Asynchronous gossip principal components analysis

Jerome Fellus, David Picard, Philippe-Henri Gosselin

Asynchronous Gossip Principal Components Analysis

Jerome FELLUS[a,*], David PICARD[a], Philippe-Henri GOSSELIN[a],

[a]*ETIS - UMR CNRS 8051 - ENSEA - Universite de Cergy-Pontoise*
6 Avenue du Ponceau 95014 Cergy, France

Abstract

This paper deals with Principal Components Analysis (PCA) of data spread over a network where central coordination and synchronous communication between networking nodes are forbidden. We propose an asynchronous and decentralized PCA algorithm dedicated to large scale problems, where "large" simultaneously applies to dimensionality, number of observations and network size. It is based on the integration of a dimension reduction step into a Gossip consensus protocol. Unlike other approaches, a straightforward dual formulation makes it suitable when observed dimensions are distributed. We theoretically show its equivalence with a centralized PCA under a low-rank assumption on training data. An experimental analysis reveals that it achieves a good accuracy with a reasonable communication cost even when the low-rank assumption is relaxed.

Keywords: Distributed Machine Learning, Dimensionality reduction, Gossip protocols

1. Introduction

Dimensionality reduction plays an important role in solving large scale machine learning problems where input data usually consists of a huge number of observations in a high-dimensional space. Classification, regression, or similarity ranking of such raw data often raise computation and storage issues. In practice, the intrinsic dimensionality of observed phenomena is much lower than the extrinsic dimension of the input space. Dimensionality reduction then aims at pro-

*Corresponding author
Email addresses: jerome.fellus@ensea.fr (Jerome FELLUS), picard@ensea.fr (David PICARD), gosselin@ensea.fr (Philippe-Henri GOSSELIN)

jecting input data into a lower-dimensional space such that subsequent learning stages keep a maximal accuracy.

Principal Components Analysis (PCA) is a linear approach to dimensionality reduction [1, 2]. Given a sample matrix $\mathbf{X} \in \mathbb{R}^{D \times n}$ made of n observations in $\mathbb{R}^D$, PCA finds an orthonormal basis $\mathbf{U}^\star = [\mathbf{u}_1 \dots \mathbf{u}_q]$, $\mathbf{u}_k \in \mathbb{R}^D$ that projects the input sample $\mathbf{X}$ into the q-dimensional subspace, $q < D$, that retains the maximal variance in $\mathbf{X}$. Equivalently, the PCA solution is the linear projection that best conserves the Gram matrix (*i.e.*, the matrix of pairwise inner-products):

$$\mathbf{U}^\star = \underset{\mathbf{U} \in \mathbb{R}^{D \times q}}{\arg\min} \left\| \mathbf{X}^\top \mathbf{X} - \mathbf{X}^\top \mathbf{U} \mathbf{U}^\top \mathbf{X} \right\|_F^2 \quad \text{s.t.} \quad \mathbf{U}^{\star\top} = \mathbf{U}^{\star -1} \tag{1}$$

The optimal conservation of the inner product makes PCA particularly suited to feed algorithms that solely rely on the inner product instead of the input data [3] (*e.g.*, Support Vector Machines). PCA enjoys a closed-form solution, as $\mathbf{U}^\star$ is made of the q leading eigenvectors of the sample covariance matrix [2]:

$$\mathbf{C} = \frac{1}{n} \mathbf{X} \mathbf{X}^\top - \mu \mu^\top \quad \text{where} \quad \mu = \frac{1}{n} \mathbf{X} \mathbf{1} \quad \text{is the sample mean} \tag{2}$$

Like most statistical learning tools, PCA was formulated for centralized setups where all data are available at a single location. This assumes that the solution can be computed by a single machine and that all intermediary results fit in the main memory. However, this assumption is unrealistic in most applicative fields that deal with very large sample matrices. For instance, in biomedical, multimedia, or remote sensing applications, D and n often grows up to millions. The sample and covariance matrices then scale in Terabytes. Moreover, the $O(D^3)$ complexity of covariance eigendecomposition translates into an exa-flop computation cost. Besides, along with the democratization of connected devices, data tends to originate from an increasing number of distributed sources with reasonable computing capacity, immersed in large unreliable networks without any central coordination. This has led to a number of so-called Distributed PCA algorithms, designed to deal with the spread of input data over multiple networking nodes.

Because computing μ and $\mathbf{C}$ would involve the full data $\mathbf{X}$ which is unknown to individual nodes, distributed PCA requires dedicated algorithms combining node-local optimization and inter-node communications. Distributed PCA encompasses two main scenarios, depending of the way the entries of $\mathbf{X}$ are spread over the networking nodes. Consider a network made of N (strongly-connected) nodes. In a Distributed Samples (DS) scenario, each node i holds a distinct sample $\mathbf{X}_i \in \mathbb{R}^{D \times n_i}$ of n_i observations (*i.e.*, the *columns* of $\mathbf{X}$ are distributed). Con-

versely, in a Distributed Coordinates (DC) scenario, each node holds all n observations, but only gets a subset $\mathbf{X}_i \in \mathbb{R}^{D_i \times n}$ of their components (*i.e.*, the *rows* of $\mathbf{X}$ are distributed). On average, each $\mathbf{X}_i$ is then N times smaller than $\mathbf{X}$, thus $n_i \ll D$ in DS case while $D_i \ll n$ in DC case. No assumption is made on their exact sizes, as they may be very different at each node. In both DS and DC scenarios, a typical objective is to provide all nodes with compressed representations of their locally-hosted observations that account for the contribution of all components. Nodes then have to find a consensus basis $\mathbf{U}^\star$ that minimizes the PCA objective defined in Equation (1) over the complete data. Usually, one also wants an operator that allows projection of future observations into the same output space, but not all approaches are able to provide such operator at a low cost. In spite of similar goals, the DS and DC scenarios have been tackled separately with very different approaches in the distributed PCA litterature [4].

In this work, we consider the asynchronous decentralized PCA problem, which specializes distributed PCA by adding the following constraints:

(C1) No sample exchange - Samples cannot be exchanged between nodes, for size, privacy or property reasons.

(C2) Asynchrony - Nodes must never wait for each other.

(C3) Decentralization - All nodes and links must play the same role. Formally, nodes and links must be selected for communication with the same probability and all nodes must run the same procedures.

(C4) Consensus - All nodes must obtain the same orthogonal basis. Node-local solutions must allow projection of future observations.

Satisfying these four constraints, a distributed PCA algorithm gains applicability to a wider range of networking situations such as sensor networks, Internet-enabled multimedia devices, cloud computing systems, etc, where central coordination or synchronous functioning can be inapplicable.

In this paper, we propose a decentralized and asynchronous algorithm called Asynchronous Gossip Principal Component Analysis (AGPCA) that satisfy all the above constraints. Our algorithm is a revision and extension of previous work presented in [5]. The original contributions of this paper are the following:

- We give a formal and in-depth description of AGPCA in the DS case, as well as the intuitions leading to the algorithm.

- We propose an extension to the DC case through a dual transcription.
- Provided a low rank property is met on the data, we give a theoretical guarantee that AGPCA yields the exact solution of the centralized PCA.
- We present experiments for both DS and DC scenarios, as well as results for various network topologies.

The remaining of this paper is organized as follows: In the next Section, we detail related works on distributed PCA algorithms. Then, we present our AGPCA algorithm for the DS case in Section 3. We present the extension to the DC case in Section 4. In section 5, we theoretically show that the output of AGPCA is identical to a centralized PCA under a low-rank assumption on the data. The last section gathers experimental results both in DS and DC case, before we conclude.

2. Related Work

In this section, we present existing algorithms for distributed PCA. We first present methods that integrate prior information on the input data. Then, we compare existing algorithms in terms of decentralization and asynchrony. Finally, we discuss the benefits of approaches based on model aggregation over those based on iterative optimization passes over the data.

2.1. Prior knowledge about input data

Existing distributed PCA approaches can be first distinguished by their level of prior knowledge about the input data matrix $\mathbf{X}$. Indeed, $\mathbf{X}$ can either carry node-local observations independently of their network relationships or integrate properties of the network graph itself. In the latter case, a typical object of interest is the adjacency matrix of the weighted network graph, whose entries correspond to some scalar relationship *between* nodes. For instance, when these entries represent estimated geographic distances between neighbouring sensors, their absolute geographic position can be recovered by computing the three principal components through distributed PCA [6].

Other methods have considered the case where the data distribution inherits specific characteristics from the network structure, such as statistical dependencies. This happens when, *e.g.*, data is generated by flowing through directed paths along the network structure, thus making data at downstream nodes dependent of data at upstream nodes, but independent from each other. Properly modeling these statistical dependencies through Graphical Models, either undirected (*e.g.*,

Decomposable Gaussian Graphical Models [7]) or directed (*e.g.*, Directed Gaussian Acyclic Graphical Models [8]), one can benefit from the natural sparsity of the concentration matrix (*i.e.*, the inverse covariance) or its Cholesky factor to estimate the principal subspace with reduced communication costs.

On the contrary, in this work the network topology has no relevance in the desired result, as information is solely carried by the nodes and not by the links. Still, link-related data can be seen as observations relative to one or both of their ends, making methods aimed at node-related data suitable for link-related data.

