

# Principal Component Analysis for Dimension Reduction in Massive Distributed Data Sets\*

Yongming Qu<sup>†</sup>    George Ostrouchov<sup>‡</sup>    Nagiza Samatova<sup>‡</sup>    Al Geist<sup>‡</sup>

## Abstract

We describe a new method for computing a global principal component analysis (PCA) for the purpose of dimension reduction in data distributed across several locations. We assume that a virtual  $n \times p$  (items  $\times$  features) data matrix is distributed by blocks of rows (items), where  $n > p$  and the distribution among  $s$  locations is determined by a given application. Our approach is to perform local PCA on local data without any data movement and then move and merge the local PCA results into a global PCA. The representation of local data by a few local principal components greatly reduces data transfers with minimal degradation in accuracy. We exploit the fact that most high-dimensional data have lower intrinsic dimensionality thus allowing a good lower-dimensional representation. Existing methods that bring data to a central location require  $O(np)$  data transfer even if only a few principal components are needed. In the worst case, when an exact PCA is computed, our algorithm is  $\min(O(np), O(sp^2))$ . It is of  $O(sp)$  data transfer complexity when intrinsic dimensionality is low or when an approximate solution is sufficient. The ability to vary data transfers by controlling precision provides a great deal of flexibility.

## 1 Introduction

Dimension reduction is a necessary step in the effective analysis of massive high-dimensional data sets. It may be the main objective in the analysis for visualization of the high-dimensional data or it may be an intermediate step that enables some other analysis such as clustering. Principal component analysis (PCA) was first introduced by Pearson [8] in 1901 and later independently developed by Hotelling [3] in 1933, where the name principal components first appears. It is also known as the Karhunen-Loeve procedure, eigenvector analysis, and empirical orthogonal functions. PCA is probably the oldest and certainly the most popular technique for computing lower-dimensional representations of multivariate data. The technique is linear in the sense that the components are linear combinations of the original variables (features), but non-linearity in the data is preserved for effective visualization. The technique can be presented as an iterative computation of the direction of highest variation followed by projection onto the perpendicular hyperplane. This quickly provides a few perpendicular directions that account for the majority of the variation in the data, giving a

---

\*This work has been supported by the MICS Division of the U.S. Department of Energy

<sup>†</sup>Iowa State University, work done in part while on a Summer 2001 Graduate Internship at the Oak Ridge National Laboratory

<sup>‡</sup>Oak Ridge National Laboratory (Managed by UT-Battelle LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.)

low dimensional representation of the data. A complete set of principal components can be viewed as a rotation in the original variable space. See, for example, [5] for a comprehensive treatment and history of principal component analysis. In this paper, we concentrate on the geometric and dimension reduction properties of PCA as applied to the data and we do not use any distributional assumptions about the data.

As Cluster Computing and the Grid are becoming the paradigms of current and future high-performance computing, massive petascale data sets distributed over a network of clusters or a Data Grid are the future in science and, particularly, simulation science. Even present massive data sets stored in multiple files, multiple disks, or multiple tapes are often too large for centralized processing. Some examples are medical records of distributed groups of individuals, sales records of distributed stores on the same group of products, climate simulations based on different initial and boundary conditions, genome characteristics for different organisms, etc. More applications are becoming available as data standards are developed across distributed locations. Wegman [11, 12] discusses the interaction of huge data sets and the limits of computational feasibility, concluding that most current data analysis techniques break down on data sets beyond about 10 megabytes. This is also true of principal component analysis.

It is natural to consider parallel and out-of-core methods for the eigenvalue or singular value decomposition of massive matrices because this is the underlying computation. A recent survey of parallel methods for spectral decomposition of a nonsymmetric matrix on distributed memory processors [1] reports communication costs of roughly  $O(p^2)$  per processor across several algorithms. Further, the parallel methods are tightly coupled, requiring a level of synchronization and a reasonably even data distribution between the local computations. Out-of-core methods by their nature transfer the entire matrix because all matrix rows need to be considered. An efficient out-of-core SVD algorithm was recently reported in [9]. A distributed algorithm for data distributed by blocks of columns (variables) is reported in the context of clustering in [6]. We present an algorithm for the complementary case of data distributed by blocks of rows.

