SPARSE DECENTRALIZED PRINCIPAL COMPONENTS ANALYSIS FOR DIMENSIONALITY REDUCTION

by

NANRUO CHEN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

Master of Engineering in Electrical Engineering

THE UNIVERSITY OF TEXAS AT ARLINGTON

MAY 2016

To my parents CHEN JUN and CHEN LIQIN

who sacrificed so much for me.

ACKNOWLEDGEMENTS

A lot of people have contributed to the success of my thesis. I learnt a lot from their guidance which will benefit me in the future.

Firstly, I would like to thank my supervising Professor; Dr. I. D. Schizas for his mentorship. His advice really helped me a lot in my research.

I also wish to thank Dr. Kamisetty R. Rao and Dr. W. Alan Davis for taking time out of their busy schedule to serve on my thesis defense committee.

April 21, 2016

ABSTRACT

SPARSE DECENTRALIZED PRINCIPAL COMPONENTS ANALYSIS FOR DIMENSIONALITY REDUCTION

Nanruo Chen, M.Engr.

The University of Texas at Arlington, 2016

Supervising Professor: Ioannis D. Schizas

Principal components analysis (PCA) is a data compression technology relying on dimensionality reduction. In a wireless sensor network, the acquired data may be spatially scattered and include many zero variables, for which a standard PCA approach cannot account for. To this end, a new algorithm is designed to solve both problems. We combine sparse principal components analysis (SPCA) and distributed principal components analysis (DPCA) together to obtain a sparse distributed principal components analysis (SDPCA) algorithm. Norm-one regularization along with the alternating direction method of multipliers (ADMM) is used for SPCA. ADMM is also employed to obtain a distributed compression algorithm that consists of computationally simple local updating recursions. Further, inter-sensor communication noise is considered. Numerical tests using both synthetic and real data demonstrate that the novel SDPCA algorithm can be applied in different siutations and gives a good principal subspace estimation result.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

A wireless sensor network (WSN) generally refers to a group of sensors which are deployed in an area for applications such as environmental monitoring, or target tracking [1]. For example, temperature and pressure monitoring, or movements of animals tracking [2]. The sensor nodes are connected via wireless radio and have an instrumental role in the system. A sensor node in a WSN generally has functions such as data collection, computation, transmission and storage [3]. Sensor nodes use their sensed data to acquire environmental information and convert it to digital signals. Then, depending on the system specifications, the sensors do computations on the acquired data before transmission or storage. Further, sensors communicate with other neighboring sensors, or communicate with a central fusion center such as a satellite or base station. These sensor nodes also have storage to preserve the acquired data, as well as software running in them and implementing various signal processing tasks.

Wireless sensor network have several limations need to be accounted for:

1. Energy is limited

Sensor nodes are quite small, thus can only carry a limited power supply. Sensor nodes are always distributed in large and complex areas, thus replacing batteries across a large number of sensors is not a good idea [4]. Energy efficiency is one of the main challenges in a sensor network.

2. Communcaiton ability is limited

The relation of required energy $\mathbf{E}$ and communication distance d is given by

$$\mathbf{E} \propto d^{\alpha} \tag{1.1}$$

Where α is the coefficient which for wireless communication is $\alpha \approx 2$ [5]. This equation shows that the energy fading will increase with square if the inter-sensor distance increases. The channel bandwidth is another problem which will limit the communication. The wireless communication channel is not wide. Thus information exchanges can not take place at high rates.

3. Computational ability is limited

The computational ability of sensors is not powerful, but a network of sensors is more powerful.

Networks of sensors follow different topologies depending on the application at hand. Main sensor network topologies include fully connected, mesh, bus, star, ring and tree [6].

1. Fully connected: Each sensor is allowed to communicate directly with all other sensors. The sensors have both abilities to transmit and receive. This network is quite reliable due to full connectivity, but it is too complex to be implemented, especially in the presence of a large number of sensors.

2. Mesh networks: A mesh topology implies that nodes could have different paths to connect with other sensors which enables robust multi-hop communications. It is highly robust and more scalable, but will consume high power and have high latencies.

3. Bus topology: In bus networks, each message sent by one sensor can be received by all the other sensors through a single channel. It is easy to install, but may have serious congestion, and it is preferred for a small number of sensors.

4. Star topology: In a star topology, each sensor cannot directly communicate with others. All sensors communicate with a centralized hub (fusion center). The fusion center will receive information from all sensors. Star topology consumes low power and can be easily enlarged, but it is not robust when the fusion center fails.

5. Ring topology: In a ring network, each sensor can only communicate with two neighbors, and all of them perform the same function. The speed of communication and information propagation is slow.

6. Tree topology: The network is divided into several levels. It is a combination of the star topology and the bus topology. The network uses a fusion center as a root node and nodes in lower levels propagate information to higher levels. It consumes less power than other topologies, but is also slow and not reliable.

1.1 Distributed sensor networks

Different from centralized wireless sensor networks, distributed topologies do not have a central fusion center. In centralized WSNs, if the fusion fails, the whole network will collapse. Distributed sensor networks are robust, have better data sensing abilities and also provide backup nodes [7].

Distributed sensor networks have to operate under the following conditions [8]:

1. Due to limited power budget, sensors can only communicate with sensors in their close vicinity, known as neighboring sensors.

2. The sensed environment may be time-vaying, requiring the sensors to process information fast and efficiently.

3. Individual nodes may easily fail.

4. There is a requirement for a changing adaptive network topology.

In order to match the aforementioned conditions, the distributed sensor network should have the following features [9]:

1. A sufficiently large number of sensors is needed for improved accuracy, and robustness when certain sensors fail.

2. The sensor network topology should be dynamic to match the presence of malfunctioning sensors, or the incorporation of new sensors in the network.

3. The sensors should be able to identify their neighboring sensors and collaborate with them forming a well-connected network.

1.2 Principal component analysis

Distributed sensor networks are challenged by the limited power buget which can limit the network's life span. One way to address the issue of limited energy is to reduce the dimensionality of the data acquired by sensors before being transmitted to a remote site. This solution is useful when the network acquires data continuously [10]. To this end, our goal is to compress the data inside the network using sensors. A data compression technology applied to wireless sensor networks has the following steps. First, apply a data compression algorithm. Then, transmit the compressed data to a remote site, and at last decompress the data and reconstructed it. A lot of research has been performed in data compression. Here we will focus on principal component analysis (PCA) for compresssing the data in distributed sensor networks.

Principal component analysis (PCA) is a data compression technology. It can reduce the number of observed data variables to a smaller number of principal components if the variables are highly correlated [11]. The original data will be converted to a set of uncorrelated entries, after finding the eigenvectors of the data covariance matrix using the acquired data. This process corresponds to an orthogonal transformation of the original data into principal components. The first principal component will have the largest eigenvalue and therefore the most information about the data, the second will have the second largest eigenvalue and so on. Thus, a large number of data can be efficiently represented by a small number of principal components. Another advantage of principal component

analysis is that the noise sensitivity is low [12], such that it can be applied in data denoising applications.

1.3 Previous work

Centralized PCA has already been developed for different settings, especially in data aggregation methods [13–15]. But most time in sensor networks, the sensor data can not be centralized, they are scattered across different sensors. PCA techniques have been developed for networks assuming that all sensors could connect to a fusion center [16]. But this method requires highly computational capabilities and communication bandwidth. To avoid this, decentralized methods are considered. Partially decentralized PCA algorithms are developed by using local computation or data aggregation [16–19]. But the problem is that a fusion center is still required for data aggregation. A method relying on the Karhunen-Loeve transformation is developed in [20], but it still needs a fusion center and can not converge in some situations. Another method assuming decomposable covariance matrices is developed in [21]. Distributed algorithms relying on in-network processing capabilities were developed in [22, 23]. The method in [23] required a fully connected graph or a special tree structure. The approach in [22] relies on consensus-averaging techniques. To increase the performance and convergence speed, a new algorithm was developed in [24]. Our main goal is to extend the distributed principal component analysis algorithm to exploit sparsity in the principal subspace trying of interest.

1.4 Distributed principal component analysis

Different from traditional PCA, distributed principal component analysis schemes have to be redesigned such that spatially scattered sensors perform the processing. For a distributed sensor network structure, data will not be collected at a fusion center, thus com-

munication between neighboring sensors will be used to estimate the principal covariance eigenspace. Also, in order to save energy, the sensors will only communicate with their neighboring sensors in a limited distance. The alternating direction method of multipliers (ADMM) method will be used here to derive the proposed algorithm.

1.5 Sparse principal component analysis (SPCA)

The principal component analysis framework will be redesigned to solve the sparsity problem. Oftentimes there will be many zero variables in a set of data, for which standard PCA cannot account for [25]. For example, in some images, there will be zero variables in the covariance eigenspace [26]. Futher, if data are properly transformed, say e.g, by the Discrete Cosine Transform (DCT), then many zeros-valued data entries will appear in the transformed domain. The presence of many zeroes in the covariance eigenspace will be used here to design a sparse PCA (SPCA) method that accounts for the zeroes and gives better estimation accuracy [27].

1.6 Contributions of thesis

The contributions of this thesis include:

1. Utilization of sparsity regularization mechanisms to improve subspace estimation accuracy compared to standard PCA approaches when sparse signals are involved.