2.2. Decentralization and asynchrony

Another classification criterion separates decentralized approaches from those which assign node-specific roles and asynchronous approaches from those based on synchronous communications.

In [9], a parallel PCA algorithm is proposed. Sufficient statistics $\mathbf{X}_i\mathbf{1}$ and $\mathbf{X}_i\mathbf{X}_i^\top$ are locally computed at all nodes and transmitted to a master node that performs a global Singular Value Decomposition (SVD) to obtain the PCA result. This approach is only suitable when the master node can handle the $O(D^3)$ complexity of the SVD and assumes that $D \times D$ covariance matrices can be exchanged on the network, which is unrealistic in many large scale contexts.

In [4], a distributed PCA algorithm for the DC scenario is proposed. Exchanging only $q \times q$ matrices, nodes iteratively maximize the variance retained by the reduced basis. Even though the process is decentralized, nodes have to update their estimates synchronously before performing any further computation, thus violating (**C2**). The whole system performance is then limited by the slowest networking node. Moreover, synchronous updating is hard to sustain in large networks and can result in overwhelming idle phases even when nodes have identical computing resources.

A fully asynchronous and decentralized Power Iteration method is proposed in [10] using random matrix sparsifications and a nested Sum-Weight Gossip averaging protocol to reduce communication costs. However, its original formulation only provides the first principal component. Synchronous passes would be required to sequentially obtain the next ones.

2.3. Multiple passes over the data vs. model aggregation

An important feature of some methods is to require only one pass over node-local data. This is the case of [9, 11], contrarily to [6, 10] which require multiple access to the $\mathbf{X}_i$ because of their iterative optimization scheme. Temporary access

to data is frequently considered in online approaches like [4] to handle stream data under short-time ergodicity and stationarity assumptions.

In [11], the authors propose an operator to aggregate Mixtures of Probabilistic PCA models (MPPCA, [12]) in a maximum-likelihood sense, without resorting to the original data used to train the models. Multiple models can thus be trained in a first phase, and aggregated in a second phase. Used in conjunction with the Gossip optimization framework proposed in [13], one can easily obtain a decentralized and asynchronous PCA algorithm. However, as developped in [14], model aggregation give best results when models are further selected through neighbors cross-validation, which seems non-trivial to achieve in an asynchronous fashion.

In [15], the authors first compute sufficient local statistics and aggregate them by means of a distributed consensus algorithm before locally computing PCA on the aggregated statistics. Aggregation of the local statistics requires exchanging $D \times D$ matrix estimates, which can be prohibitive for high-dimensional input spaces. Our approach is akin to [15], but solves this problem by computing PCA *before* the distributed aggregation.

3. Asynchronous Gossip PCA for Distributed Samples scenario

In this section, we introduce our proposed AGPCA algorithm, which is suitable for both DS and DC scenarios. This section details AGPCA for the DS scenario. We present the extension of AGPCA to the DC scenario in the next section.

In a DS scenario, each node hosts a local sample $\mathbf{X}_i \in \mathbb{R}^{D \times n_i}$ made of n_i observations $[\mathbf{x}_1, \ldots, \mathbf{x}_{n_i}]$ in $\mathbb{R}^D$. In such scenario, AGPCA provides all nodes with the same orthogonal basis $\mathbf{U}^\star \in \mathbb{R}^{D \times q}$ that spans the q-principal subspace of the full covariance matrix $\mathbf{C}$ as defined by Equation (2). Each node i can then project its local data $\mathbf{X}_i$ to obtain $\mathbf{Y}_i = \mathbf{U}^{\star\top}\mathbf{X}_i$, where $\mathbf{Y}_i \in \mathbb{R}^{q \times n_i}$ is the compressed representation of $\mathbf{X}_i$. Importantly, any future observation $\mathbf{x}_{new}$ can be compressed in the same way: $\mathbf{y}_{new} = \mathbf{U}^{\star\top}\mathbf{x}_{new}$, even if $\mathbf{X}_i$ is deleted or was only accessible during training (*e.g.*, streaming data case).

The intuition behind AGPCA is built upon the following 4 main principles:

1. Distributed PCA can be formulated as a distributed averaging of covariance matrices followed by local eigendecompositions.
2. Asynchronous and decentralized averaging of covariance matrices is possible through Sum-Weight Gossip consensus protocols [16].
3. By properly defining uniform scaling and sum operators for orthogonal bases, we can extend the Sum-Weight Gossip protocol to exchange only the

Algorithm 1 AGPCA-DS Emission procedure

$\mathbf{a}_i \leftarrow \mathbf{X}_i\mathbf{1}$; $\mathbf{G}_i \leftarrow \mathbf{X}_i^\top\mathbf{X}_i$; $w_i \leftarrow n_i$
$(\mathbf{V}_i, \mathbf{L}_i) \leftarrow$ *eigendecompose*$(\mathbf{G}_i)$
$\mathbf{U}_i \leftarrow \mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}}$
loop
  $j \leftarrow$ *randomNeighbor*(i)
  $(\mathbf{a}_i, \mathbf{L}_i, w_i) \leftarrow \frac{1}{2}(\mathbf{a}_i, \mathbf{L}_i, w_i)$
  Send $(\mathbf{a}_i, \mathbf{U}_i, \mathbf{L}_i, w_i)$ to j
end loop

Algorithm 2 AGPCA-DS Reception procedure

loop
  Upon receipt of $(\mathbf{a}_j, \mathbf{U}_j, \mathbf{L}_j, w_j)$
  $\mathbf{a}_i \leftarrow \mathbf{a}_i + \mathbf{a}_j$; $w_i \leftarrow w_i + w_j$
  $\mathbf{Q}_0 \leftarrow \mathbf{U}_i$
  for $t \in [0, max\_t[$ **do**
    $(\mathbf{Q}_{t+1}, \mathbf{R}_{t+1}) \leftarrow$ *QR*$(\mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top\mathbf{Q}_t + \mathbf{U}_j\mathbf{L}_j\mathbf{U}_j^\top\mathbf{Q}_t)$
  end for
  $\mathbf{U}_i \leftarrow \mathbf{Q}_{max\_t}$; $\mathbf{L}_i \leftarrow$ *diag*$(\mathbf{R}_{max\_t})$
end loop

q leading eigenvectors of the local covariance matrices, and consequently achieve a better network efficiency.

4. Using the duality between the eigendecomposition of the covariance and Gram matrices through Singular Value Decomposition (SVD), we can compute the local Gram matrices ($n_i \times n_i$) instead of the local covariance matrices ($D \times D$) and consequently reduce the memory usage at each node.

In the remaining of this section, we detail each of these points leading to AGPCA procedures for the DS scenario to be concurrently run at each node as presented in Algorithm 1 and 2[1].

[1]When implementing AGPCA, care should be taken that iterations of Algorithm 1 and Algorithm 2 be mutually exclusive, to ensure integrity of the concurrently updated variables.

3.1. Distributed PCA as a distributed averaging problem

From Equation (1), the sample correlation matrix $\mathbf{X}\mathbf{X}^\top$ and the sample sum $\mathbf{X}\mathbf{1}$ are sufficient statistics to compute the PCA solution. As observed in [15, 9], these statistics can be obtained by computing partial statistics $\mathbf{X}_i\mathbf{1}$ and $\mathbf{X}_i\mathbf{X}_i^\top$ at each node i and summing over i. As a result, the full sample mean and covariance matrix can be computed as a distributed average of locally-computed means $\mu_i = \frac{1}{n_i}\mathbf{X}_i\mathbf{1}$ and correlation matrices $\mathbf{B}_i = \mathbf{X}_i\mathbf{X}_i^\top$:

$$\mu = \frac{1}{n}\mathbf{X}\mathbf{1} = \frac{1}{n}\sum_i^N \mathbf{X}_i\mathbf{1} = \frac{1}{\sum_i n_i}\sum_i^N n_i\mu_i \tag{3}$$

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \mu\mu^\top = \frac{1}{n}\sum_i^N \mathbf{X}_i\mathbf{X}_i^\top - \mu\mu^\top = \frac{1}{\sum_i n_i}\sum_i^N \mathbf{B}_i - \mu\mu^\top \tag{4}$$

Provided these weighted averages are computed in a decentralized and asynchronous fashion, all nodes get the full covariance matrix $\mathbf{C}$. Locally eigendecomposing $\mathbf{C}$ at every nodes finally gives the global PCA solution $\mathbf{U}^\star$.

Remark that $\mathbf{X}_i\mathbf{1}$ and $\mathbf{X}_i\mathbf{X}_i^\top$ are easily computed together in a single pass over $\mathbf{X}_i$. Also notice that once local sufficient statistics are computed, input samples can be dropped without concern. Therefore, such a scheme is suitable when streaming data.

3.2. Gossip protocols for asynchronous decentralized averaging

To compute Equations (3-4) in a decentralized and asynchronous fashion, AG-PCA uses a Sum-Weight Gossip protocol [16, 17].

Sum-Weight protocols are an asynchronous subclass of Gossip consensus algorithms for distributed averaging. Gossip consensus algorithms proceed by iterative linear combinations of node estimates through randomized communications.

