{\tau,j}^{\kappa}\}_{\tau=0}^{t}$. In a setting where the number of sensor measurements $t+1$ is fixed, the previous property will not cause problems. However, in a time-varying setting where sensors are continuously acquiring data, t will keep increasing, causing challenges to the applicability of batch D-PCA due to the increasing memory, communication and computational requirements. Building on batch D-PCA we derive an adaptive D-PCA scheme that is capable to process data online, while having a manageable computational, communication and memory cost.

In batch D-PCA the separable PCA cost in (4) is time-invariant, i.e. t is fixed. For a constant t batch D-PCA consisted of multiple coordinate descent cycles that could be run until, e.g., the PCA cost does not decrease below a desired threshold. In an adaptive setting sensors acquire new data at every time instant t, thus t is increasing and the cost (4) will be augmented with new data terms. Thus, it is pertinent at t to apply a small number of coordinate cycles; here one coordinate cycle per t is employed. Thus, the time and coordinate cycle indices coincide, i.e., $\kappa = t$. During t there will be K nested consensus iterations carrying out (12) and (13). Different from batch D-PCA, (12) and (13) will be carried out only for the most recent multipliers and principal vectors $\{\mathbf{v}_{t,j}^{j'}, \mathbf{y}_{t,j}\}_{j=1}^{p}$, and not for all $\tau = 0, \ldots, t$ as in the batch implementation.

During $t+1$, adaptive D-PCA updates $\mathbf{C}_{:j}$ as

$$\mathbf{C}_{:j}(t+1) = \left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}(K)\mathbf{y}_{\tau,j}^T(K)\right]^{-1}\textstyle\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}(K)x_\tau(j), \quad (14)$$

and employs K consensus iterations for updating $\mathbf{y}_{t,j}$ and $\mathbf{v}_{t,j}$ as

$$\mathbf{v}_{t+1,j}^{j'}(k) = \mathbf{v}_{t+1,j}^{j'}(k-1) + 0.5c[\mathbf{y}_{t+1,j}(k) - \mathbf{y}_{t+1,j'}(k)], \quad (15)$$

$$\begin{aligned}\mathbf{y}_{t+1,j}(k+1) =& \left[2\mathbf{C}_{:j}(t+1)(\mathbf{C}_{:j}(t+1))^T + c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times \Big[2\mathbf{C}_{:j}(t+1)x_{t+1}(j) - \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{t+1,j}^{j'}(k) - \mathbf{v}_{t+1,j'}^{j}(k)) \\ &+c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{t+1,j}(k) + \mathbf{y}_{t+1,j'}(k))\Big], \quad (16)\end{aligned}$$

where $k = 1, \ldots, K$ for (15) and $k = 0, 1, \ldots, K-1$ for (16), while coordinate cycle superscripts have been removed since one cycle takes place per t. The multipliers at $t = 0$, namely $\mathbf{v}_{j,0}(0)$, are initialized randomly, whereas at time instant $t > 0$ warm-starts are employed to set $\mathbf{v}_{t,j}(0) = \mathbf{v}_{t-1,j}(K)$. Further, during instant t the $\mathbf{y}_{t,j}$ variables are initialized as $\mathbf{y}_{t,j}(0) = \mathbf{C}_{:j}(t)x_t(j)$. Let $\mathbf{M}_{x,t}$ denote the matrix inverted in (14), and $\mathbf{m}_{xy,t} := \sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}(K)x_\tau(j)$. Since consensus is applied only for $\mathbf{y}_{t,j}$ and $\mathbf{v}_{t,j}^{j'}$, the updates $\mathbf{y}_{\tau,j}(K)$ and $\mathbf{v}_{\tau,j}^{j'}(K)$ for $\tau < t$ remain constant for the time instances $\tau+1, \tau+2, \ldots$. Thus, $\mathbf{M}_{x,t}$ and $\mathbf{m}_{xy,t}$ can be adaptively updated at sensor j as $\mathbf{M}_{x,t} = \mathbf{M}_{x,t-1} + \mathbf{y}_{t,j}(K)\mathbf{y}_{t,j}^T(K)$ and $\mathbf{m}_{xy,t} = \mathbf{m}_{xy,t-1} + \mathbf{y}_{t,j}(K)x_t(j)$, respectively.

Summarizing, adaptive D-PCA will involve the following steps during $t+1$: Sensor j updates recursively $\mathbf{M}_{x,t}$ and $\mathbf{m}_{xy,t}$ and uses them to update $\mathbf{C}_{:j}(t+1)$ via (14). Then, K consensus recursions are employed in order to obtain $\mathbf{y}_{t+1,j}(K)$. To this end, the initialization $\mathbf{y}_{t+1,j}(0) = \mathbf{C}_{:j}(t+1)x_{t+1}(j)$ and $\mathbf{v}_{t+1,j}(0) = \mathbf{v}_{t,j}(K)$ takes place. During consensus iteration k sensor j receives from its neighbors $j' \in \mathcal{N}_j$ vectors $\mathbf{y}_{t+1,j'}(k)$ and updates its multipliers $\mathbf{v}_{t+1,j}^{j'}(k)$ via (15). Then, sensor j receives the multipliers $\{\mathbf{v}_{t+1,j'}^{j}(k)\}_{j'\in\mathcal{N}_j}$ which are used along with $\{\mathbf{y}_{t+1,j'}(k)\}_{j'\in\mathcal{N}_j}$ to form $\mathbf{y}_{t,j}(k+1)$ via (16). Once $\mathbf{y}_{t+1,j}(K)$ are formed across sensors, the process is repeated.