2. Combination of distributed techniques along with norm-one regularization, to achieve a novel distributed sparse approach that can operate in distributed sensor networks.

3. Develop an algorithm that is robust in the presence of noise, and can efficiently compress sensor data to save bandwidth and energy.

1.7 Outline of work

This research mainly focused on a sparse distributed principal component analysis algorithm for dimensionality reduction in distributed wireless sensor networks. The background and motivation of this sparse distributed algorithm will be introduced, including the introduction of wireless sensor network and principal component analysis. In Chapter 2, the design of this algorithm will be divided into two parts. Sparsity imposing mechanisms will be introduced first, while ADMM will be used to solve a decentralized formulation. Norm-one regularization will be combined with ADMM to devise a pertinent decentralized algorithm. The impact of inter-sensor communication noise will also be considered in Chapter 2. In Chapter 3, extensive numerical tests on both synthetic and real data will demonstrate the good performance of the proposed scheme in different scenarios. The performance will be assessed and compared in different situations. Finally, Chapter 4 will provide some concluding remarks along with future research directions.

CHAPTER 2

A SPARSE DISTRIBUTED PRINCIPAL COMPONENT ANALYSIS FRAMEWORK

Estimating the eigenvector subspace of a data covariance matrix is essential in compressing the data acquired across spatially scattered sensors. Futher, the covariance eigenvectors when properly transformed, using the DCT transform, contain many zero entries that can simplify estimation. To this end, the PCA formulation will be expanded here with a sparsity imposing mechanism, relying on norm-one regularization to find the zero entries and estimate the nonzero entries. The alternating direction method of multiplies will be employed to obtain a distributed compression algorithm that consists of computationally simple local updating recursions. Only neighboring sensors will collaborate and communicate to solve this complex data compression problem.

2.1 Problem Statement

Consider a sensor network with p sensors, where each sensor can only communicate with the neighbors within a radius d. Let j be the identifier for each sensor. The neighbors of sensor j within communication range d are denoted by set $\mathcal{N}_j$. Assuming that the links between sensors are symmetric, the sensor network can be represented as an undirected connected graph. The sensor network can be characterized by an adjacency matrix $\mathbf{E} \in \mathbb{R}^{p\times p}$, where $\mathbf{E}_{ij} = \mathbf{E}_{ji} = 1$ for $\mathbf{i} \in \mathcal{N}_j$ and $\mathbf{E}_{ji} = 0$ if $\mathbf{i} \notin \mathcal{N}_j$. The measurements of each sensor j will be denoted as $\{x_\tau(j)\}_{j=1}^p$ (where τ is the time index). The scattered measurements can be defined as $\mathbf{x}_\tau := [x_\tau(1), \ldots, x_\tau(p)]^T$, the vector that contains all sensor measurements at time instant τ. There is additive zero-mean possibly colored noise w_τ, present in the acquired sensor measurements: x_τ=s_τ+w_τ, where $s_\tau \in \mathbb{R}^{p\times 1}$ is the

original signal without noise. The signal of interest s_t in practice is low-dimensional and can be represented as

$$\mathbf{s}_\tau = \mu_x + \sum_{\rho=1}^{r} \pi_{\tau,\rho}\mathbf{u}_{s,\rho} \qquad t = 1, ..., n, \tag{2.1}$$

where μ_x is the mean of s_t, $\pi_{\tau,\rho}$ denotes the zero-mean independent projection coefficients, and $\{\mathbf{u}_{x,\rho}\}_{\rho=1}^{r}$, is an unknown orthogonal basis of dimension r, while $r \leq p$ [26].

The covariance matrix of x_τ can be written as $\mathbf{\Sigma}_x = \mathbf{\Sigma}_s + \mathbf{\Sigma}_w$. Assuming that the original signal s_τ and noise w_τ are independent, while removing μ_x, the expression (2.1) can be written as

$$\check{x}_\tau := \mathbf{F}x_\tau = \sum_{\rho=1}^{r} \pi_{\tau,\rho}\mathbf{F}\mathbf{u}_{s,\rho} + \mathbf{F}w_\tau \tag{2.2}$$

where $\mathbf{F}$ is a proper transformation matrix. Formula (2.2) indicates that the data can be decomposed according to a linear combination of an orthogonal basis [26]. By choosing the most significant elements, and removing weak elements, the sensor data can be compressed.

Our main goal in this thesis is to estimate the transformed vectors $\check{u}_{s,\rho}$, especially in the presence of many zero entries ($\mathbf{\Sigma}_s$ has sparse eigenvectors) by developing a sparse distributed principal component analysis (SDPCA). Aiming to compress data $\mathbf{x}$, linear dimensionality reduction is performed at the encoder by left-multiplying $\mathbf{x}$ with a matrix $\mathbf{C} \in \mathbb{R}^{r\times p}$, where $r \leq p$ and r is the subspace dimension. Then after $\mathbf{Cx}$ is received at the decoder, in order to reconstuct $\mathbf{x}$, a matrix $\mathbf{B} \in \mathbb{R}^{p\times r}$ will be left-multiplied with $\mathbf{Cx}$ [26]. Because the task is to get a good reconstruction estimate of $\mathbf{x}$, it need to minimize the mean-square error (MSE), with respect to matrices $\mathbf{B}$ and $\mathbf{C}$ according to the following formulation

$$(\mathbf{B}_o, \mathbf{C}_o) \in \arg\min_{\mathbf{B},\mathbf{C}} \mathbf{E}[\|\mathbf{X} - \mathbf{BCX}\|^2]. \tag{2.3}$$

The formulation in (2.3) has a solution given as $\mathbf{B}_o$=$\mathbf{U}_{x,r}$, $\mathbf{C}_o$=$\mathbf{U}_{x,r}^T$. Then, the formula (2.3) can be expressed as

$$\mathbf{C}_o \in \arg\min_{\mathbf{C}} \mathbf{E}[\|\mathbf{X} - \mathbf{C}^T\mathbf{C}\mathbf{X}\|^2]. \tag{2.4}$$

Now, because the ensemble covariance matrices are not available, $\mathbf{C}_o$ and $\mathbf{B}_o$ can not be found. This challenge is resolved by replacing the cost in (2.4) with its sample-averaged vesion $(t+1)^{-1}\arg\min_{\mathbf{C}}[\|\mathbf{X} - \mathbf{C}^T\mathbf{C}\mathbf{X}\|_{\mathbf{F}}^2]$. Then, the cost function in the noiseless case can be written as

$$\mathbf{J}(\mathbf{C}_o) = (t+1)^{-1}\arg\min_{\mathbf{C}}[\|\mathbf{X} - \mathbf{C}^T\mathbf{C}\mathbf{X}\|_{\mathbf{F}}^2] \tag{2.5}$$

where $\mathbf{X} := [\mathbf{x}_0 \dots \mathbf{x}_t]$ and $t+1$ denotes the total number of data available.

2.2 Sparse principal component analysis

To exploit the sparsity (many zeroes) in the covariance subspace, norm-one regularization will be used in PCA in (2.4). The cost of PCA in (2.5) is regularized with norm-one mechanism [26], and it gets the following form

$$\mathbf{C} \in \arg\min_{\mathbf{C}}(t+1)^{-1}[\|\mathbf{X} - \mathbf{C}^T\mathbf{C}\mathbf{X}\|_{\mathbf{F}}^2] + \textstyle\sum_{\rho=1}^{q}\sum_{j=1}^{p}\lambda_\rho(|\mathbf{C}(\rho,j)|), \tag{2.6}$$

where $\{\lambda_\rho\}_{\rho=1}^q$ are sparsity-controlling coefficients adjusting the number of zeroes in $\mathbf{C}$. Since (2.6) is a nonconvex problem and can not be solved efficiently [28], the formulation (2.6) can be written as

$$\mathbf{C} \in \arg\min_{\mathbf{C}}(t+1)^{-1}[\|\mathbf{X} - \mathbf{C}^T\mathbf{C}\mathbf{X}\|_{\mathbf{F}}^2] + \textstyle\sum_{\rho=1}^{q}\sum_{j=1}^{p}\lambda_\rho(|\mathbf{Z}(\rho,j)|), \tag{2.7}$$

$$\text{s. to } \mathbf{C} = \mathbf{Z}.$$

In order to solve (2.7), a Lasso based alternating direction method of multipliers (ADMM) will be introduced to deal with this problem [28]. After setting $\mathbf{y}_\tau = \mathbf{C}\mathbf{x}_\tau$ for $\tau = 0, 1, 2, \ldots, t$, the cost function of $\mathbf{C}$ in (2.7) can be expressed as

$$\mathbf{J}(\mathbf{C}) = (t+1)^{-1} \sum_{j=1}^{p} \sum_{\rho=1}^{q} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T y_{\tau,j})^2 + \sum_{j=1}^{p} \sum_{\rho=1}^{q} \lambda_\rho |\mathbf{Z}(\rho, j)|. \quad (2.8)$$

Further, let $\mathbf{Z}_j = \mathbf{C}_{:j}$, then the augmented Lagrangian function associated with (2.8) can be written as

$$\mathcal{L} = (t+1)^{-1} \sum_{j=1}^{p} \sum_{\rho=1}^{q} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T y_{\tau,j})^2 + \sum_{j=1}^{p} \sum_{\rho=1}^{q} \lambda_\rho |\mathbf{Z}(\rho, j)|,$$

$$+ \sum_{j=1}^{p} \mathbf{u}^T (\mathbf{Z}_j - \mathbf{C}_{:j}) + \sum_{j=1}^{p} 0.5c \|\mathbf{C}_{:j} - \mathbf{Z}_j\|_2^2 \quad (2.9)$$

where c is a positive penalty coefficient.