An example of Gossip averaging protocol is Newscast [18]. In Newscast, random node pairs regularly average their estimates until reaching consensus. Assuming each node i holds a local vector estimate $\mathbf{v}_i$ in a vector space $\mathcal{V}$, at any random time t two random nodes s and r awake, exchange their estimates and update as follows:

$$\mathbf{v}_s(t+1) = \frac{1}{2}(\mathbf{v}_s(t) + \mathbf{v}_r(t)) \qquad \mathbf{v}_r(t+1) = \frac{1}{2}(\mathbf{v}_s(t) + \mathbf{v}_r(t)) \tag{5}$$

This update rule entails two properties:

- **Mass conservation**. The sum of the estimates over the network is conserved:

$$\sum_i \mathbf{v}_i(t) = \sum_i \mathbf{v}_i(0)$$

- **Convergence to consensus**. The variance of the estimates tends to zero:

$$\lim_{t\to\infty} \sum_i \left| \mathbf{v}_i(t) - \sum_j \mathbf{v}_j(t) \right|^2 = 0$$

A trivial consequence of these properties is $\forall i, \lim_{t\to\infty} \mathbf{v}_i(t) = \frac{1}{N}\sum_j \mathbf{v}_j(0)$. Unfortunately, Newscast requires synchronous updating of random node pairs, thus violating constraint **(C2)**.

Sum-Weight protocols aims at removing pairwise synchronization by adding a new estimate $w_i \in \mathbb{R}$ called weight, with initial value $w_i(0) = 1$. In contrast to Newscast, all nodes send their current sum and weight to randomly selected neighbors following independent Poisson emission clocks and select targets independently at random *without* waiting for their answer. The update rules of the sender node s and receiver node r are modified as follows:

$$\mathbf{v}_s(t+1) = \frac{1}{2}\mathbf{v}_s(t) \qquad w_s(t+1) = \frac{1}{2}w_s(t) \tag{6}$$

$$\mathbf{v}_r(t+1) = \mathbf{v}_r(t) + \frac{1}{2}\mathbf{v}_s(t) \qquad w_r(t+1) = w_r(t) + \frac{1}{2}w_s(t) \tag{7}$$

As first shown in [16] under synchronous assumptions, and then in [17] for the general case, the quotient of the two estimates converges to the desired average.

$$\forall i, \quad \lim_{t\to\infty} \frac{1}{w_i(t)}\mathbf{v}_i(t) = \frac{1}{\sum_j w_j(0)} \sum_j \mathbf{v}_j(0) = \frac{1}{N}\sum_j \mathbf{v}_j(0) \tag{8}$$

Moreover, convergence to the consensus is exponential provided that the network has a sufficient conductance [19]. In this case, the number of message exchanges required to achieve a given estimation error ε scales logarithmically with the number of nodes and ε.

As weighted averages, μ and $\mathbf{C}$ can be estimated using the above-defined Sum-Weight protocol, by defining node-local estimates $\mathbf{a}_i(t), \mathbf{B}_i(t)$ and weights $w_i(t)$ such that:

$$\mathbf{a}_i(0) = \mathbf{X}_i\mathbf{1} \qquad \mathbf{B}_i(0) = \mathbf{X}_i\mathbf{X}_i^\top \qquad w_i(0) = n_i \tag{9}$$

By applying the Gossip protocol defined by Equations (6-7) to $\mathbf{a}_i(t)$,$\mathbf{B}_i(t)$ and $w_i(t)$, we get a covariance estimate $\mathbf{C}_i(t)$:

$$\mathbf{C}_i(t) = \frac{\mathbf{B}_i(t)}{w_i(t)} - \frac{\mathbf{a}_i(t)\mathbf{a}_i(t)^\top}{w_i(t)^2} \tag{10}$$

Note that initial estimates $\mathbf{C}_i(0)$ are the covariance matrices of their corresponding $\mathbf{X}_i$. The limit in (8) shows that each $\frac{\mathbf{a}_i(t)}{w_i(t)}$ tends to the global mean μ and each $\mathbf{C}_i$ tends to the global covariance matrix $\mathbf{C}$:

$$\forall i, \begin{cases} \lim\limits_{t\to\infty} \dfrac{\mathbf{a}_i(t)}{w_i(t)} = \dfrac{\sum_i \mathbf{X}_i^\top \mathbf{1}}{\sum_i n_i} = \dfrac{\sum_i n_i \mu_i}{\sum_i n_i} = \mu \\ \lim\limits_{t\to\infty} \mathbf{C}_i(t) = \dfrac{\sum_i \mathbf{B}_i(0)}{\sum_i w_i(0)} - \mu\mu^\top = \dfrac{1}{n}\sum\limits_i \mathbf{X}_i\mathbf{X}_i^\top - \mu\mu^\top = \mathbf{C} \end{cases} \tag{11}$$

Once each node gets a sufficiently accurate estimate for $\mathbf{C}$, the final PCA result can be locally computed at any node i by eigendecomposition of $\mathbf{C}_i$.

Remark this strategy, which will be referred to as *Late-PCA* in the rest of the paper, is used in [15] in a synchronous consensus framework. Yet it suffers one major drawback: updating estimates $\mathbf{B}_i$ using Equations (6-7) requires transmission of $D \times D$ matrices, which can be too large to exchange on the network. In [5], we thus proposed to reduce matrices $\mathbf{B}_i$ by means of local PCA *before* their transmission, resulting in what we called an *Early-PCA* scheme.

3.3. Gossiping in the compressed domain: Early PCA

There are two main reasons that make the Late PCA approach unwanted, both based on the fact that such a scheme does not benefit from the rank-deficiency of the local covariance matrices implied by $n_i \ll D$. Firstly, the size of exchanged information is homogeneous to the input statistics ($D \times D$), not to the output result ($D \times q$). Therefore, we vainly exchange information that will be canceled in the end, because dimensionality reduction happens *after* the distributed averaging phase. Secondly, the distributed nature of the process does not allow any computational advantage for high dimension, since all nodes have to perform the same $\mathcal{O}(D^3)$ eigendecomposition operation. Consequently, if the eigendecomposition of $\mathbf{C}$ is the computational bottleneck due to large values of D, the distributed scheme is unlikely to bring any speed-up gain.

The Early-PCA approach aims at moving the dimension reduction step *before* the distributed averaging step, in order to take advantage of the rank deficiency

of $\mathbf{C}_i$. To this purpose, we reformulate the updates rules in Equations (6-7) so as to handle eigenpairs instead of the full matrices $\mathbf{B}_i$. This allows us to implicitly compute the average of very large matrices without ever resorting to their explicit form but rather using their factorized expression.

Observe that the senders update rule (6) is a simple uniform scaling. Assuming s holds the decomposed form $\mathbf{U}_s\mathbf{L}_s\mathbf{U}_s^\top$ of its estimate $\mathbf{B}_s$, the update is simply:

$$\mathbf{U}_s(t+1) = \mathbf{U}_s(t) \quad \mathbf{L}_s(t+1) = \frac{1}{2}\mathbf{L}_s(t) \tag{12}$$

This update leads to the emission procedure of AGPCA to be concurently run at every node of the network, as presented in Algorithm 1.

Adapting the receivers update rule (7) is slightly more involving since we need to compute the eigendecomposition of a sum of two matrices given their own eigendecompositions. We propose to rely on the well-known Orthogonal Iterations technique (as in [6]) to fit an orthonormal basis to the principal subspace of an input matrix by iterating QR decompositions. Assume $\mathbf{B}_s(t)$ and $\mathbf{B}_r(t)$ are respectively factorized as $\mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top$ and $\mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top$. Starting from a random $D \times q$ basis $\mathbf{Q}_0$, and denoting by $QR(\cdot)$ the economy QR decomposition, we iteratively compute

$$\begin{aligned}\mathbf{Q}_{\tau+1}\mathbf{R}_{\tau+1} &= QR((\mathbf{B}_s + \mathbf{B}_r)\mathbf{Q}_\tau) \\ &= QR(\mathbf{U}_s\mathbf{L}_s(\mathbf{U}_s^\top\mathbf{Q}_\tau) + \mathbf{U}_r\mathbf{L}_r(\mathbf{U}_r^\top\mathbf{Q}_\tau))\end{aligned} \tag{13}$$

$\mathbf{Q}_\tau$ tends to become an orthonormal basis for the q-principal subspace of $\mathbf{B}_s(t) + \mathbf{B}_r(t)$, with corresponding eigenvalues on the diagonal of $\mathbf{R}_\tau$. Thus, $\mathbf{Q}_\infty$ gives $\mathbf{U}_r(t{+}1)$ and the diagonal entries of $\mathbf{R}_\infty$, denoted by $diag(\mathbf{R}_\infty)$, give $\mathbf{L}_r(t{+}1)$.

The parentheses in Equation (13) are of great importance. Indeed, observe that instead of expanding $\mathbf{U}_s\mathbf{L}_s\mathbf{U}_s^\top$ and $\mathbf{U}_r\mathbf{L}_r\mathbf{U}_r^\top$, we first multiply $\mathbf{U}_s^\top$ and $\mathbf{U}_r^\top$ by $\mathbf{Q}_\tau$, resulting in a $q \times q$ matrix. By doing so, we never store any $D \times D$ matrix, and the QR decomposition is rather performed on a $D \times q$ matrix.