Because our goal is dimension reduction, a form of approximation by a lower-dimensional object, we can begin with local dimension reduction. In contrast to the “exact” computation performed by the parallel and out-of-core methods, we can distribute approximation itself and trade a small amount of approximation for a large amount of data transfer and synchronization. An approximation approach is also applied in [6] for data distributed by blocks of columns. The fact that most high-dimensional data is of much lower intrinsic dimensionality [10] is the concept that makes such an approach practical. The remainder of this paper is organized as follows. We begin in Section 2 with some notation for PCA. Section 3 presents the concepts needed for merging local PCA into a global PCA. Section 4 presents the formal algorithm. Performance on synthetic data is discussed in Section 5.

## 2 Notation and Background for Principal Components Analysis

Let  $\mathbf{X}$  be an  $n \times p$  data matrix, whose rows are the observations (items) and columns are the variables (features). For simplicity of exposition and because our algorithm is most effective in this case, we assume that  $n > p$ . The data covariance matrix  $\mathbf{S}$  is given by

$$n\mathbf{S} = \mathbf{X}^T(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X} = \mathbf{X}^T\mathbf{X} - n^{-1}\bar{\mathbf{x}}\bar{\mathbf{x}}^T, \quad (1)$$

where  $\bar{\mathbf{x}} = n^{-1}\mathbf{1}^T\mathbf{X}$  is the vector of  $p$  column means and  $\mathbf{1}$  is a vector of all ones. Note that the column-centering operator  $(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)$  is idempotent. Often  $n - 1$  is used instead of  $n$  in (1) when the data are a sample from some larger population and properties of the population are of interest. The principal component analysis is given by the eigenvalue decomposition [2] of

$$n\mathbf{S} = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^T, \quad (2)$$

where  $\mathbf{U}$  is the matrix of principal component coefficients, and  $\mathbf{\Lambda}^2 = \text{diag}(\lambda_1^2, \lambda_2^2, \dots, \lambda_p^2)$  is the diagonal matrix of eigenvalues ordered from largest to smallest. The same information can be obtained from the related singular value decomposition of the column-centered data matrix,

$$(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^T, \quad (3)$$

without explicitly forming the covariance matrix.

Taking  $\tilde{\mathbf{U}}$  as the first  $k$  columns of  $\mathbf{U}$  corresponding to  $\tilde{\mathbf{\Lambda}}^2 = \text{diag}(\lambda_1^2, \dots, \lambda_k^2)$ , the largest  $k$  eigenvalues, gives

$$\mathbf{Z} = (\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X}\tilde{\mathbf{U}}^T \quad (4)$$

as the best  $k$ -dimensional representation of the original column-centered  $p$ -dimensional data in the linear least-squares sense [5].

The quality of this approximation is given by the ratio

$$\alpha = \frac{\sum_{i=1}^k \lambda_i^2}{\sum_{i=1}^p \lambda_i^2}, \quad (5)$$

which is the proportion of total variation in the original  $p$  variables explained by the  $k$  principal components.

Our purpose in this paper is to develop an algorithm for global PCA of a data matrix that is distributed among  $s$  locations as

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_s \end{bmatrix}, \quad (6)$$

where  $\mathbf{X}_i$  are  $n_i \times p$  matrices with  $n = \sum_{i=1}^s n_i$ . The allocation of rows to locations depends on the application and need not be balanced.

### 3 Merging Distributed PCA

The crossproduct of the data matrix can be expressed as a sum of row outer products. In [7], within a parallel computing context, we note that a row partition of the data matrix (6) induces a partition on this sum of row outer products. In a similar fashion, here we first show how the  $\mathbf{X}(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X}$  matrix can be partitioned into a sum of the local  $\mathbf{X}_i^T(\mathbf{I} - n_i^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X}_i$  matrices and a term involving the local column means.