Let $\kappa = 0, 1, \ldots,$ denote the index for a coordinate descent cycle. Then, applying the Lasso-based ADMM, the problem can be solve by the following updating equation, see e.g., [28]

$$\mathbf{C}_{:j}(\kappa+1) = \arg\min (t+1)^{-1} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T(\kappa) y_{\tau,j})^2 + u^T(\kappa)(\mathbf{C}_{:j}(\kappa) - \mathbf{Z}_j(\kappa))$$
$$+ 0.5c \|\mathbf{C}_{:j}(\kappa) - \mathbf{Z}_j(\kappa)\|_2^2, \quad (2.10)$$

the $\mathbf{Z}(\rho, j)$ entry in $\mathbf{Z}$ is updated as:

$$\mathbf{Z}_{\rho,j}(\kappa+1) = \arg\min \lambda_\rho |\mathbf{Z}_{\rho,j}(\kappa)| + u_\rho^T(\kappa)(\mathbf{C}_{\rho,j}(\kappa+1) - \mathbf{Z}_{\rho,j}(\kappa)) + 0.5c(\mathbf{C}_{\rho,j}(\kappa+1) - \mathbf{Z}_{\rho,j}(\kappa))^2, \quad (2.11)$$

and the multiplier is updated as:

$$\mathbf{u}(\kappa+1) = \mathbf{u}(\kappa) + c(\mathbf{C}_{:j}(\kappa+1) - \mathbf{Z}_j(\kappa+1)) \tag{2.12}$$

Applying first-order optimality conditions with respect to $\mathbf{C}$ on (2.2), the following equation can be obtained

$$-2\mathbf{y}_{\tau,j}\left[x_\tau(j) - (\mathbf{C}_{:j}(\kappa+1))^T\mathbf{y}_{\tau,j}\right] + \mathbf{u}(\kappa) + 0.5 * 2 * \mathbf{c}(\mathbf{C}_{:j}(\kappa+1) - \mathbf{Z}_j(\kappa)) = 0. \tag{2.13}$$

Then, $\mathbf{C}_{:j}$ can be updated as

$$\mathbf{C}_{:j}(\kappa+1) = [(t+1)^{-1}\textstyle\sum_{\tau=0}^{t} 2\mathbf{y}_{\tau,j}\mathbf{y}_{\tau,j}^T + \mathbf{cI}]^{-1} \times [(t+1)^{-1}\textstyle\sum_{\tau=0}^{t} 2\mathbf{y}_{\tau,j}x_{\tau,j} - \mathbf{u} + \mathbf{cZ}_j(\kappa)] \tag{2.14}$$

After applying first-order optimality conditions on $\mathbf{Z}$ in (2.11), the following equation can be obtained

$$\frac{\lambda_\rho|\mathbf{Z}_{\rho,j}(\kappa+1)|}{\mathbf{d}_{\mathbf{Z}_{\rho,j}(\kappa+1)}} - \mathbf{u}_\rho(\kappa) - \mathbf{c}(\mathbf{C}_{\rho,j}(\kappa+1) - \mathbf{Z}_{\rho,j}(\kappa+1)) = 0 \tag{2.15}$$

Then, $\mathbf{Z}_{\rho,j}$ can be updated during iteration κ as:

$$\mathbf{Z}_{\rho,j}(\kappa+1) = sgn(\mathbf{C}_{\rho,j}(\kappa+1) + \frac{\mathbf{u}(\kappa)}{\mathbf{c}}) \times (|\frac{\mathbf{C}_{\rho,j}(\kappa+1) + \mathbf{u}(\kappa)}{c}| - \frac{\lambda}{c})_+ \tag{2.16}$$

where $sgn(\cdot)$ denotes signum function and $(\cdot)_+ = max(\cdot, 0)$.

Then, updating formulas (2.12), (2.14) and (2.16) are used to form an iterative algorithm to update matrix $\mathbf{C}$, which estimates the sparse covariance subspace. Here, $\mathbf{y}_\tau$ is treated as a known parameter. The proposed algorithm is tabulated below as Algorithm 1.

Algorithm 1 is a centralized approach that requires full connectivity of all sensors. So far, we considered $\mathbf{y}$ as a known parameter, but in a distributed sensor network setting a constraint $\mathbf{y}_{\tau,1} = \mathbf{y}_{\tau,2} = \ldots = \mathbf{y}_{\tau,p}$ should be introduced in (2.7) to ensure all sensors find consistent estimates of $\mathbf{y}_\tau$. To satisfy this constraint, the sensors need to communicate

Algorithm 1 Sparse Principal Component Analysis

1: Every sensor j gathers $t+1$ measurements $\{x_j(\tau)\}_{\tau=0}^{t}$.

2: **for** $\kappa = 1, 2, \ldots$ **do**

3: Sensor j updates $\mathbf{C}_{:j}$ via (2.14).

4: Updating of $\mathbf{Z}_{\rho,j}$ via (2.16).

5: Updating of $\mathbf{u}$ via (2.12).

6: **end for**

7: If $(\|\mathbf{C}_{:j}^{\kappa+1} - \mathbf{C}_{:j}^{\kappa}\|_2) \leq \epsilon$ then stop (for desired tolerance ϵ).

data with their neighboring sensors. We will introduce a distributed principal component analysis based on ADMM to deal with this problem in the following section.

2.3 Distributed Principal Component Analysis

Starting with the ensemble covariance matrix whose principal subspace needs to be estimated. The covariance is given as:

$$\mathbf{\Sigma}_x = \mathbb{E}[\mathbf{x}_\tau \mathbf{x}_\tau^T]. \tag{2.17}$$

If the number of data is quite large, equation (2.17) can be approximated as

$$\hat{\mathbf{\Sigma}_x} \approx \frac{1}{t+1} \sum_{\tau=0}^{t} [\mathbf{x}_\tau \mathbf{x}_\tau^T]. \tag{2.18}$$

Performing eigenvalue decomposition on the covariance matrix gives

$$\mathbf{\Sigma}_x = \mathbf{U}_x \mathbf{\Lambda}_x \mathbf{U}_x^T \tag{2.19}$$

where $\mathbf{U}_x \in \mathbb{R}^{p \times p}$ is the square matrix whose columns contain the eigenvectors, and $\mathbf{\Lambda}_x$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues. By principal component analysis, the principal eigenspace $\mathbf{U}_{x,r}$ can be found by estimating the r principal eigenvectors of the covariance matrix $\mathbf{\Sigma}_x$ using the gathered sensor data. The

problem will be solved in a distributed sensor network. Setting $\mathbf{y}_\tau = \mathbf{C}\mathbf{x}_\tau$, the formulation in (2.5) can be written as

$$\{\hat{\mathbf{C}}, \hat{\mathbf{y}}_\tau\} = \arg\min_{\mathbf{C},\mathbf{y}_\tau} (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_\tau - \mathbf{C}^T\mathbf{y}_\tau\|_2^2, \tag{2.20}$$

whose optimal solution corresponds to principal eigenspace of $\hat{\mathbf{\Sigma}}_x$ in (2.18), namely $\hat{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}$. Applying first-order optimality in (2.20), $\hat{\mathbf{y}}_\tau = (\hat{\mathbf{C}}\hat{\mathbf{C}}^T)^{-1}\hat{\mathbf{C}}\mathbf{x}_\tau$ can be obtained. This means that $\mathbf{C}$ should match the equation $\hat{\mathbf{C}}^T(\hat{\mathbf{C}}\hat{\mathbf{C}}^T)^{-1}\hat{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}\hat{\mathbf{U}}_{x,r}^T$. Then, by singular value decomposition, $\hat{\mathbf{C}} = \mathbf{U}_c\mathbf{S}_c\mathbf{V}_c^T$. Then, it follows that $\mathbf{U}_{c,r} = \hat{\mathbf{U}}_{x,r}\mathbf{W}$, where $\mathbf{W}$ is an arbitrary $r \times r$ unitary matrix. Thus, $\mathbf{U}_{x,r}$ can be estimated from $\hat{\mathbf{C}}$ up to a unitary matrix ambiguity. Next, the cost in (2.20) is split as follows

$$\mathbf{J}(\mathbf{C}, \{y_{\tau,j}\}_{\tau=0}^{t}) = (t+1)^{-1} \sum_{j=1}^{p} \sum_{\tau=0}^{t} (x_\tau - \mathbf{C}_{:j}^T y_{\tau,j})^2 \tag{2.21}$$