Finally, the dominant eigenvectors $\mathbf{U}^\star$ of $\mathbf{C}_i$ are obtained by locally computing its eigendecomposition using Orthogonal Iterations (parentheses are also added to highlight the computational gain):

$$\mathbf{C}_i = \frac{1}{w_i(t)}(\mathbf{U}_i\mathbf{L}_i)(\mathbf{U}_i^\top) - \frac{1}{w_i(t)^2}(\mathbf{a}_i(t))(\mathbf{a}_i(t)^\top) \tag{14}$$

This update leads to the reception procedure concurrently run at every node and presented in Algorithm 2.

Since typically $q \ll D$, combining Algorithms 1 and 2 allows to define an averaging protocol that exchanges $D \times q$ messages while staying equivalent to the original Sum-Weight protocol defined by Equations (6-7), provided that the rank of $\mathbf{X}\mathbf{X}^\top$ is lower than q. This is formally supported in the theoretical analysis presented in Section 5.

3.4. Dual relation between covariance and Gramian eigendecompositions

While Early-PCA gets rid of $D \times D$ entities in the networking step, we can further avoid $D \times D$ matrices in the entire algorithm. This is particularly useful when covariance matrices do not fit in node memory. To this end, we must avoid explicit computation and storage of the initial $\mathbf{B}_i(0)$ locally computed in Equation (9), reminding that we are only interested in their q principal subspace. Thankfully, the eigendecompositions $\mathbf{B}_i = \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$ can be obtained from those of the $n_i \times n_i$ local Gram matrices $\mathbf{X}_i^\top\mathbf{X}_i = \mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i^\top$, through their well-known relation with the Singular Value Decomposition (SVD) of $\mathbf{X}$. Indeed,

$$\mathbf{B}_i^2 = \mathbf{X}_i\mathbf{X}_i^\top\mathbf{X}_i\mathbf{X}_i^\top = \mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i^\top\mathbf{X}_i^\top = (\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}})\mathbf{\Lambda}_i^2(\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}})^\top$$
$$\text{and} \qquad (\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}})^\top(\mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}}) = \mathbf{I}$$

Then,

$$\mathbf{U}_i = \mathbf{X}_i\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}} \qquad \text{and} \qquad \mathbf{L}_i = \mathbf{\Lambda}_i \tag{15}$$

We thus compute, store and factorize $\mathbf{X}_i^\top\mathbf{X}_i$ (which is $n_i \times n_i$) instead of $\mathbf{X}_i\mathbf{X}_i^\top$ (which is $D \times D$) and obtain $\mathbf{U}_i$ and $\mathbf{L}_i$ with no additional storage cost.

4. Asynchronous Gossip PCA for Distributed Coordinates scenario

In this section, we present the DC scenario and show that AGPCA can solve it as well. Contrarily to the DS scenario, each node hosts a local sample $\mathbf{Z}_i \in \mathbb{R}^{D_i \times n}$ made of $D_i \ll D$ dimensions of the same set of n observations. The full data $\mathbf{Z}$ corresponds to stacking the $\mathbf{Z}_i$ in rows:

$$\mathbf{Z}^\top = [\mathbf{Z}_1^\top, \ldots, \mathbf{Z}_N^\top] \tag{16}$$

Observations thus lie in the direct sum of the locally observed subspaces $\bigoplus_i \mathbb{R}^{D_i}$. Let $\mu = \mathbf{Z}\mathbf{1}/n$ denote the data mean and $\tilde{\mathbf{Z}} = \mathbf{Z} - \mu\mathbf{1}^\top$ the centered data.

In DC scenarios, it is useless to provide nodes with a $D \times q$ orthogonal basis since nodes only hold part of the input vectors and are thus unable to perform

Algorithm 3 AGPCA-DC Emission Procedure

$\mathbf{C}_i \leftarrow \mathbf{Z}_i\mathbf{Z}_i^\top$; $w_i \leftarrow D_i$
$(\mathbf{U}_i, \mathbf{L}_i) \leftarrow$ *eigendecompose*$(\mathbf{C}_i)$
$\mathbf{V}_i \leftarrow \mathbf{Z}_i^\top \mathbf{U}_i \mathbf{L}_i^{-\frac{1}{2}}$
loop
  $j \leftarrow$ *randomNeighbor*(i)
  $(\mathbf{V}_i, \mathbf{L}_i, w_i) \leftarrow \frac{1}{2}(\mathbf{V}_i, \mathbf{L}_i, w_i)$
  Send $(\mathbf{V}_i, \mathbf{L}_i, w_i)$ to j
end loop

Algorithm 4 AGPCA-DC Reception Procedure

loop
  Upon receipt of $(\mathbf{V}_j, \mathbf{L}_j, w_j)$
  $w_i \leftarrow w_i + w_j$
  $\mathbf{Q}_0 \leftarrow \mathbf{V}_i$
  for $t \in [0, max\_t[$ **do**
    $(\mathbf{Q}_{t+1}, \mathbf{R}_{t+1}) \leftarrow$ *QR*$(\mathbf{V}_i\mathbf{L}_i\mathbf{V}_i^\top\mathbf{Q}_t + \mathbf{V}_j\mathbf{L}_j\mathbf{V}_j^\top\mathbf{Q}_t)$
  end for
  $\mathbf{V}_i \leftarrow \mathbf{Q}_{max\_t}$; $\mathbf{L}_i \leftarrow$ *diag*$(\mathbf{R}_{max\_t})$
end loop

the projection. Instead, AGPCA directly provides a compressed representation $\mathbf{Y} \equiv \mathbf{U}^{\star\top}\tilde{\mathbf{Z}}$ that account for all dimensions of all observations.

Using the same ideas that allowed us to improve the efficiency in the DS case, we rely on the duality between the covariance and the Gram eigendecompositions. Recall that

$$\mathbf{C} = \frac{1}{n}\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top = \mathbf{U}^\star\mathbf{L}^\star\mathbf{U}^{\star\top} \tag{17}$$

Introducing the SVD of the centered data $\tilde{\mathbf{Z}} = \mathbf{U}\mathbf{L}^{\frac{1}{2}}\mathbf{V}^\top$, where $\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top = \mathbf{U}\mathbf{L}\mathbf{U}^\top$ and $\tilde{\mathbf{Z}}^\top\tilde{\mathbf{Z}} = \mathbf{V}\mathbf{L}\mathbf{V}^\top$, we have $\mathbf{U}^\star = \mathbf{U}$ and $\mathbf{L}^\star = \mathbf{L}/n$. Consequently,

$$\mathbf{Y} = \mathbf{U}^{\star\top}\tilde{\mathbf{Z}} = n\mathbf{L}^{\star\frac{1}{2}}\mathbf{V}^\top \tag{18}$$

Define $\mathbf{D} = \tilde{\mathbf{Z}}^\top\tilde{\mathbf{Z}}/D$.

Clearly its eigendecomposition is $\mathbf{V}(\mathbf{L}/D)\mathbf{V}^\top = \mathbf{V}(n\mathbf{L}^\star/D)\mathbf{V}^\top$, and

$$\begin{aligned}\mathbf{D} &= \frac{1}{D}(\mathbf{Z} - \frac{1}{n}\mathbf{Z}\mathbf{1}\mathbf{1}^\top)^\top(\mathbf{Z} - \frac{1}{n}\mathbf{Z}\mathbf{1}\mathbf{1}^\top) \\ &= \left(I - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\frac{\mathbf{Z}^\top\mathbf{Z}}{D}\left(I - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right) \end{aligned} \tag{19}$$

Observing that Equation 19 only involves $\mathbf{Z}^\top\mathbf{Z}/D$, we remark that $\mathbf{Z}^\top\mathbf{Z}/D = (\sum_i \mathbf{Z}_i^\top\mathbf{Z}_i)/(\sum_i D_i)$ is a weighted average of the locally-computable uncentered Gram matrices $\mathbf{Z}_i^\top\mathbf{Z}_i$. This average can be computed using the same protocol as for the DS case, defined by Equations (12-13). Using the factors $\mathbf{V}$ and $\mathbf{L}/D$ of $\mathbf{D}$ in Equation (18), we can obtain the compressed representations scaled by $\sqrt{D}$. For most applications, this scaling factor has no impact. In applications for which the output basis needs to be orthonormal (and not only orthogonal), the scale can be recovered by gossiping D_i with an initial weight on 1, provided the size of the network is known at each node.

4.1. Early-PCA aggregation in DC scenarios

Clearly, the Early-PCA aggregation scheme described in Section 3 also applies: We only need to compute and exchange the eigendecomposed forms $\mathbf{Z}_i^\top\mathbf{Z}_i = \mathbf{V}_i\mathbf{L}_i\mathbf{V}_i^\top$. Since local Gram matrices are $n \times n$ and are much larger than the local correlation matrices (of size $D_i \times D_i \ll n \times n$), we use the same dual property as in Section 3 to obtain the Gram eigendecomposition from the local correlation matrices $\mathbf{Z}_i\mathbf{Z}_i^\top = \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$:

$$\mathbf{Z}_i = \mathbf{U}_i\mathbf{L}_i^{\frac{1}{2}}\mathbf{V}_i^\top \quad \Rightarrow \quad \mathbf{V}_i = \mathbf{Z}_i^\top\mathbf{U}_i\mathbf{L}_i^{-\frac{1}{2}}$$

The initial $\mathbf{V}_i\mathbf{L}_i\mathbf{V}_i^\top$ can then be obtained at a low storage and computation cost. The corresponding emission and reception procedures are shown in Algorithms 3 and 4.