Note that the idempotent centering operator in (1) that we shall denote by  $\mathbf{C}$  can be written as a sum of two idempotent centering operators

$$\begin{aligned}
\mathbf{C} &= [\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T] \\
&= \left[ I - \begin{pmatrix} n_1^{-1}\mathbf{1}\mathbf{1}^T & 0 & \cdots & 0 \\ 0 & n_2^{-1}\mathbf{1}\mathbf{1}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n_s^{-1}\mathbf{1}\mathbf{1}^T \end{pmatrix} \right] + \\
&\quad \left[ \begin{pmatrix} n_1^{-1}\mathbf{1}\mathbf{1}^T & 0 & \cdots & 0 \\ 0 & n_2^{-1}\mathbf{1}\mathbf{1}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n_s^{-1}\mathbf{1}\mathbf{1}^T \end{pmatrix} - n^{-1}\mathbf{1}\mathbf{1}^T \right] \\
&= \mathbf{C}_w + \mathbf{C}_b
\end{aligned} \tag{7}$$

The first operator computes differences from local column means (within locations) and the second operator computes local column mean differences from global column means (between locations). Also note that  $\mathbf{C}_w\mathbf{C}_b = \mathbf{0}$ , so that the centering operators are mutually orthogonal. In terms of the covariance matrix we have

$$\begin{aligned}
n\mathbf{S} &= \mathbf{X}\mathbf{C}\mathbf{X} \\
&= \mathbf{X}\mathbf{C}_w\mathbf{X} + \mathbf{X}\mathbf{C}_b\mathbf{X} \\
&= \sum_{i=1}^s \mathbf{X}_i^T (\mathbf{I} - n_i^{-1}\mathbf{1}\mathbf{1}^T) \mathbf{X}_i + \mathbf{X}\mathbf{C}_b\mathbf{X} \\
&= \sum_{i=1}^s n_i \mathbf{S}_i + \mathbf{X}\mathbf{C}_b\mathbf{X}
\end{aligned} \tag{8}$$

So the global covariance is partitioned into a weighted sum of local covariance and a "between" locations covariance computed from the local means.

Using the partition (8) we can insert the local PCA

$$n\mathbf{S} = \sum_{i=1}^s \mathbf{U}_i \mathbf{\Lambda}_i^2 \mathbf{U}_i^T + \sum_{i=1}^s n_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T. \tag{9}$$

In order to reduce the amount of data transmitted to a central location, we can use only  $k_i$  local eigenvectors corresponding to the largest  $k_i$  eigenvalues that achieve a given quality of approximation as in (4). Letting  $R_i$  be the required proportion of variation at location  $i$

$$n\tilde{\mathbf{S}} = \sum_{i=1}^s \tilde{\mathbf{U}}_i \tilde{\mathbf{\Lambda}}_i^2 \tilde{\mathbf{U}}_i^T + \sum_{i=1}^s n_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T \tag{10}$$

where  $\bar{\mathbf{x}}_i$  is a vector of the local column means and each  $\tilde{\mathbf{U}}_i \tilde{\mathbf{\Lambda}}_i^2 \tilde{\mathbf{U}}_i^T$  decomposition is based on  $k_i$  largest local eigenvalues. The  $k_i$  need not be the same at each location and this depends on  $R_i$ . We can either fix  $k_i = k$ , letting  $R_i$  vary between locations or we can fix  $R_i = R$  and let  $k_i$  vary

between locations. For a fixed bandwidth of transmission we use the former and for a fixed quality of approximation we use the latter.

Given a reconstructed  $n\tilde{\mathbf{S}}$  at a central location we can proceed with computing the global principal components as we outline in the next section.

Because of linear optimality properties of principal components, this algorithm combines optimal (for a fixed  $k_i$ ) linear approximations of local data covariance. Although local optimality is not sufficient for global optimality, we expect to be close for the following reasons. First, if locations are very homogeneous, global principal components are close to local principal components thus close to optimal results are obtained. If locations are very heterogeneous, the majority of the variation is carried in the second sum of equation (10), which is a measure of location heterogeneity and is computed exactly.