Introducing the consensus constraint $\mathbf{y}_{\tau,1} = \mathbf{y}_{\tau,2} = \ldots = \mathbf{y}_{\tau,p}$, the following minimization formulation can be obtained [29]:

$$\begin{aligned} &\arg\min_{\mathbf{C},\mathbf{y}_\tau} (t+1)^{-1} \sum_{j=1}^{p} \sum_{\tau=0}^{t} (x_\tau - \mathbf{C}_{:j}^T y_{\tau,j})^2, \\ &\text{s. to } \mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,j'}, \quad j' \in \mathcal{N}_j \text{ and } \tau = 0, \ldots, t. \end{aligned} \tag{2.22}$$

In order to solve (2.22), the alternating direction method of multipliers (ADMM) will be employed. Introducing auxiliary variables $\mathbf{z}_{\tau,j}^{j'}$ for $j' \in \mathcal{N}_j$, then the constraints in (2.22) can be equivalently rewritten as

$$\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j}^{j'} \text{ and } \mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j'}^{j} \text{ for } j' \in \mathcal{N}_j \text{ and } j \neq j', \quad j = 1, \ldots, p, \quad \tau = 0, \ldots, t. \tag{2.23}$$

After considering two more Lagrange multipliers $\mathbf{v}_{\tau,j}^{j'}$ and $\mathbf{w}_{\tau,j}^{j'}$, the augmented Lagrangian function can be written as

$$\mathcal{L}[\mathbf{C}, \{\mathbf{y}_{\tau,j}\}_{\tau=0,j=1}^{t,p}, \mathbf{v}, \mathbf{w}] = (t+1)^{-1} \sum_{j=1}^{p} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_{\tau,j})^2$$

$$+\sum_{\tau=0}^{t}\sum_{j=1}^{p}\sum_{b\in\mathcal{N}_j}\left[(\mathbf{v}_{\tau,j}^{j'})^T(\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j}^{j'})+(\mathbf{w}_{\tau,j}^{j'})^T(\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j'}^{j})\right]+0.5c\sum_{\tau=0}^{t}\sum_{j=1}^{p}\sum_{j'\in\mathcal{N}_j}\left[\|\mathbf{y}_{\tau,j}-\mathbf{y}_{\tau,j'}\|_2^2\right], \tag{2.24}$$

Let $\kappa = 0, 1, \ldots,$ denote the index for a coordinate descent cycle, and $k = 1, \ldots, K$ indicate the ADMM iteration index within a coordinate cycle. This means the iteration time for updating $\mathbf{C}_{:j}$ is κ, and in each coordinate descent cycle, we also need to update $\mathbf{y}_{\tau,j}$. The number of ADMM iterations in each cycle for updating $\mathbf{y}_{\tau,j}$ will be K. Thus, the most up-to-date value of $\mathbf{y}_{\tau,j}$ is expressed as $\mathbf{y}_{\tau,j}((\kappa+1)K)$. Next, we apply first-order optimality conditions with respect to $\mathbf{C}_{:j}$ and $\mathbf{y}_{\tau,j}$. First, applying first-order optimality conditions with respect to $\mathbf{C}_{:j}$ is (2.24), we obtain

$$-2\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j)+2\left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K))(\mathbf{y}_{\tau,j}((\kappa+1)K)))^T\right]\mathbf{C}_{:j}(k+1)=0, \tag{2.25}$$

From (2.25), the update for $\mathbf{C}_{:j}$ can be obtained as

$$\mathbf{C}_{:j}(\kappa+1)=\left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)(\mathbf{y}_{\tau,j}((\kappa+1)K))^T\right]^{-1}\times\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j). \tag{2.26}$$

Then, applying first-order optimality with respect to $\mathbf{y}_{\tau,j}$ in (2.24) gives

$$-2(\mathbf{C}_{:j}(\kappa+1))\left[x_\tau(j)-(\mathbf{C}_{:j}(\kappa+1))^T\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)\right]+\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(\kappa)+\mathbf{w}_{\tau,j}^{\kappa+1,j'}(\kappa))$$
$$+0.5c\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[2\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)-2\mathbf{y}_{\tau,j'}^{\kappa+1}(k+1)\right]=0. \tag{2.27}$$

From this, $\mathbf{y}_{\tau,j}$ can be updated as

$$\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = \left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + 2c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \times \left[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \sum_{b\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k)) + c\sum_{j'\in\mathcal{N}_j}(\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{z}_{\tau,j'}^{\kappa+1,j}(k))\right]. \tag{2.28}$$

The two Lagrange multipliers $\mathbf{v}_{\tau,j}^{j'}$ and $\mathbf{w}_{\tau,j}^{j'}$ can be updated by the following recursive updates:

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + c[\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)], \tag{2.29}$$

$$\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k-1) + c[\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)], \tag{2.30}$$

In order to update $\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)$, first-order optimality conditions need to be applied with respect to $\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)$ which gives

$$\begin{aligned}\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1) &= \arg\min_{\mathbf{z}_{\tau,j}^{j'}}\left[\mathcal{L}_a(\mathbf{y}_{j'},\mathbf{v},\mathbf{w},\mathbf{y}_j(k+1))\right] \\ &= \arg\min_{\mathbf{z}_{\tau,j}^{j'}}[\sum_{j=1}^{p}\sum_{\tau=0}^{t}(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_{\tau,j}(k+1))^2 \\ &+ \sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[(\mathbf{v}_{\tau,j}^{\kappa+1,j'})^T(\mathbf{y}_{\tau,j}(k+1) - \mathbf{z}_{\tau,j}^{j'}) + (\mathbf{w}_{\tau,j}^{\kappa+1,j'})^T(\mathbf{y}_{\tau,j}(k+1) - \mathbf{z}_{\tau,j'}^{j})\right] \\ &+ 0.5c\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[\|\mathbf{y}_{\tau,j}(k+1) - \mathbf{z}_{\tau,j}^{j'}\|_2^2 + \|\mathbf{y}_{\tau,j}(k+1) - \mathbf{z}_{\tau,j'}^{j}\|_2^2\right],\end{aligned} \tag{2.31}$$

differentiating with respect to $\mathbf{z}_{\tau,j}^{j'}$, the following equation can be obtained

$$\begin{aligned}&-(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j'}^{\kappa+1,j}(k)) \\ -c(\mathbf{y}_{\tau,j}(k+1) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1) &+ \mathbf{y}_{\tau,j'}(k+1) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)) = 0,\end{aligned} \tag{2.32}$$

then

$$2c\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k) - c(\mathbf{y}_{\tau,j'}(k+1) + \mathbf{y}_{\tau,j}(k+1)) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j'}^{\kappa+1,j}(k). \tag{2.33}$$

Thus, the update for $\mathbf{z}_{\tau,j}^{\kappa+1,j'}$ is given as

$$\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1) = 0.5[\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k+1)] + 0.5c^{-1}[\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k)], \tag{2.34}$$

Equation (2.34) gives the update for $\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1)$. If we initialize $\mathbf{v}_{\tau,j}^{0,j'}(0) = \mathbf{w}_{\tau,j'}^{0,j}(0)$, it will result $\mathbf{v}_{\tau,j}^{\kappa,j'}(k) = \mathbf{w}_{\tau,j'}^{\kappa,j}(k)$, then using (2.34) into (2.32) gives the multiplier update

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + c(\mathbf{y}_{\tau,j}^{\kappa+1}(k) - 0.5\left[\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k+1)\right]), \tag{2.35}$$

from which it follows that $\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k)$ can be updated as:

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + 0.5c(\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{y}_{\tau,j'}^{\kappa+1}(k)). \tag{2.36}$$

Futher, because $\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k) = -\mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)$, (2.34) in (2.28) gives the updating equation

$$\begin{aligned}\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = &\left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + 2c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times \Big[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) - \mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)) \\ &\qquad + c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{\tau,j}^{\kappa+1}(k) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k))\Big].\end{aligned} \tag{2.37}$$

After that, using (2.26),(2.36) and (2.37) to form the distributed PCA algorithm, tabulated as Algorithm 2.

2.4 Sparse distributed principal component analysis

In order to complete the sparse distributed principal component analysis algorithm, the sparse principal component analysis (SPCA) algorithm in Section 2.2 and the dis-

Algorithm 2 Distributed Principal Component Analysis [29]

1: Lagrange multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(-1)\}_{j'\in\mathcal{N}_j}$, sensor local estimates $\mathbf{y}_{\tau,j}^{\kappa+1}(0)$, and variables $\mathbf{y}_{\tau,j'}^{\kappa+1}(0)$ are randomly initialized.

2: Every sensor j gathers $t+1$ measurements $\{x_j(\tau)\}_{\tau=0}^{t}$.

3: **for** $\kappa = 1, 2, \ldots$ **do**

4: Each sensor j updates $\mathbf{C}_{:j}(\kappa+1) \in \mathbb{R}^{r\times 1}$ via (2.26).

5: **for** $k = 1, 2, \ldots, K$ **do**

6: Sensor j updating the multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j'\in\mathcal{N}_j}$ using (2.36).

7: Sensor j receiving the consensus variables from all its neighbors in the subset $\mathcal{N}_j$ and estimates $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ using (2.37).

8: **end for**

9: If $(\|\hat{\mathbf{C}}_{:j}^{\kappa+1} - \hat{\mathbf{C}}_{:j}^{\kappa}\|_2) \leq \epsilon$ then stop (for desired tolerance ϵ).