Once the averaging protocol has provided all nodes with the eigendecomposition of $\mathbf{Z}^\top\mathbf{Z}/D$, we can locally apply Orthogonal Iterations to Equation (19) to obtain the eigendecomposition $\mathbf{V}(\mathbf{L}^\star/D)\mathbf{V}^\top$ of $\mathbf{D}$ at all nodes. Each node finally computes the full compressed data using $\mathbf{Y} = n(\mathbf{L}^\star/D)^{\frac{1}{2}}\mathbf{V}^\top$.

This shows that using our AGPCA strategy, we can efficiently solve distributed PCA for both DS and DC scenarios by just swapping the roles of covariance and Gram matrices. A nice consequence is that theoretical results in one case hold in the other one.

4.2. Projection of subsequent observations

In DC scenarios, projecting future observations is not as straightforward as in the DS case, because new observations generate new entries in *all* nodes. Thus, all nodes have to participate in the projection of new observations. Remark the compressed representation $\mathbf{y}$ of a new observation $\mathbf{z}$ is computed by:

$$\mathbf{y} = n(\mathbf{L}^\star/D)^{-\frac{1}{2}}\mathbf{V}^\top\frac{\mathbf{Z}^\top\mathbf{z}}{D} \tag{20}$$

Observing that $\mathbf{Z}^\top\mathbf{z}/D = \sum_i \mathbf{Z}_i^\top\mathbf{z}_i/D$ is also a weighted average, $\mathbf{y}$ can easily be recovered by simply Gossiping $\mathbf{Z}_i^\top\mathbf{z}_i$ with initial weights D_i and combining the result with the previously obtained factors $\mathbf{V}$ and $\mathbf{L}/D$.

5. Theoretical analysis

In this section, we theoretically prove that AGPCA yields the same solution as a centralized PCA, provided assumptions are made on the rank of input data. We limit our analysis to the DS case, as the same arguments can easily be translated to the DC case by substituting the covariance matrices by the Gram matrices. Noting $\mathbf{X} \in \mathbb{R}^D$ the input data, this result is expressed in the following theorem:

Theorem 1. *If $rank(\mathbf{X}) \leq q$, then AGPCA yields the exact solution of a PCA run over $\mathbf{X}$:*

$$\forall i, \mathbf{U}_i^\top\mathbf{X}_i = \mathbf{U}^{\star\top}\mathbf{X}_i \tag{21}$$

PROOF. The centralized PCA uses the following eigendecomposition:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \mu\mu^\top = \mathbf{U}^\star\mathbf{L}^\star\mathbf{U}^{\star\top} \tag{22}$$

The proof sketch is as follows: We first show that the local decompositions $\mathbf{X}_i\mathbf{X}_i^\top = \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top$ perfectly reconstruct the local data $\mathbf{X}_i$. Then, we show our Gossip protocol aggregating the $\mathbf{U}_i$ allows for a perfect reconstruction of $\sum_i \mathbf{X}_i\mathbf{X}_i^\top$ at every node. The proof is completed by remarking AGPCA provides the eigendecomposition of the covariance matrices obtained from the reconstruction of $\sum_i \mathbf{X}_i\mathbf{X}_i^\top$.

To prove local factorizations induce no loss of information, we use the assumption that $rank(X) \leq q$. We thus have:

$$rank(\mathbf{X}) \leq q \Rightarrow rank(\mathbf{X}\mathbf{X}^T) \leq q \tag{23}$$

Let us denote $\mathcal{H}_q = span(\mathbf{X}) \subset \mathbb{R}^q$. Since at any node i, $\mathbf{X}_i$ is a subset of $\mathbf{X}$, we have $span(\mathbf{X}_i) \subseteq \mathcal{H}_q$, and consequently $rank(\mathbf{X}_i\mathbf{X}_i^\top) \leq q$. It follows that the initial local decompositions keeping only the q leading eigenvectors (the others being associated with null eigenvalues) perfectly reconstruct the local correlation matrices:

$$\forall i, \mathbf{U}_i\mathbf{L}_i\mathbf{U}_i^\top = \mathbf{X}_i\mathbf{X}_i^\top = \mathbf{B}_i \tag{24}$$

Considering the Gossip averaging, we first prove our update rules in Equations (12-13) are equivalent to the Gossip updates in Equations (6-7). Remark that $\mathbf{a}_s$ and w_s are explicitly updated using Equation (6-7). Checking validity of the sender update of $\mathbf{B}_s$ (line 6 in Algorithm 1) is trivial:

$$\mathbf{B}_s(t+1) = \mathbf{U}_s(t)\frac{1}{2}\mathbf{L}_s(t)\mathbf{U}_s(t)^T = \frac{1}{2}\mathbf{B}_s(t).$$

On the receiver side, provided that $span(\mathbf{U}_s(t)) \subseteq \mathcal{H}_q$ and $span(\mathbf{U}_r(t)) \subseteq \mathcal{H}_q$, using Othogonal Iterations to obtain the q leading eigenvectors of the weighted sum has two interesting properties: The reconstruction is perfect, that is

$$\mathbf{U}_r(t+1)\mathbf{L}_r(t+1)\mathbf{U}_r(t+1) = \mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top + \frac{1}{2}\mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top \tag{25}$$

and the obtained basis is in the same subspace, *i.e.*, $\mathbf{U}_r(t+1) \subseteq \mathcal{H}_q$. These properties come from the fact that $span(\mathbf{U}_r \cup \mathbf{U}_s) \subseteq \mathcal{H}_q$, and thus $rank(\mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top + \mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top/2) \leq q$ Remarking that initially $\forall i, span(\mathbf{U}_i(0)) \subset \mathcal{H}_q$, it follows by induction that at any time t:

$$\begin{aligned}
\mathbf{B}_r(t+1) &= \mathbf{U}_r(t+1)\mathbf{L}_r(t+1)\mathbf{U}_r(t+1) \\
&= \mathbf{U}_r(t)\mathbf{L}_r(t)\mathbf{U}_r(t)^\top + \frac{1}{2}\mathbf{U}_s(t)\mathbf{L}_s(t)\mathbf{U}_s(t)^\top \\
&= \mathbf{B}_r(t) + \frac{1}{2}\mathbf{B}_s(t)
\end{aligned}$$

That is, Equation (13) is equivalent to Equation (7).

Convergence of Equations (6-7) to the network average at all nodes has already been proved [16, 17]. For completeness though, we show that Equations (6-7) drive all $\mathbf{C}_i$ to $\mathbf{C}$. Let us introduce the Euclidean errors E_i between covariances

locally reconstructed by our protocol and the exact covariance matrix:

$$\begin{aligned}\forall i, \quad E_i(t) &= \|\mathbf{C}_i(t) - \mathbf{C}\|_F^2 \\ &= \left\| \frac{1}{w_i(t)}\mathbf{B}_i(t) - \frac{1}{w_i(t)^2}\mathbf{a}_i(t)\mathbf{a}_i(t)^\top - \frac{1}{n}\mathbf{X}\mathbf{X}^\top + \mu\mu^\top \right\|_F^2 \\ &\leq \left\| \frac{1}{w_i(t)}\mathbf{B}_i(t) - \frac{1}{n}\mathbf{X}\mathbf{X}^\top \right\|_F^2 + \left\| \frac{1}{w_i(t)^2}\mathbf{a}_i(t)\mathbf{a}_i(t)^\top - \mu\mu^\top \right\|_F^2\end{aligned}$$

From Equations (9), and (3-4), we have $n = \sum_i w_i(0)$, $\mathbf{X}\mathbf{X}^\top = \sum_i \mathbf{B}_i(0)$ and $\mu = (\sum_i \mathbf{a}_i(0))/(\sum_i w_i(0))$. Then, we obtain

$$E_i(t) \leq \left\| \frac{\mathbf{B}_i(t)}{w_i(t)} - \frac{\sum_j \mathbf{B}_j(0)}{\sum_j w_j(0)} \right\|_F^2 + \left\| \frac{\mathbf{a}_i(t)\mathbf{a}_i(t)^\top}{w_i(t)^2} - \frac{(\sum_j \mathbf{a}_j(0))(\sum_j \mathbf{a}_j(0))^\top}{(\sum_j w_j(0))^2} \right\|_F^2 \tag{26}$$

According to Equations (6-7), each single entry of $(\mathbf{B}_i, \mathbf{a}_i, w_i)$ is identically updated. The following Lemma then shows that every quotient of $\mathbf{B}_i$ or $\mathbf{a}_i$ and w_i tends to its network average (the proof is deferred to Appendix A):