## 4 The Distributed Principal Component Algorithm

In this section, we discuss the algorithm and some computational issues. The algorithm begins by setting two parameters  $\alpha$  and  $k$  to guide the approximation. Either the number of local principal components is at least  $k$  or the minimum proportion of variation explained locally is at least  $\alpha$ . The algorithm satisfies both requirements, so if only one needs to be satisfied, the other is set to zero. The descriptive statistics computed in each location are  $(n_i, \alpha_i, k_i, \bar{\mathbf{x}}_i, \tilde{\mathbf{\Lambda}}_i, \tilde{\mathbf{U}}_i)$ . where

$n_i$  is the number of observations in the  $i^{th}$  location.

$\alpha_i$  is the total variance of the  $i^{th}$  data set.

$k_i$  is the number of principal components selected from the  $i^{th}$  data set.

$\bar{\mathbf{x}}_i$  is the vector of column means of the  $i^{th}$  data set.

$\tilde{\mathbf{\Lambda}}_i$  is a diagonal matrix containing  $k_i$  largest eigenvalues in descending order (only the diagonal entries are stored).

$\tilde{\mathbf{U}}_i$  is a matrix whose columns are the  $k_i$  eigenvectors corresponding to the  $k_i$  eigenvalues in  $\tilde{\mathbf{\Lambda}}_i$ .

### The Algorithm: DPCA

1. Set and send local requirements  $\alpha$  and  $k$  to each location.
2. Compute descriptive statistics  $(n_i, \alpha_i, k_i, \bar{\mathbf{x}}_i, \tilde{\mathbf{\Lambda}}_i, \tilde{\mathbf{U}}_i)$  at each location  $i = 1, \dots, s$ , such that  $\frac{\sum_{j=1}^{k_i} \lambda_j^2}{\alpha_i} \geq \alpha$  and  $k_i \geq k$ .
3. Transmit descriptive statistics to central location when complete.
4. Compute  $n\tilde{\mathbf{S}}$  according to equation (10) from the collective descriptive statistics.
5. Compute global principal components  $n\tilde{\mathbf{S}} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ .
6. Set  $\alpha_g$  and  $k_g$  requirements for global principal components.
7. Transmit at least the first  $k_g$  columns of  $\mathbf{U}$  as  $\hat{\mathbf{U}}$  such that  $\frac{\sum_{j=1}^{k_i} d_j}{\sum_{j=1}^p d_j} \geq \alpha_g$  and also  $b\bar{x}$  back to each location.

8. Compute  $\hat{\mathbf{Z}}_i = (\mathbf{X}_i - \mathbf{1}\bar{\mathbf{x}}^T)\hat{\mathbf{U}}$  as the  $k_g$ -dimensional representation of the globally column-centered  $p$ -dimensional data.

Steps (1) and (6) determine the quality of the approximation. The local parameters  $\alpha$  and  $k$  determine the quality of local approximation and  $k_g$  controls the global approximation. For a single parameter control, one may set  $k_g = k$  and  $\alpha = 0$  for a fixed number of principal components. A

variation based single parameter control may be  $\alpha$ ,  $k = 0$ , and  $k_g$  such that  $\frac{\sum_{i=1}^{k_g} d_i}{\sum_{i=1}^p d_i} \geq \alpha$ .

The control parameter choices may depend on expectations about the data distribution. For example if each location describes a very different population and has a distinct mean  $\bar{\mathbf{x}}_i$ , then low quality approximation may be good enough, because a large amount of global variability is in the differences between locations that is carried in the mean vectors. Another reason for low quality approximation may be local data privacy issues. On the other hand, if we expect data to be distributed randomly among locations, then more local principal components may be needed to capture sufficient global variability.

Numerical issues in computing a covariance matrix can arise when rows of the data matrix have widely different scales. The distributed algorithm is, in fact, better in this sense than a centralized algorithm because groups of rows are centered separately. Although this improvement would be apparent only if we compute an exact global PCA communicating the full local principal component decomposition.