10: **end for**

tributed principal component analysis (DPCA) algorithm in Section 2.3 will be combined. In this new algorithm, $\mathbf{y}_{\tau,j}$ in SPCA will be updated using the DPCA part, while the formula (2.26) in DPCA responsible for updating $\mathbf{C}_{:j}$ will be replaced by the SPCA part. Thus, the updating recursions for sparse distributed PCA (SDPCA) are given next

$$\mathbf{C}_{:j}(\kappa+1) = [\textstyle\sum_{\tau=0}^{t} 2\mathbf{y}_{\tau,j}\mathbf{y}_{\tau,j}^{T} + \mathbf{cI}]^{-1} \times [(t+1)^{-}1 \textstyle\sum_{\tau=0}^{t} 2\mathbf{y}_{\tau,j}x_{\tau,j} - \mathbf{u} + \mathbf{c}\mathbf{Z}_j(\kappa)], \quad (2.38)$$

whereas the ρ entry of vector $\mathbf{Z}_j(\kappa)$ is valued as:

$$\mathbf{Z}_{\rho,j}(\kappa+1) = sgn(\mathbf{C}_{\rho,j}(\kappa+1) + \frac{\mathbf{u}(\kappa)}{\mathbf{c}}) \times (|\frac{\mathbf{C}_{\rho,j}(\kappa+1) + \mathbf{u}(\kappa)}{c}| - \frac{\lambda}{c})_{+}, \quad (2.39)$$

further the multipliers are updated as:

$$\mathbf{u}(\kappa+1) = \mathbf{u}(\kappa) + c * (\mathbf{C}_{:j}(\kappa+1) - \mathbf{Z}_j(\kappa+1)) \quad (2.40)$$

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + 0.5c(\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{y}_{\tau,j'}^{\kappa+1}(k)), \quad (2.41)$$

while the principal components vector at sensor j is given as

$$\begin{aligned}\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = &\left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + 2c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times \Big[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) - \mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)) \\ &\qquad + c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{\tau,j}^{\kappa+1}(k) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k))\Big], \end{aligned} \tag{2.42}$$

The algorithm summarized in (2.38)-(2.42) is tabulated as Algorithm 3.

Algorithm 3 Sparse Distributed Principal Component Analysis

1: Lagrange multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(-1)\}_{j'\in\mathcal{N}_j}$,sensor local estimates $\mathbf{y}_{\tau,j}^{\kappa+1}(0)$, variables $\mathbf{y}_{\tau,j'}^{\kappa+1}(0)$, $\mathbf{Z}_j$ and $\mathbf{u}$ are randomly initialized.

2: Every sensor j gathers $t+1$ measurements $\{x_j(\tau)\}_{\tau=0}^t$.

3: **for** $\kappa = 1, 2, \ldots$ **do**

4: Each sensor j updates $\mathbf{C}_{:j}(\kappa+1) \in \mathbb{R}^{r\times 1}$ via (2.38).

5: Sensor j updates $\{\mathbf{Z}_{\rho,j}(\kappa+1)\}$ using (2.39).

6: Sensor j updates the multipliers $\mathbf{u}(\kappa+1)$ using (2.40).

7: **for** $k = 1, 2, \ldots, K$ **do**

8: Sensor j updates the multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j'\in\mathcal{N}_j}$ using (2.41).

9: Sensor j receives the consensus variables from all its neighbors and estimates $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ using (2.42).

10: **end for**

11: If $(\|\hat{\mathbf{C}}_{:j}^{\kappa+1} - \hat{\mathbf{C}}_{:j}^{\kappa}\|_2) \leq \epsilon$ then stop (for desired tolerance ϵ).

12: **end for**

2.5 Online algorithm

The above algorithm summarized in (2.38)-(2.42) is a batch algorithm and needs large computational power and storage. In practical applications, sensors constantly ac-

quire new data which needs increasing memory, communication and computational requirements. As time progresses, the algorithm summarized in (2.38)-(2.42) may not be operational. To deal with this problem, an online algorithm is obtained that is computationally efficient and memory efficient. To this end, we introduce the following updating recursions that are adaptive in nature. This means that at every time when t increases, there is no need to calculate everything from 'scratch', we just rely on the most recent updates and newly acquired data to obtain the following updates:

$$\mathbf{C}_{:j}^{t+1}(\kappa+1) = [\sum_{\tau=0}^{t} 2\mathbf{y}_{\tau,j}(K)\mathbf{y}_{\tau,j}^{T}(K)+\mathbf{cI}]^{-1}\times[\sum_{\tau=0}^{t+1} 2\mathbf{y}_{\tau,j}(K)x_{\tau,j}-\mathbf{u}^{t+1}(\kappa)+\mathbf{c}\mathbf{Z}_{j}^{t+1}(\kappa)] \tag{2.43}$$

$$\mathbf{Z}_{\rho,j}^{t+1}(\kappa+1) = sgn(\mathbf{C}_{\rho,j}^{t+1}(\kappa+1) + \frac{\mathbf{u}^{t+1}(\kappa)}{\mathbf{c}}) \times (|\frac{\mathbf{C}_{\rho,j}^{t+1}(\kappa+1) + \mathbf{u}^{t+1}(\kappa)}{c}| - \frac{\lambda}{c})_{+}. \tag{2.44}$$

$$\mathbf{u}^{t+1}(\kappa+1) = \mathbf{u}^{t+1}(\kappa) + c * (\mathbf{C}_{:j}^{t+1}(\kappa+1) - \mathbf{Z}_{:j}^{t+1}(\kappa+1)) \tag{2.45}$$

The principal component vectors are updated at sensor j as:

$$\begin{aligned}\mathbf{y}_{t+1,j}(k+1) = &\left[2\mathbf{C}_{:j}(t+1)(\mathbf{C}_{:j}(t+1))^{T} + 2c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times [2\mathbf{C}_{:j}(t+1)x_{t+1}(j) \\ &- \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{t+1,j}^{j'}(k) - \mathbf{v}_{t+1,j'}^{j}(k) + c\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{t+1,j}(k) + \mathbf{y}_{t+1,j'}(k))],\end{aligned} \tag{2.46}$$

while the Lagrange multipliers are obtained as:

$$\mathbf{v}_{t+1,j}^{j'}(k) = \mathbf{v}_{t+1,j}^{j'}(k-1) + 0.5c[\mathbf{y}_{t+1,j}(k) - \mathbf{y}_{t+1,j'}(k)], k = 1,\ldots,K \;\; and \;\; j' \in \mathcal{N}_j. \tag{2.47}$$

To obtain an online processing algorithm, we set $\mathbf{M}_{x,t}$=$\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}(K)\mathbf{y}_{\tau,j}^{T}(K)$ and $\mathbf{m}_{xy,t}$=$\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}(K)x_{\tau,j}$ used in (2.43). These quantities can be updates as:

$$\mathbf{M}_{x,t}=\mathbf{M}_{x,t-1}+\mathbf{y}_{\tau,j}(K)\mathbf{y}_{\tau,j}^{T}(K), \tag{2.48}$$

$$\mathbf{m}_{xy,t}=\mathbf{M}_{xy,t-1}+\mathbf{y}_{\tau,j}(K)x_{\tau,j} \tag{2.49}$$

Applying (2.48) (2.49) in (2.43), when updating $\mathbf{C}_{:j}^{t+1}(\kappa+1)$, the sensor j does not require storing all data history, and can be updated in an adaptive fashion. The algorithm summarized in (2.43)-(2.47) is tabulated as Algorithm 4.

2.6 Inter-sensor communication noise

In this section, considering the impact of inter-sensor communication noise. Inter-sensor noise affects the information exchanged between sensors. Information is received from sensors $j'\in\mathcal{N}_j$ when sensor j updates variables $\mathbf{v}_{t+1,j}^{j'}(k)$ and $\mathbf{y}_{t+1,j'}(k)$. For example, in (2.47), to update $\mathbf{v}_{t+1,j}^{j'}$, the sensor j needs to acquire variables $\mathbf{y}_{t+1,j'}(k)$ from its neighbors $j'\in\mathcal{N}_j$. Also in (2.46), the variables $\mathbf{v}_{t+1,j'}^{j}(k)$ and $\mathbf{y}_{t+1,j'}(k)$ need to be acquired. Then, when sensor j acquires variables $\mathbf{v}_{t+1,j'}^{j}(k)$ and $\mathbf{y}_{t+1,j'}(k)$, in the presence of communication noise, it will receive $\mathbf{v}_{t+1,j'}^{j}(k)+{\boldsymbol{\zeta}_j}^{t+1,j'}(k)$ and $\mathbf{y}_{t+1,j'}(k)+{\boldsymbol{\eta}_j}^{t+1,j'}(k)$, where ${\boldsymbol{\zeta}_j}^{t+1,j'}(k)$ and ${\boldsymbol{\eta}_j}^{t+1,j'}(k)$ correspond to zero-mean communication noise contaminating the link from sensor j' to j. Then (2.46) and (2.47) can be written as

$$\begin{aligned}\mathbf{y}_{t+1,j}(k+1)=&\left[2\mathbf{C}_{:j}(t+1)(\mathbf{C}_{:j}(t+1))^{T}+2c|\mathcal{N}_j|\mathbf{I}\right]^{-1}\\&\times[2\mathbf{C}_{:j}(t+1)x_{t+1}(j)-\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{t+1,j}^{j'}(k)\\&-(\mathbf{v}_{t+1,j'}^{j}(k)+{\boldsymbol{\zeta}_j}^{t+1,j'}(k))+c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{t+1,j}(k)+\mathbf{y}_{t+1,j'}(k)+{\boldsymbol{\eta}_j}^{t+1,j'}(k))],\end{aligned} \tag{2.50}$$

Algorithm 4 Adaptive Sparse Distributed Principal Component Analysis

1: Lagrange multipliers $\{\mathbf{v}_{0,j}^{j'}(-1)\}_{j'\in\mathcal{N}_j}$,sensor local estimates $\mathbf{y}_{0,j}^{\kappa+1}(0)$, variables $\mathbf{y}_{0,j'}^{\kappa+1}(0)$, $\mathbf{Z}_j^0(0)$ and $\mathbf{u}^0(0)$ are randomly initialized.