4. COMMUNICATION AND COMPUTATIONAL COSTS

Next, we study the communication and computational costs associated with the adaptive D-PCA scheme, summarized in (14), (15) and (16), and compare it with the related approach in [10]. The computational complexity for carrying out (14) is $\mathcal{O}(r^2)$ which is dictated by the inversion of $\mathbf{M}_{x,t}$ which can be done by employing the matrix inversion lemma [15, pg. 571]. Updating of $\mathbf{v}_{t,j}^{j'}$ and $\mathbf{m}_{xy,t}$ has a complexity of the order of $\mathcal{O}(r)$, while the associated complexity for forming $\mathbf{y}_{t,j}$ is $\mathcal{O}(r^2)$ which is imposed by the matrix inversion that again can be carried out using the matrix inversion lemma. Thus, the computational complexity per time instant t and consensus iteration in adaptive D-PCA is $\mathcal{O}(r^2)$. The computational complexity of the algorithm proposed in [10], abbreviated as DLS-PCA, is also $\mathcal{O}(r^2)$.

Next, we consider the communication costs associated with D-PCA and DLS-PCA. In D-PCA, sensor j has to transmit $r(|\mathcal{N}_j|+1)$ scalars per consensus iteration corresponding to the entries of the multipliers $\{\mathbf{v}_{t,j}^{j'}(k)\}_{j\in\mathcal{N}_j}$ and the local estimate $\mathbf{y}_{t,j}(k)$. Given that at each time instant t there are K consensus iterations taking place, the total transmission load per sensor during time instant t is $rK(|\mathcal{N}_j|+1)$. In DLS-PCA each sensor has to transmit $rK(r+1)$ to carry out consensus-iterations involving $r\times r$ matrices and $r\times 1$ vectors. If $r<|\mathcal{N}_j|$ is smaller than the size of the single-hop neighborhoods then DLS-PCA has a smaller transmission cost, whereas if $r>|\mathcal{N}_j|$ D-PCA will have an advantage. When considering the reception cost it can be seen that in D-PCA each sensor receives $2rK|\mathcal{N}_j|$ scalars during K consensus iterations pertaining to the vectors $\{\mathbf{v}_{t,j'}^{j}(k),\mathbf{y}_{t,j'}(k)\}_{j'\in\mathcal{N}_j}$. However, in DLS-PCA sensor j receives $(r+1)rK|\mathcal{N}_j|\geq 2rK|\mathcal{N}_j|$ due to the reception of $\mathcal{N}_j$ $r\times r$ matrices and $r\times 1$ vectors per consensus iteration. Thus, D-PCA has a smaller reception cost for $r>1$. Even though the transmission cost of D-PCA may be greater than the one in DLS-PCA, it will be corroborated via numerical examples that the higher transmission cost in D-RLS pays off in improved convergence rates and steady-state performance. The work in [14] thoroughly studied why ADMM exhibits better convergence properties over consensus-averaging [7] when it comes to least-squares estimation [14]. Similar arguments can be carried over the present PCA setting supporting the better convergence properties of ADMM observed via simulations.