An algorithm that begins by centralizing all the data requires  $O(np)$  data transfer even if only a few principal components are needed. This is simply because all the data must be transferred if no local computation is done. A simple minded alternative might be to compute the local sufficient statistics, which include the local covariance matrices and means, and transfer those to a central location for the PCA computation. This approach requires  $O(sp^2)$  data transfer, which can be larger than  $O(np)$  if  $n_i < p$  in a significant proportion of the sites. Our algorithm requires  $O(p \sum_{i=1}^s k_i)$  data transfer. Note that  $k_i \leq \min(n_i, p)$ . This is because at most  $\min(p, n_i)$  eigenvalues are non-zero. In the worst case, when an exact PCA is required,  $O(p \sum_{i=1}^s k_i) \leq \min(O(np), O(sp^2))$ . So that our worst case is the better of the centralized approach and the sufficient statistics approach. When a fast approximate solution is required, the  $k_i$  are a small constant and our algorithm requires  $O(sp)$  data transfers. This is also often the case for a precise solution because a low intrinsic dimensionality exists in most high-dimensional data settings [10]. The ability to vary the data transfers by varying the required precision provides a great deal of flexibility.

## 5 Numerical Experiments

Two experiments on synthetic data are conducted to show the effectiveness of the algorithm. In both experiments, we distribute the data randomly among locations giving homogeneous local mean vectors. This is in fact the most unfavorable case because heterogeneous location mean vectors carry more global variability than homogeneous mean vectors. This puts the highest burden on the local principal component approximation in equation (10) because little variability is contained between the local means .

Our implementation of the distributed algorithm is in R [4]. It currently runs on one processor but operates on distributed data sets as needed to assess the accuracy and data transfer performance of the algorithm. We do not report execution times as these are ultimately dependent on processors used in a distributed situation and on the distribution balance of the data.

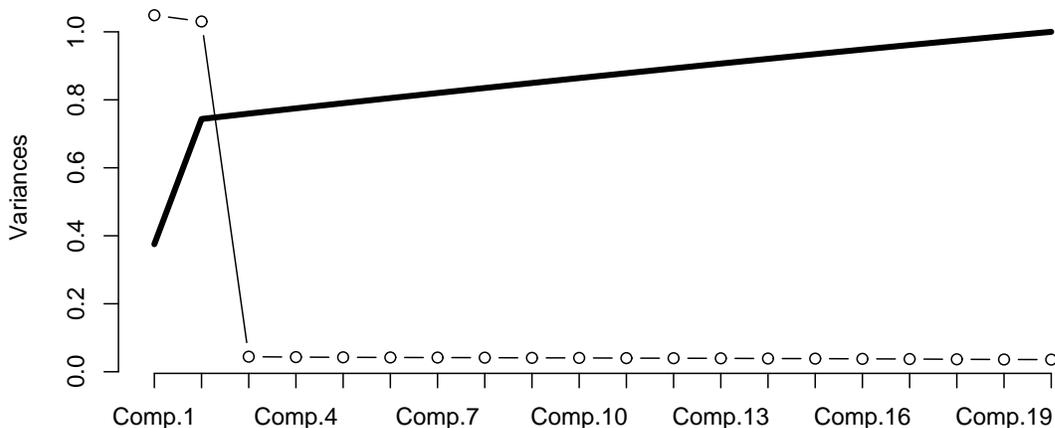


Figure 1: Scree plot showing individual and cumulative component variance.

## 5.1 Generating the synthetic data

The data used in both experiments are generated in the same way: a  $d$ -dimensional Gaussian data set in  $p$ -dimensional space ( $d < p$ ) contaminated with some  $p$ -dimensional Gaussian noise. More formally, the data are generated by the following steps:

1. Generate  $d$ -dimensional data  $\mathbf{A} = \{a_{ij}\}_{n \times d}$  with  $a_{ij} \stackrel{iid}{\sim} N(0,1)$ .
2.  $\mathbf{X} = \mathbf{A}\mathbf{H}^T + \mathbf{E}$ , where  $\mathbf{H}$  is any  $p \times d$  matrix whose columns are orthonormal vectors, and the elements of  $\mathbf{E}$  are identically independently distributed (*iid*) as  $N(0, \sigma^2)$  (normal distribution with mean 0 and variance  $\sigma^2$ ). Here, for simplicity, we let  $\mathbf{H} = \{h_{ij}\}_{p \times d}$ , where

$$h_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

3. The data  $\mathbf{X}$  are divided into  $s$  groups evenly. If  $n$  cannot be divided by  $s$ , let the first  $n \bmod s$  groups contain  $n/s + 1$  observations each, and the remaining groups contain  $n/s$  observations each.