Lemma 2. *Under update rules in Equations (6-7):*

$$\forall i, \quad \lim_{t\to\infty} \frac{\mathbf{B}_i(t)}{w_i(t)} = \frac{\sum_j \mathbf{B}_j(0)}{\sum_j w_j(0)} \quad \textit{and} \quad \lim_{t\to\infty} \frac{\mathbf{a}_i(t)}{w_i(t)} = \frac{\sum_j \mathbf{a}_j(0)}{\sum_j w_j(0)} \tag{27}$$

In turn, it implies that both terms in Equation (26) tend to zero, and therefore entails the convergence of local estimates to the covariance matrix:

$$\lim_{t\to\infty} \mathbf{C}_i(t) = \mathbf{C} \tag{28}$$

□

6. Experiments

In this section, we experimentally evaluate AGPCA, both on synthetic and natural data. Experiments for Distributed Samples and Distributed Coordinates scenarios are reported separately. Synthetic data were obtained by sampling n observations from a D-dimensional Gaussian distribution, which covariance matrix

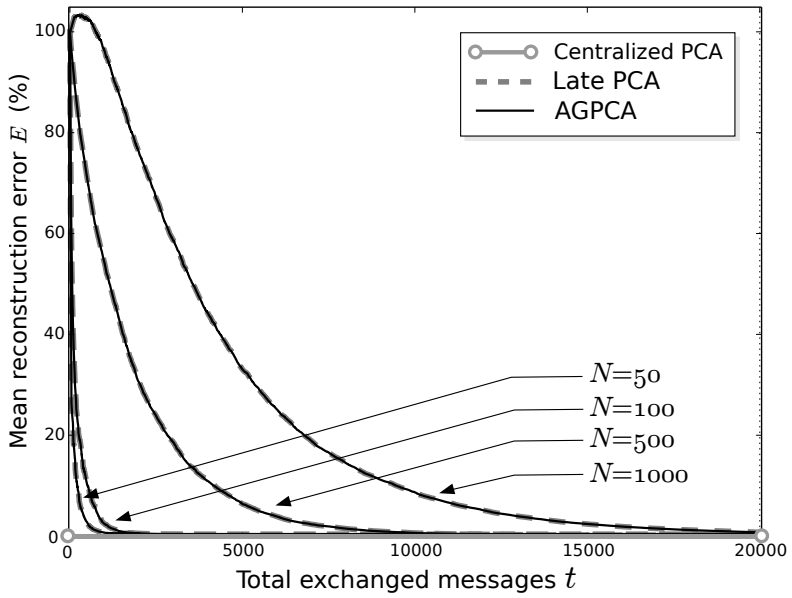

Figure 1: Convergence of AGPCA compared to a Late PCA strategy and the centralized PCA solution on a synthetic dataset drawn from a Gaussian distribution ($D = 200, p = 30, n = 10000$), spread on $N = 100$ nodes. Here q is set to D.

was arbitrarily generated with rank $p \ll D$ and such that all dimensions are correlated in $\mathbb{R}^D$. For natural data, we used the MNIST handwritten digits dataset, which contains 60000 grayscale images of 28×28 pixels, that is, $D = 784$ [20]. Unless specified, the network is assumed fully connected, sender-receiver pairs being uniformly selected at random.

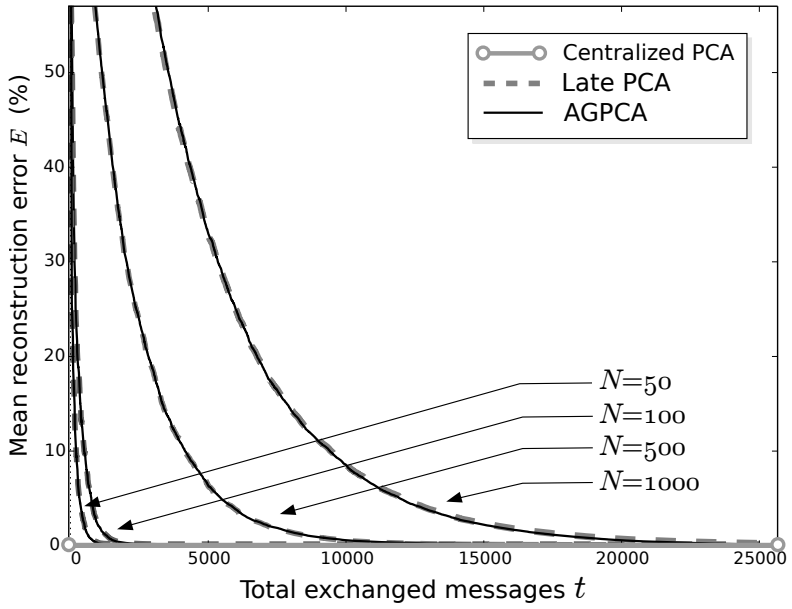

Figure 2: Convergence of AGPCA on the same data and network as Figure 1, but setting $q = 30 = p \ll D$.

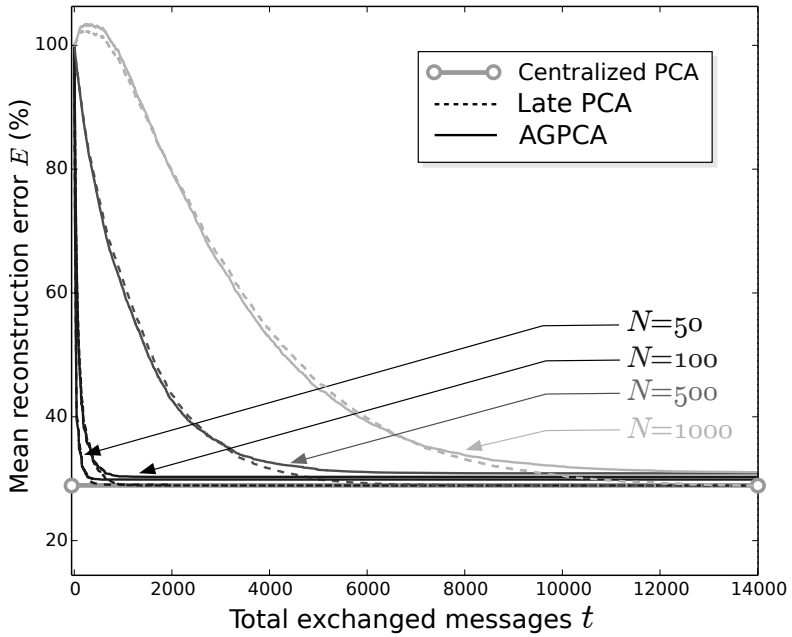

Figure 3: Convergence of AGPCA on the same data and network as Figure 1, but setting $q = 10 < p$.

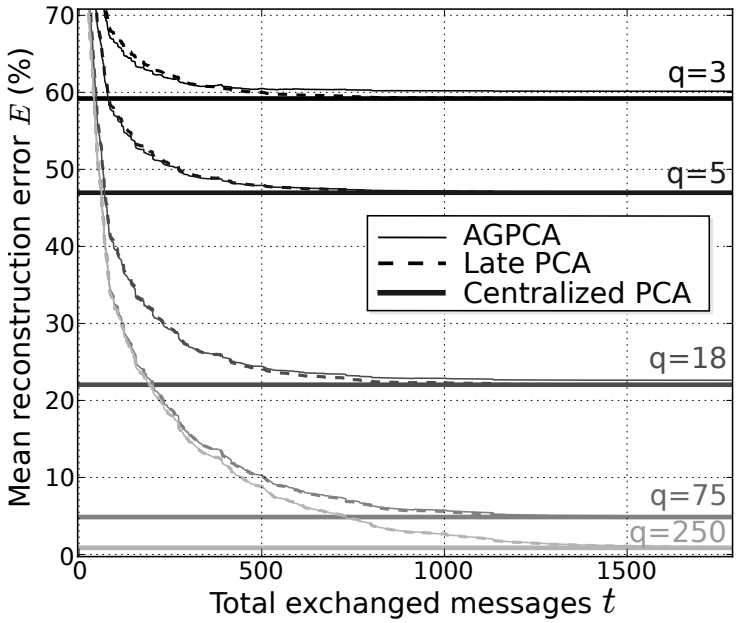

Figure 4: Convergence of AGPCA on MNIST spread over $N = 100$ nodes, for various q.

6.1. Distributed Samples scenario

In DS scenario, the objective of AGPCA is two-fold : *(i)* providing all nodes with the same projection matrix $\mathbf{U}$, *(ii)* making $\mathbf{U}$ as close as possible to the solution $\mathbf{U}^\star$ of a centralized PCA solution computed over the aggregated network data $\mathbf{X}$. We thus measure the average Euclidean reconstruction error between the locally reconstructed covariances $\mathbf{C}_i$ and the exact $\mathbf{C}$, defined by $E = \|\mathbf{C}_i(\infty) - \mathbf{C}\|_F^2 / \|\mathbf{C}\|_F^2$. We considered three cases: $q = D$, $q = p$, and $q < p$:

- When $q = D$, the centralized PCA solution $\mathbf{U}^\star$ reconstructs the covariance with zero error. In this case, experiments on synthetic data show that AGPCA asymptotically provides all nodes with the exact $\mathbf{U}^\star$. Figure 1 illustrates the convergence to this optimum versus time and show that both Early and Late PCA strategies allow a perfect reconstruction.