2: Every sensor j gathers $t+1$ measurements $\{x_j(\tau)\}_{\tau=0}^t$.

3: **for** $t = 0, 1, 2, \ldots$ **do**

4: **for** $\kappa = 1, 2, \ldots, K$ **do**

5: Each sensor j updates $\mathbf{C}_{:j}^{t+1}(\kappa+1) \in \mathbb{R}^{r\times 1}$ via (2.38).

6: Sensor j updates $\{\mathbf{Z}_{\rho,j}^{t+1}(\kappa+1)\}$ using (2.39).

7: Sensor j updates the multipliers $\mathbf{u}^{t+1}(\kappa+1)$ using (2.40).

8: **end for**

9: Initialize $\mathbf{y}_{t+1,j}(0) = \mathbf{C}_{:j}(t+1)\mathbf{x}_\tau(j)$ and $\mathbf{v}_{t+1}(0) = \mathbf{v}_t(K)$

10: **for** $k = 1, 2, \ldots, K$ **do**

11: Sensor j updates the multipliers $\{\mathbf{v}_{t+1,j}^{j'}(k)\}_{j'\in\mathcal{N}_j}$ using (2.41).

12: Sensor j receives the consensus variables from all its neighbors and estimates $\mathbf{y}_{t+1,j}(k+1)$ using (2.42).

13: **end for**

14: If $(\|\hat{\mathbf{C}}_{:j}^t - \hat{\mathbf{C}}_{:j}^t\|_2) \leq \epsilon$ then stop (for desired tolerance ϵ).

15: **end for**

$$\mathbf{v}_{t+1,j}^{j'}(k) = \mathbf{v}_{t+1,j}^{j'}(k-1) + 0.5c[\mathbf{y}_{t+1,j}(k) - (\mathbf{y}_{t+1,j'}(k) + \boldsymbol{\eta}_j{}^{t+1,j'}(k))], k = 1, \ldots, K \tag{2.51}$$

The effect of communication noise will be tested on the performance of the algorithm. It will be seen in Chapter 3 that the proposed framework is robust in the presence of noise.

CHAPTER 3

NUMERICAL TESTS AND DISCUSSION

In this chapter, the performance of the sparse distributed principal component analysis algorithm developed in Chapter 2 will be tested. The online algorithm will be used to estimate different numbers of principal components, under different numbers of sensors to demonstrate that the novel algorithms are able to work for different network sizes. Different numbers of training data will be used to show the accuracy rate. Futher, inter-sensor noise will be applied to demonstrate the robustness of the proposed technique different from existing alternatives. Finally, real data corresponding to CO2 levels measured in an area will be used to advocate the flexibility and accuracy of the proposed scheme.

3.1 Subspace projection estimation error for different network settings

We consider the number of sensors in the sensor network as p=20,30 or 50. These sensors will be randomly distributed in the area $[0 \times 1] \times [0 \times 1]$. The communication range distance for each sensor is $d = 0.25$ units. Each sensor collects $t = 1000$ observations. The number of ADMM iterations is set at $K = 20$. Other parameters are set at $c = 1$ and $\lambda = 0.1$ (for $r = 1$ eigenvector), $\lambda = [0.1, 0.01]$ (for $r = 2$ eigenvectors), $\lambda = [0.1, 0.09, 0.01]$ (for $r = 3$ eigenvectors). Here, monitoring how the subspace projection estimation error $e(t) := \|\mathbf{C}^T(t)(\mathbf{C}(t)\mathbf{C}^T(t))^{-1}\mathbf{C}(t) - \mathbf{U}_{x,r}\mathbf{U}_{x,r}^T\|_F^2$,where $\|\cdot\|_F$ is the Frobenius norm, behaves for an increasing number of data. In order to get an average error value, we will run the algorithm for 100 Monte Carlo runs.

Figure 3.1, 3.2, and 3.3 compare the performace of SDPCA with respect to the error $e(t)$ for a different number of principal components $r = 1, 2, 3$ and different numbers of

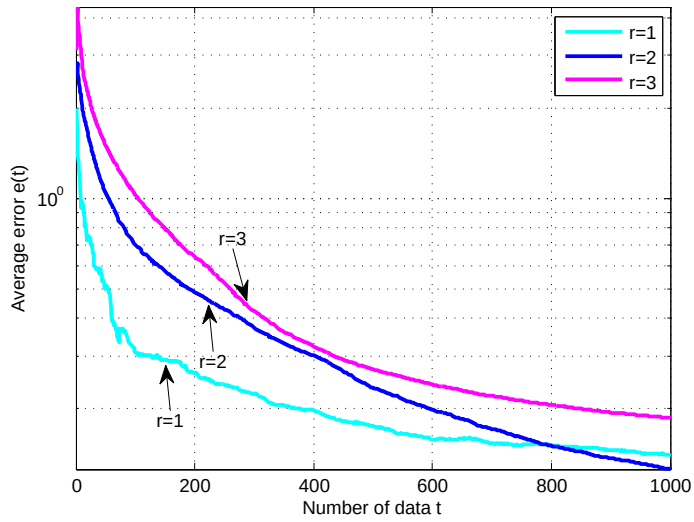

Figure 3.1. Subspace projection estimation error $e(t)$ vs. time index t for $r = 1, 2, 3$, where the number of sensors is $p = 20$.

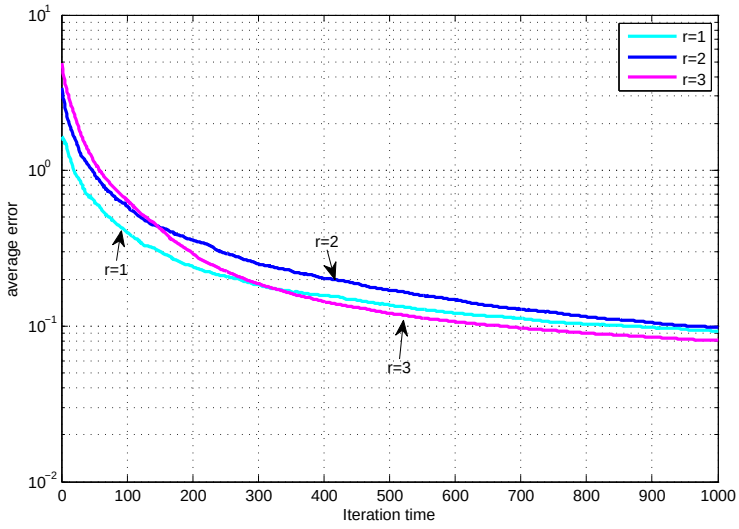

Figure 3.2. Subspace projection estimation error $e(t)$ vs. time index t for $r = 1, 2, 3$, where the number of sensors is $p = 30$.

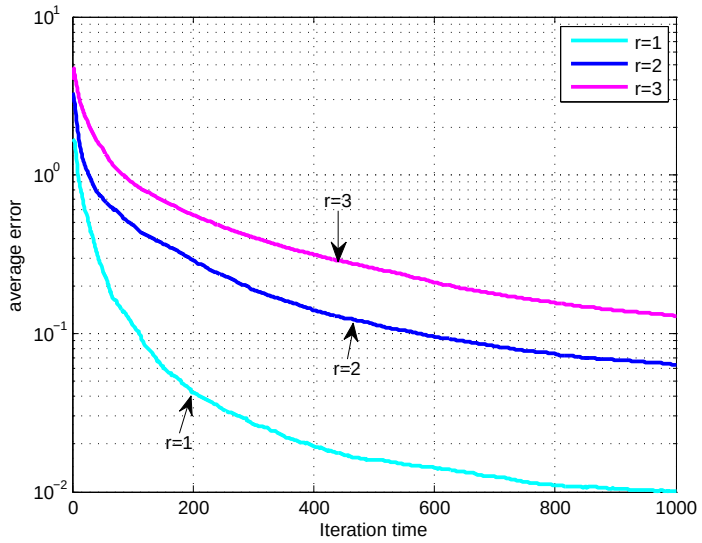

Figure 3.3. Subspace projection estimation error $e(t)$ vs. time index t for $r = 1, 2, 3$, where the number of sensors is $p = 50$.

sensors $p = 20, 30, 50$. It can be seen that in different situations, the error $e(t)$ will decrease and become steady when sensors obtain more and more observations. It also shows that for different principal components r, $e(t)$ is not affected by the number of principal components r. However, when the number of sensors increases to $p = 50$, estimating $r = 1$ eigenvector is easier than estimating $r = 3$ eigenvectors.