5. NUMERICAL TESTS

Here we compare the performance achieved by the novel adaptive D-PCA approach with the one corresponding to DLS-PCA in [10]. The subspace projection estimation error $e(t):=\|\mathbf{C}^T(t)(\mathbf{C}(t)\mathbf{C}^T(t))^{-1}\mathbf{C}(t)-\mathbf{U}_{x,r}\mathbf{U}_{x,r}^T\|_F^2$ with $\|\cdot\|_F$ denoting the Frobenius norm, is used as a performance metric. Such a metric quantifies how 'close' the columns of the estimate $\mathbf{C}^T(t)$ lie on the principal subspace spanned by the columns of $\mathbf{U}_{x,r}$. A connected network consisting of $p=16$ sensors, randomly placed in $[0,1]\times[0,1]$, is considered. Two sensors communicate as long as their distance is less than $d=0.3$ (communication range). The columns $\mathbf{C}_{:j}(0)$ and multipliers $\mathbf{v}_j^{j'}(0)$ are initialized randomly. The covariance matrix $\mathbf{\Sigma}_x$ is randomly generated as $\mathbf{H}\mathbf{H}^T$, where $\mathbf{H}$ contains normally distributed entries. Fig. 1 (top) depicts $e(t)$ for the case where D-PCA and DLS-PCA are estimating $r=1$ principal eigenvector of $\mathbf{\Sigma}_x$. D-PCA and DLS-PCA were tested for either $K=5$, or $K=12$ consensus iterations per time instant t. The parameter c in D-PCA is set $c=4$, while the step-size parameter needed in DLS-PCA was set to $\gamma=10^{-3}$. Both parameters were determined via numerical trials such that both approaches achieve the best possible convergence rate and steady-state performance. A systematic approach for selecting c is underway. As it can be seen D-PCA exhibits a much faster convergence rate than DLS-PCA, while it converges to a lower steady state error $e(t)$. As expected, increasing the number of consensus iterations K leads to better steady-state estimation performance for both D-PCA and DLS-PCA. Fig. 1 (bottom) depicts $e(t)$ for $r=2$. In this case we test D-PCA and DLS-PCA for either $K=5$, or $K=15$ consensus iterations. Again $e(t)$ in adaptive D-PCA decreases at a faster rate than the DLS-PCA approach, while the subspace estimation performance improves considerably as the number of consensus iterations K increases. In order to boost the convergence rate of DLS-PCA we increased γ to 10^{-2}. The convergence rate improved, however the magenta curve indicates considerable fluctuations of $e(t)$.

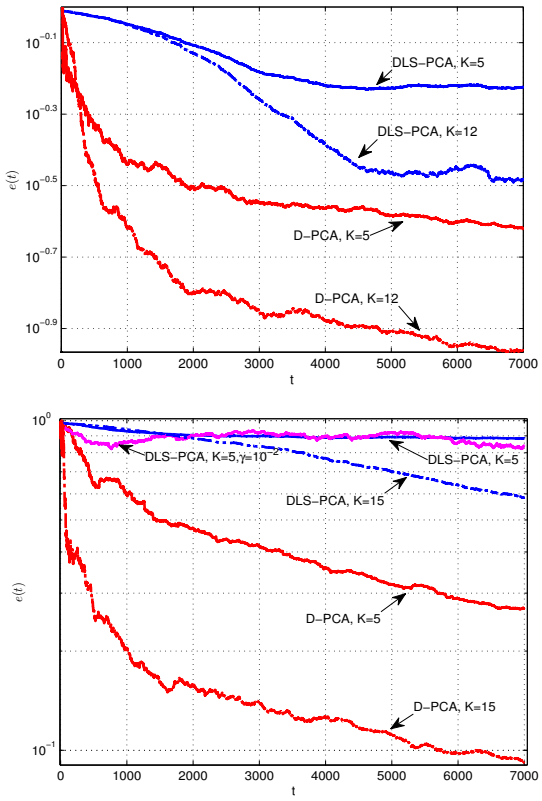

Fig. 1. Subspace projection estimation error $e(t)$ vs. time index t for r=1 (top); and r=2 (bottom).

6. CONCLUDING REMARKS

A distributed PCA scheme was developed for recovering the principal data covariance eigenspace across sensors. The approach followed entailed: i) Formulation of the PCA cost as a separable constrained optimization problem; and ii) application of the coordinate descent technique combined with ADMM to tackle the separable PCA minimization formulation in a distributed fashion. Computationally simple local updating recursions were obtained that involve only single-hop inter-sensor communications. Numerical examples demonstrated the improved performance of the novel algorithm over existing alternatives.

7. REFERENCES

[1] D. R. Brillinger, *Time Series: Data Analysis and Theory*, Expanded Edition, Holden Day, 1981.

[2] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. on Sig. Processing*, vol. 43, no. 1, pp. 95–107, 1995.

[3] T. Hastie, R. Tibshirani, and D. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Second Edition, Springer, 2009.

[4] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Applicat.*, vol. 106, no. 1, pp. 69–84, 1985.

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

[6] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *Proc. of 11th IEEE Workshop on Sig. Proc. Advances in Wir. Com. (SPAWC)*, June 2010, pp. 1 –5.

[7] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, Sept. 2004.

[8] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Par. and Distr. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.

[9] Z. Meng, A. Wiesel, and A. O. Hero, "Distributed principal component analysis on networks via directed graphical models," in *in Proc. of IEEE Intl. Conf. on Acoust., Speech and Sig. Proc*, March 2012, pp. 2877 –2880.

[10] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensors networks," *IEEE Journal of Sel. Topics in Sig. Proc.*, vol. 5, no. 4, pp. 725–738, 2011.

[11] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed. Belmont, MA: Athena Scientic, 1999.

[12] D. P. Bertsekas, *Nonlinear Programming*, Second Edition, Athena Scientific, 2003.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[14] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc wsns with noisy links- Part I: Distributed estimation of deterministic signals," *IEEE Trans. on Sig. Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[15] S. M. Kay, *Fundamental of Statistical Signal Processing: Estimation Theory*, Prentice Hall, 1993.