- When $q = p \ll D$, Figure 2 confirms the theoretical result of Theorem 1, as convergence to the optimum still holds if the intrinsic dimension p of $\mathbf{X}$ is lower than q.

- By contrast, when $q < p$, the ideal projection itself accounts for less than 100% of the variance in the data. In such case, AGPCA still guarantees

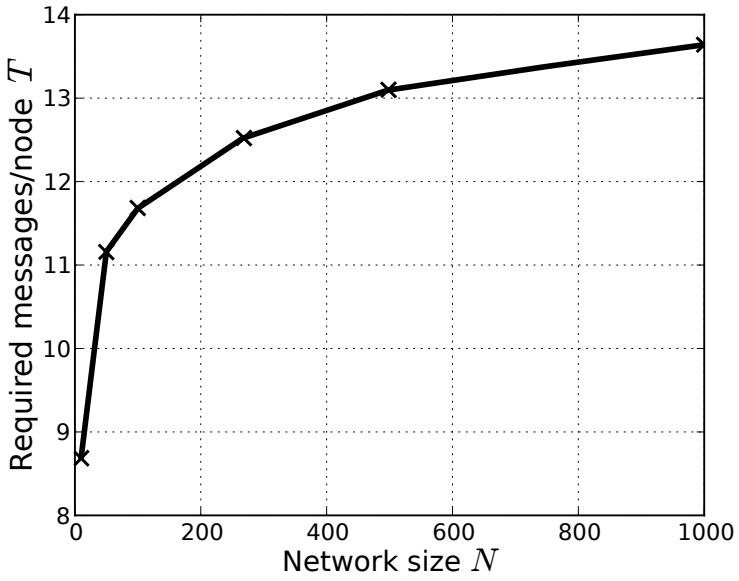

Figure 5: Average number of message emissions per node required to converge against the number of nodes in the network, evaluated on MNIST with $q = 50$.

that all nodes converge to the same projection matrix $\mathbf{U}$. This ensures that any observation is projected into the same output space whichever the node we consider. However, the error induced by the Orthogonal Iterations steps leads to a slight deviation from the optimal global PCA solution $\mathbf{U}^\star$. Experiments reported in Figure 3 for various q reveal that even for high compression rates, the deviation from the optimal basis is still very low.

Similar experiments were conducted on the MNIST dataset, whose intrinsic dimensionality was shown in previous works to be much lower than $D = 784$. Figure 4 highlights that AGPCA achieves the same reconstruction error as a centralized PCA up to the numerical precision for $q \geq 75$. For lower values, it is slightly suboptimal, but the deviation is kept under 1% until $q = 3$, and always under 2%. Figure 4 also shows that the convergence rate of the aggregation protocol is not impacted by switching from a Late PCA to an Early PCA scheme. Interestingly, lower values for q bring faster convergence.

Concerning communication costs, the efficiency of AGPCA is illustrated in Figure 5. This figure displays the number of messages each node has to emit in order to reach convergence (convergence is assumed when E improves by less than 0.01% between two message events). This number of messages per node appears to logarithmically scale with the number of nodes in the network. This

ensures easy scaling to large networks.

6.2. Distributed Coordinates scenario

In a DC scenario, our accuracy criterion is still the Euclidean reconstruction error, but it is now computed as $E = \|\mathbf{Y}_i^T\mathbf{Y}_i - \mathbf{X}^T\mathbf{X}\|$. To evaluate AGPCA in such scenarios, we spread a synthetic dataset similar to Figure 1 by assigning a uniformly drawn number of dimensions D_i to each node i, such that node 1 gets the D_1 first dimensions, node 2 gets the next D_2 and so forth, and such that $\sum_i D_i = D = 10000$. In this scenario, Figure 6 shows that AGPCA behaves identically to the DS case, which is a logical consequence of using the same strategy.

6.3. Influence of network connectivity

Network topology usually has a great impact on speed, accuracy or even applicability of distributed learning algorithms. Gossip protocols have long been acclaimed for their fine behavior in a wide range of networking situations. It is worth highlighting that the Sum-Weight protocol used in AGPCA can be used to build asynchronous generalizations of broadcast, spanning tree, workers-master and scale-free connectivity:

- In an asynchronous broadcast scheme, senders still waken independently, but the same message is sent to all their neighbors. To obtain a broadcast protocol, we just need to replace the $1/2$ coefficient in (6-7) by $1/N$ to respect mass conservation.

- In an asynchronous spanning tree communication scheme, the network has only $N-1$ links, which is the minimal number of links such that the network stays connected. In our experiments, we used the worst conditioned non-degenerate tree topology where nodes have at most 3 neighbors.

- In a workers-master scheme, $N-1$ nodes called workers hold local datasets and can only send messages to a single master node. This master is allowed to send messages to all workers, and may hold a local sample or not (if not, its estimates are initialized to 0). In our experiments, we considered two settings. In the first one, the workers and the master are selected for emission with the same probability, *i.e.*, the master works at the same frequency as the workers. In the second one, the master is selected N times more often than workers, *e.g.*, assuming it is a high-throughput server.

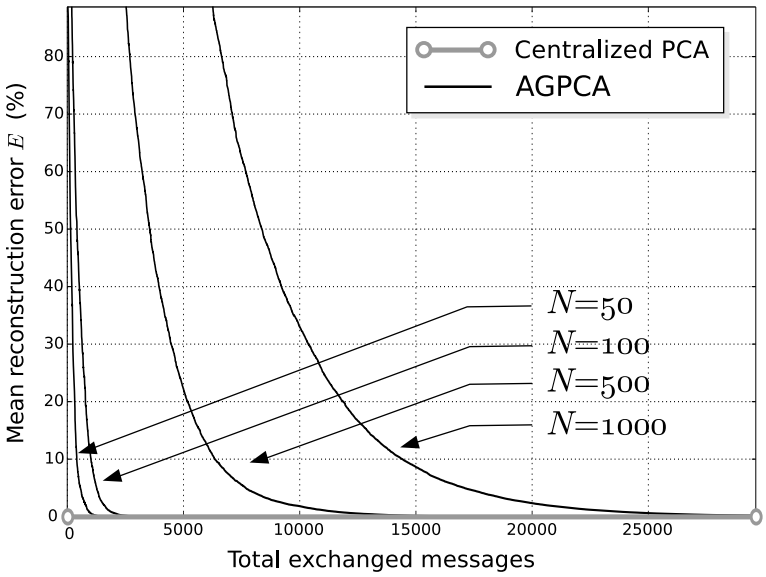

Figure 6: Convergence of AGPCA in a Distributed Coordinates scenario on synthetic data similar to Figure 1 ($D = 10000$, $n = 200, p = 30$). Each node holds a number of dimensions of the complete data drawn uniformly in $\{1, \ldots, \frac{2D}{N} - 1\}$. Here, $q = p = 30$.

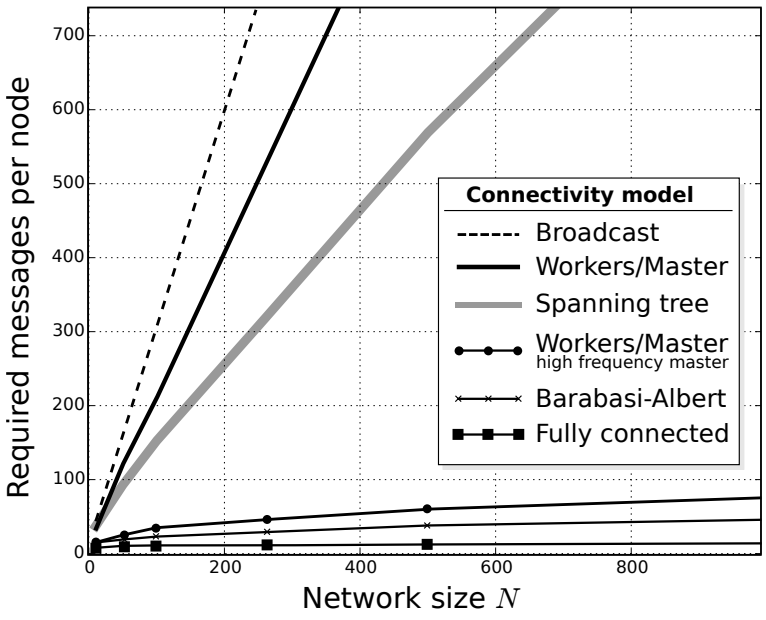

Figure 7: Number of messages per node to reach convergence for various connectivity and networking models. Here, $q = p = 30$ (same data as Figure 6).

- An example of a scale-free network is the Barabási-Albert (BA) random graph model, which has a power-law distribution of node degrees. Nodes are connected to random neighbors such that the number of nodes having k neighbors scales in k^{-3}.

We run AGPCA for each of these connectivity models, varying the number of involved nodes (synthetic data similar to Figure 1). Figure 7 displays the number of messages per node to reach convergence (in the same sense as Figure 5) and compares it to the fully-connected setup we considered so far. Broadcast, workers-master and spanning tree schemes exhibit a poor scaling which appears linear with the number of nodes. By contrast, full and BA connectivity show a logarithmic dependence of the communication costs on the network size. In the workers-master case, we can recover a logarithmic scaling by allowing the master to communicate N-times faster than the workers (in practice, such performances at the master can be unrealistic).