3.2 Subspace projection estimation error for a different number of ADMM iterations

Here, the number of sensors is set as $p = 50$. Then, we test the error achieved by SDPCA $e(t)$ for different number of ADMM iterations, namely $K = 5, 15, 20$. The error $e(t)$ is plotted versus the number of measurements t. Other parameters will be set as $p = 50$, $c = 1$, $\lambda = 0.1$. 100 Monte Carlo iterations are executed to obtain the average value of $e(t)$. Figure 3.4 shows that as K increases, the estimation performance of SDPCA improves, and

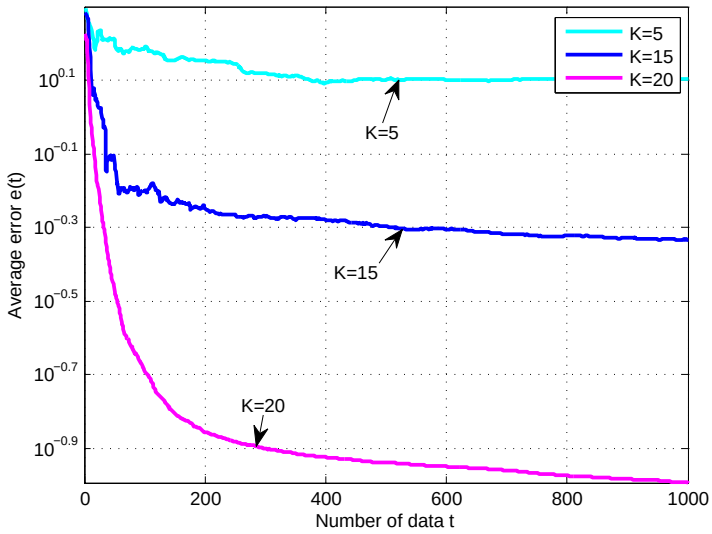

Figure 3.4. Subspace projection estimation error $e(t)$ vs. time index t for $r = 1$ and $p = 50$, while the ADMM number of iterations is set as $K = 5, 15, 20$.

decreases as t increases.

3.3 Probability of correctly recovering zero entries in principal subspace

In this test, the zero entries in estimated $\mathbf{C}$ and true subspace $\mathbf{U}_{x,r}$ will be compared. If $\mathbf{C}$ is correct, then $\mathbf{C}$ will have zero entries in the same place as $\mathbf{U}_{x,r}$. The entries in $\mathbf{C}$ and $\mathbf{U}_{x,r}$ may not be exactly zero, thus we use thresholding. When the elements are smaller than a desired tolerance, then they will be set to zero. Also, other parameters will be set as $p = 50$, $c = 1$ and $\lambda = 0.1$. Figure 3.5 indicates that as more observations are acquired the probability of correctly recovering zero entries increases.

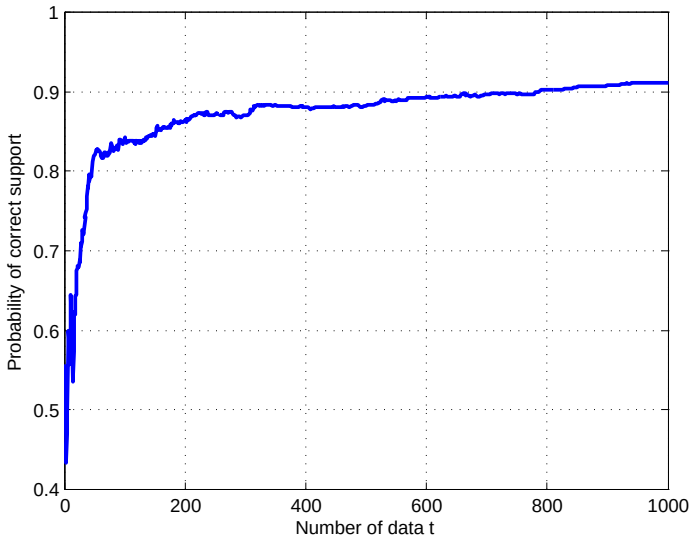

Figure 3.5. Probability vs. time index when estimating $r = 1$ principal eigenvector.

3.4 Inter-sensor communication noise

Here, the inter-sensor noise is considered for different signal-to-noise ration (SNR) values and setting the number of sensors to $p = 50$, parameters $c = 1$ and $\lambda = 0.1$ (for $r = 1$ eigenvector),$\lambda = [0.1, 0.01]$ (for $r = 2$ eigenvectors), $\lambda = [0.1, 0.09, 0.01]$ (for $r = 3$ eigenvectors).

Figures 3.6,3.7 and 3.8 obviously show that lower noise during communication will allow SDPCA to get a better performance. It can be seen that there may exist a tolerance for SNR. If the SNR is bigger than the tolerance, the inter-sensor noise will not affect the performance too much. This is to be contrasted with existing PCA approaches that in the presence of communication noise always diverge.

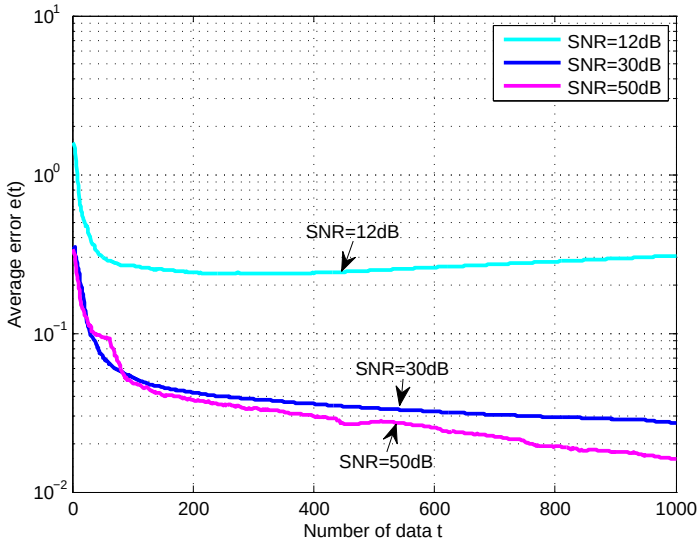

Figure 3.6. Subspace projection estimation error $e(t)$ vs. time index t for $r = 1$ in the presence of inter-sensor noise and SNR values 12dB, 30dB and 50dB.

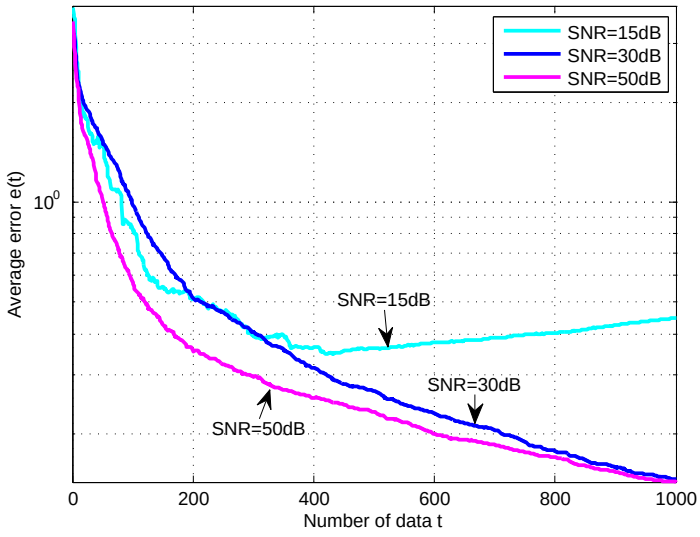

Figure 3.7. Subspace projection estimation error $e(t)$ vs. time index t for $r = 2$ in the presence of inter-sensor noise and SNR values 15dB, 30dB and 50dB.

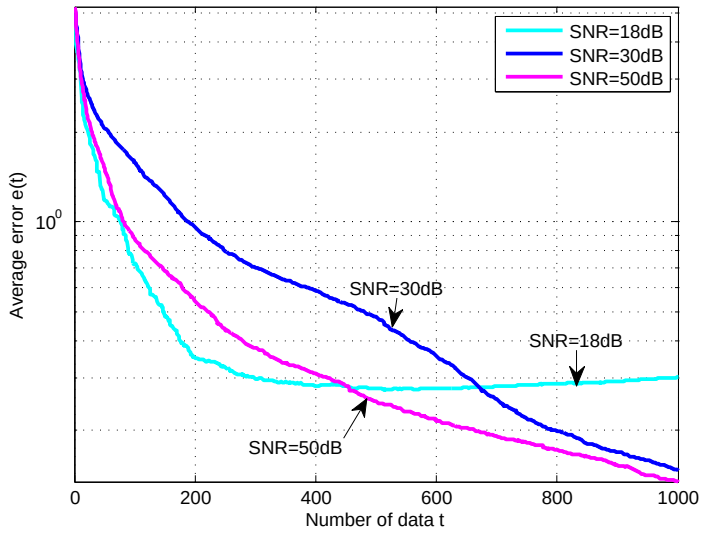

Figure 3.8. Subspace projection estimation error $e(t)$ vs. time index t for $r = 3$ in the presence of inter-sensor noise and SNR values 18dB, 30dB and 50dB.

3.5 Real data

A set of data showing the CO_2 levels from The Berkeley Atmospheric CO_2 Observation Network [30]. The set of data is coming from 28 sites and each one will have 1728 observations. The data are normalized to zero-mean and unit corvariance before processing. The parameters will be set at $c = 1$ and $\lambda = 0.1$ (for $r = 1$ eigenvector),λ=[0.1,0.01] (for $r = 2$ eigenvectors), λ=[0.1,0.09,0.01] (for $r = 3$ eigenvectors)

Similar to the case of using synthetic data, DSPCA works efficiently even for real data with the performance improving as the number of data increases.

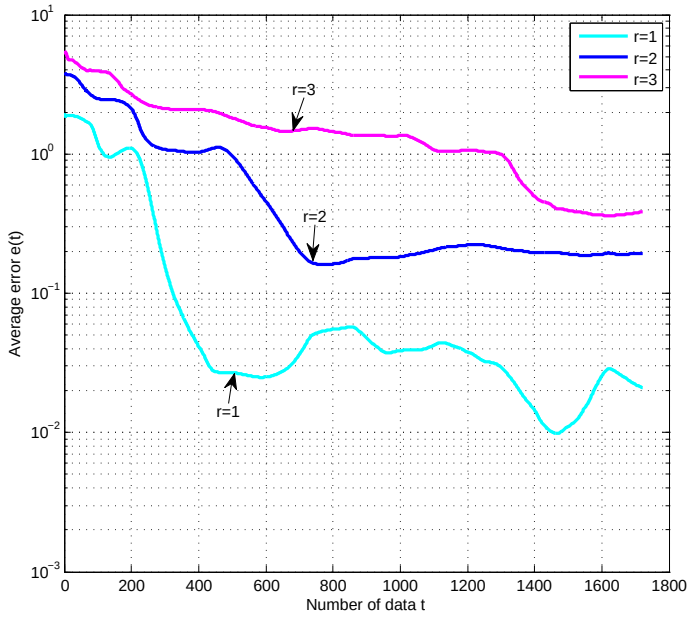

Figure 3.9. Subspace projection estimation error $e(t)$ vs. time index t when estimating $r = 1, 2, 3$ principal components with real data .

3.6 Sparsity versus no sparsity exploitation

The value of λ is a parameter which should be carefully selected. Here, we compare with $\lambda = 0$ and $\lambda = 0.1$ in the real data used earlier. Other parameters are $c = 1$, $r = 1$ and $K = 20$.

Figure 3.10 depicts that the proposed SDPCA performs much better than standard PCA which does not exploit sparsity ($\lambda = 0$). Thus, the proposed framework has the potential to use the zeros present in the principal eigenspace and achieve better performance than standard PCA ($\lambda = 0$, no sparsity).

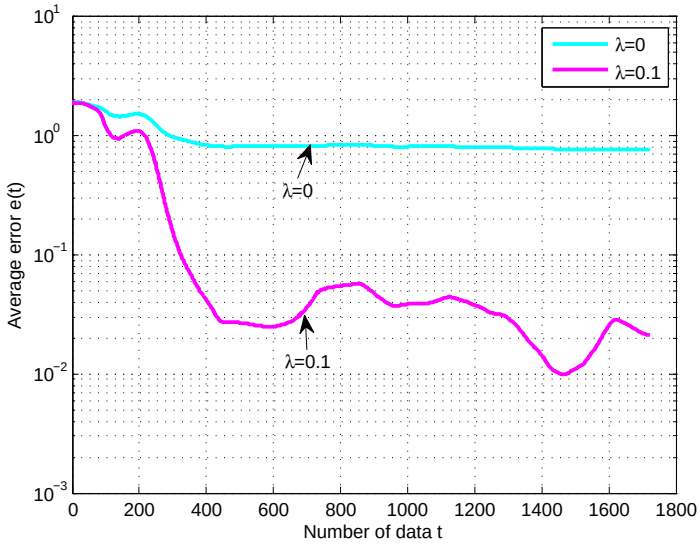

Figure 3.10. Subspace projection estimation error $e(t)$ vs. time index t when estimating $r = 1$ eigenvector for $\lambda = 0$ (no sparsity) and $\lambda = 0.1$.

CHAPTER 4

CONCLUSIONS AND FUTURE DIRECTIONS

This thesis focused on developing a sparse distributed principal components analysis algorithm (SDPCA). The main goal is to improve upon existing compression technologies relying on dimensionality reduction. The design of the algorithm was divided into two parts, one is a sparsity utilization component and the other is a distributed mechanism. Then, we combined these two parts together to obtain the SDPCA algorithm. Norm-one regulation along with the alternating direction method of multipliers were employed. Simple local updating recursions were obtained that are robust in the presence of noise. Different numerical tests were applied in different situations, like different sizes of sensor networks, different ADMM iterations and different noise SNR settings. Both synthetic data and real data were applied in the tests. These tests demonstrated that the novel algorithm can be effectively applied in different situations and gives good principal subspace estimation results.

Future directions involve the following research tasks:

1. Develop adaptive algorithms for time-varying covariance matrices. So far, the algorithm designed is relying on a stationary covariance matrix. But sometimes the covariance matrix is dynamic, thus it is necessary to find a way to track the changing eigenvectors and perform adaptive compression.

2. Denoising. In this algorithm, we only considered the inter-sensor noise, but the sensing noise may also have effects in data processing. A low sensing SNR may result in poor estimation performance. Denoising techniques relying on sparse distributed PCA will be devised.

3. Heterogeneous sensor networks. Oftentimes sensing systems acquire more than one types of measurements. These different type of data may be correlated with each other. Techniques to use and process heterogeneous data, while performing compression can be considered in future research.

REFERENCES

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] S. Prasanna and S. Rao, "An overview of wireless sensor networks applications and security," *International Journal of Soft Computing and Engineering (IJSCE), ISSN*, pp. 2231–2307, 2012.

[3] I. Stojmenovic, *Handbook of sensor networks: Algorithms and architectures*. John Wiley & Sons, 2005, vol. 49.

[4] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1567–1576.

[5] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*, vol. 1. IEEE, 2006, pp. 286–295.

[6] S. Sharma, D. Kumar, and K. Kishore, "Wireless sensor networks-a review on topologies and node architecture," *International Journal of Computer Sciences and Engineering*, vol. 1, no. 2, pp. 19–25, 2013.

[7] M. M. Zanjireh and H. Larijani, "A survey on centralised and distributed clustering routing algorithms for wsns," in *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*. IEEE, 2015, pp. 1–6.

[8] M. Haenggi, "Distributed sensor networks: a cellular nonlinear network perspective," *International journal of neural systems*, vol. 13, no. 06, pp. 405–414, 2003.

[9] A. Bharathidasan and V. A. S. Ponduru, "Sensor networks: An overview," in *IEEE INFOCOM*, vol. 4, 2002.

[10] Y.-C. Wang, "Data compression techniques in wireless sensor networks," *Pervasive Computing, New York: Nova Science Publishers, Inc*, 2012.

[11] D. D. Suhr, "Principal component analysis vs. exploratory factor analysis," *SUGI 30 proceedings*, vol. 203, p. 230, 2005.

[12] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman, "An overview of principal component analysis," *Journal of Signal and Information Processing*, vol. 4, no. 3B, p. 173, 2013.

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

[14] 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.

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

[16] Z. jian Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *In Proceedings of International workshop on Advanced Parallel Processing Technologies (APPT*, 2005.

[17] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson, "Distributed clustering using collective principal component analysis," *Knowledge and Information Systems*, vol. 3, p. 2001, 1999.

[18] H. Qi, T. wei Wang, and J. D. Birdwell, "Global principal component analysis for dimensionality reduction in distributed data mining," 2004.

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

[20] M. Gastpar, P. L. Dragotti, and M. Vetterli, "The distributed karhunen–loeve transform," *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5177–5196, 2006.

[21] 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.

[22] 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.

[23] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Internal Report KU Leuven ESAT-SCD*, 2013.

[24] I. D. Schizas and A. Aduroja, "A distributed framework for dimensionality reduction and denoising," *Signal Processing, IEEE Transactions on*, vol. 63, no. 23, pp. 6379–6394, 2015.

[25] Q. Zhao, D. Meng, and Z. Xu, "Robust sparse principal component analysis," *Science China Information Sciences*, vol. 57, no. 9, pp. 1–14, 2014.

[26] I. D. Schizas and G. B. Giannakis, "Covariance eigenvector sparsity for compression and denoising," *Signal Processing, IEEE Transactions on*, vol. 60, no. 5, pp. 2408–2421, 2012.

[27] O. Yilmaz and A. N. Akansu, "Quantization of eigen subspace for sparse representation," *IEEE Transactions on Signal Processing*, vol. 63, no. 14, pp. 3576–3585, 2015.

[28] 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.

[29] A. Aduroja, I. D. Schizas, and V. Maroulas, "Distributed principal components analysis in sensor networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5850–5854.

[30] http://beacon.berkeley.edu/Sites.aspx.