The poor behavior of the broadcast scheme can be explained by remarking that senders always emit $N-1$ messages containing *identical* estimates. In the point-to-point case each message integrates the contribution of *all* preceding updates, thus making estimates mixing much faster. Concerning the spanning tree topology, its bad mixing properties have long been studied in the Gossip protocols and rumor spreading litterature. Due to its high average path length, information must flow, on average, through a much larger number of intermediary nodes to transit between any two peers.

To summarize, AGPCA, like asynchronous Gossip protocols in general, are best-suited for networks with maximally randomized communications and lowest average path lengths.

7. Conclusion

We presented an asynchronous and decentralized algorithm to solve PCA when data is spread over a network. Based on the integration of a dimensionality reduction operator into a Sum-Weight gossip averaging protocol, it is best suited for large setups with high extrinsic dimension, massive samples and large networks. Unlike other algorithms, it is applicable both in Distributed Samples and Distributed Coordinates scenarios, thanks to the duality between covariance and Gram matrices decompositions. Our theoretical and experimental studies show that it is formally equivalent to running a traditional PCA when the complete data has an intrinsic dimension lower than the output dimension, otherwise providing a low-error approximation of the optimum.

Perspectives include application to large-scale dimension reduction problems, where traditional methods are unapplicable, such as signatures compression in web-scale multimedia retrieval. Besides, while we considered a static scenario where data is provided all at once, Gossip protocols also enjoy nice dynamics when data and/or connectivity are time-evolving. Our approach could then be extended to deal with time-related phenomena, such as concept drift and dynamic networks.

Acknowledgements

This work is funded by the Culture 3D Cloud project as part of the french Funds for Digital Societies.

References

[1] K. Pearson, Liii. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11) (1901) 559–572.

[2] H. Hotelling, Analysis of a complex of statistical variables into principal components., Journal of educational psychology 24 (6) (1933) 417.

[3] C. M. Bishop, et al., Pattern recognition and machine learning, Vol. 1, springer New York, 2006.

[4] A. Bertrand, M. Moonen, Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed pca, Signal Processing 104 (2014) 120–135.

[5] J. Fellus, D. Picard, P.-H. Gosselin, et al., Dimensionality reduction in decentralized networks by gossip aggregation of principal components analyzers, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013, pp. 171–176.

[6] D. Kempe, F. McSherry, A decentralized algorithm for spectral analysis, Journal of Computer and System Sciences 74 (1) (2008) 70–83.

[7] A. Wiesel, A. O. Hero, Decomposable principal component analysis, Signal Processing, IEEE Transactions on 57 (11) (2009) 4369–4377.

[8] Z. Meng, A. Wiesel, A. O. Hero, Distributed principal component analysis on networks via directed graphical models, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE, 2012, pp. 2877–2880.

[9] C. Ordonez, N. Mohanam, C. Garcia-Alvarado, Pca for large data sets with parallel data summarization, Distributed and Parallel Databases 32 (3) (2014) 377–403.

[10] S. B. Korada, A. Montanari, S. Oh, Gossip PCA, in: Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, ACM, 2011, pp. 209–220.

[11] P. Bruneau, M. Gelgon, F. Picarougne, Aggregation of probabilistic PCA mixtures with a variational-bayes technique over parameters, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 702–705.

[12] M. E. Tipping, C. M. Bishop, Mixtures of probabilistic principal component analyzers, Neural computation 11 (2) (1999) 443–482.

[13] A. Nikseresht, M. Gelgon, Gossip-based computation of a gaussian mixture model for distributed multimedia indexing, Multimedia, IEEE Transactions on 10 (3) (2008) 385–392.

[14] A. Nikseresht, Estimation de modèles de mélange probabilistes: une proposition pour un fonctionnement réparti et décentralisé, Ph.D. thesis, Université de Nantes (2008).

[15] S. V. Macua, P. Belanovic, S. Zazo, Consensus-based distributed principal component analysis in wireless sensor networks, in: Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on, IEEE, 2010, pp. 1–5.

[16] D. Kempe, A. Dobra, J. Gehrke, Gossip-based computation of aggregate information, in: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, FOCS '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 482–.

[17] F. Iutzeler, P. Ciblat, W. Hachem, Analysis of sum-weight-like algorithms for averaging in wireless sensor networks, CoRR abs/1209.5912.

[18] M. Jelasity, W. Kowalczyk, M. Van Steen, Newscast computing, Tech. rep., Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands (2003).

[19] D. Shah, Gossip algorithms, Now Publishers Inc, 2009.

[20] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[21] A. Rhodius, On the maximum of ergodicity coefficients, the dobrushin ergodicity coefficient, and products of stochastic matrices, Linear algebra and its applications 253 (1) (1997) 141–154.

[22] E. Seneta, Non-negative matrices and Markov chains, Springer, 2006.

Appendix A. Proof of Lemma 2

PROOF. Let's independently consider any entry of $\mathbf{B}_i$ or $\mathbf{a}_i$, denoted using the free variable x_i. Let $\mathbf{x} = (x_i)_i$ gather all the corresponding entries from each node i. Using the definition of $\mathbf{x}$, we can rewrite Equations (6-7) into a matrix form:

$$\mathbf{x}(t+1)^\top = \mathbf{x}(t)^\top \mathbf{K}(t) \quad \text{where} \quad \mathbf{K}(t) \equiv \mathbf{I} + \frac{1}{2}\mathbf{e}_s(\mathbf{e}_r - \mathbf{e}_s)^\top \tag{A.1}$$

$\mathbf{K}(t)$ is a random $N \times N$ matrix where (s, r) is the selected sender-receiver pair at time t. We can then write $\mathbf{x}(t)^\top = \mathbf{x}(0)^\top \mathbf{P}(t)$, where $\mathbf{P}(t) \equiv \prod_{u \leq t} \mathbf{K}(u)$. Observe that $\forall t, \mathbf{K}(t)^\top \mathbf{1} = \mathbf{1}$ and $\mathbf{P}(t)^\top \mathbf{1} = \mathbf{1}$, meaning that $\mathbf{K}(t)$ and $\mathbf{P}(t)$ are row-stochastic. A row-stochastic matrix that is irreducible is said to be scrambling (see [21]). Establishing irreducibility of $\mathbf{K}$ amounts to check if the network graph is strongly connected, *i.e.*, for any two nodes (i, j) there exists a route from i to j through available connections. Assuming bidirectional communication is allowed between any pair of connected nodes, the only constraint to ensure $\mathbf{K}$ is irreducible is that the network graph is connected, which is a logical assumption in the distributed computing setup we consider. Consequently, $\mathbf{K}$ is a scrambling matrix. Scrambling matrices are shown to be weakly ergodic [22], that is, the

product of t realizations tends to have identical rows as t grows. This can be proved by introducing the coefficient of ergodicity $\tau(\cdot)$ defined by

$$\forall \mathbf{A} \in \mathbb{R}^{N\times N}, \quad \tau(\mathbf{A}) = \frac{1}{2}\max_{i,j}\|\mathbf{A}^T(\mathbf{e}_i - \mathbf{e}_j)\|_1$$

In words, $\tau(\mathbf{A})$ corresponds to the maximum $\mathcal{L}_1$ distance between any two rows of $\mathbf{A}$. $\mathbf{A}$ has equal rows if and only if $\tau(\mathbf{A}) = 0$. When $\mathbf{P}(t)$ is the forward product of t realizations of a scrambling random matrix $\mathbf{K}$, [22] states

$$\tau(\mathbf{P}(t+1)) = \tau(\mathbf{P}(t)\mathbf{K}(t+1)) \leq \tau(\mathbf{P}(t))\lambda_2(t),$$

where $\lambda_2(t)$ is the second largest eigenvalue of $\mathbf{K}(t+1)$. Then, $\tau(\mathbf{P}(t)) \leq \lambda_2^t$, where λ_2 is the maximal second largest eigenvalue over all realizations of $\mathbf{K}$. $\mathbf{K}(t)$ being row-stochastic, we have $0 < \lambda_2 < 1$. Hence,

$$\lim_{t\to\infty} \tau(\mathbf{P}(t)) = 0$$

Recalling that $\mathbf{x}^\top(t) = \mathbf{x}^\top(0)\mathbf{P}(t)$, if $\mathbf{P}(t)$ has equal rows we get

$$\forall i, \quad \frac{x_i(t)}{w_i(t)} = \frac{\sum_j x_j(0)\mathbf{P}_{ji}(t)}{\sum_j w_j(0)\mathbf{P}_{ji}(t)} = \frac{\mathbf{P}_{1,i}(t)\sum_j x_j(0)}{\mathbf{P}_{1,i}(t)\sum_j w_j(0)}$$

Simplifying $\mathbf{P}_{1i}(t)$ yields our final result:

$$\forall i, \quad \lim_{t\to\infty} \frac{x_i(t)}{w_i(t)} = \frac{\sum_j x_j(0)}{\sum_j w_j(0)}$$

□