In two experiments that follow, we take  $n = 5,000$ ,  $p = 20$ ,  $d = 2$ , and  $\sigma = 0.2$ . With these parameters, the theoretical covariance matrix has two eigenvalues of 1.04 and eighteen eigenvalues of 0.04. This is confirmed by a scree plot of a realization of this matrix in Figure 5.1. Linear approximations beyond the first two components that represent the intrinsic dimensionality of the data are difficult because the noise is spherical. In Experiment II, in addition to  $\sigma = 0.2$ , we also provide results for increased noise  $\sigma = 0.5$ .

## 5.2 Experiment I

Here, we fix the proportion of variation explained by the distributed principal components (DPC) or the central principal components (CPC) to 0.8. A threshold of variation explained by local PCA

Table 1: Comparison of DPC and CPC for a Fixed Percentage of Variation, 10 Simulations each ( $\alpha = 0.8$ )

	$s$	1	5	10	20	50	100	200	400	500	1000
$k_{ae}$	mean	1.000	1.167	1.181	1.283	1.314	1.333	1.333	1.333	1.333	1.183
	s.d.	.000	.000	.054	.081	.060	.000	.000	.000	.000	.053
$T_{ae}$	mean	.003	.014	.027	.051	.113	.197	.323	.494	.558	.797
	s.d.	.000	.000	.000	.000	.001	.002	.003	.002	.004	.002
$d_a$	mean	.200	.199	.199	.198	.197	.196	.193	.190	.189	.179
	s.d.	.001	.001	.002	.001	.002	.002	.002	.001	.002	.001

is specified as  $\alpha = \sqrt{0.8}$ . Note that 0.8 requires roughly five principal components as can be seen in Fig. 5.1. This extracts principal components from the difficult 20-dimensional spherical noise region beyond the first two components. For this reason, and because of the homogeneous distribution among locations, we consider this experiment to be near the worst case that could be encountered in real data. We run 10 simulations for each value of  $s$ . The means and standard deviations (s.d.) of  $k_{ae}, T_{ae}, d_a, R$  are shown in Table 1, where

1.  $k_{ae}$  is the ratio of number of DPC and that of CPC defined by

$$k_{ae} = \frac{k_a}{k_e} \quad (12)$$

where  $k_a$  is the number of DPC and  $k_e$  is the number of CPC.

2.  $T_{ae}$  is the ratio of the transmission costs defined by

$$T_{ae} = \frac{T_a}{T_e} \quad (13)$$

where  $T_a = (\sum k_i)(p+1) + s(p+3)$  and  $T_e = np$ .

3.  $d_a$  is the relative  $L_2$  distance between the data approximated by DPC and the original data  $\mathbf{X}$  defined by

$$d_a = \frac{\|(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)(\hat{\mathbf{X}} - \mathbf{X})\|_2}{\|(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X}\|_2}, \quad (14)$$

where  $\hat{\mathbf{X}} = \hat{\mathbf{Z}}\hat{\mathbf{U}}$  is the dimension reduced data represented in the original  $p$ -dimensional space.

Note that  $s = 1$  is the CPC case.

In Table 1, the proportion of variation explained by the distributed principal components (DPC) or the central principal components (CPC) is fixed to no less than 0.8. Based on the construction of DPC, the number of DPC is no less than that of CPC, and the ratio  $k_{ae}$  reflects the goodness of the approximation. The smaller  $k_{ae}$  implies better DPC. From the table,  $k_{ae}$  is small when  $s$  is small or large, and it is large for the middle values of  $k$ . This is reasonable. When  $s$  is small, the number of observations in each data set is relatively large and each local covariance matrix is close to the global covariance matrix by the Law of Large Numbers. When  $s$  is large, the information can be well summarized by the local means, which are transmitted exactly. Figure 2 plots  $k_{ae}$  and  $T_{ae}$  versus  $s$ . Note that a maximum 33% increase in the number of components means that we usually require one or two more components.

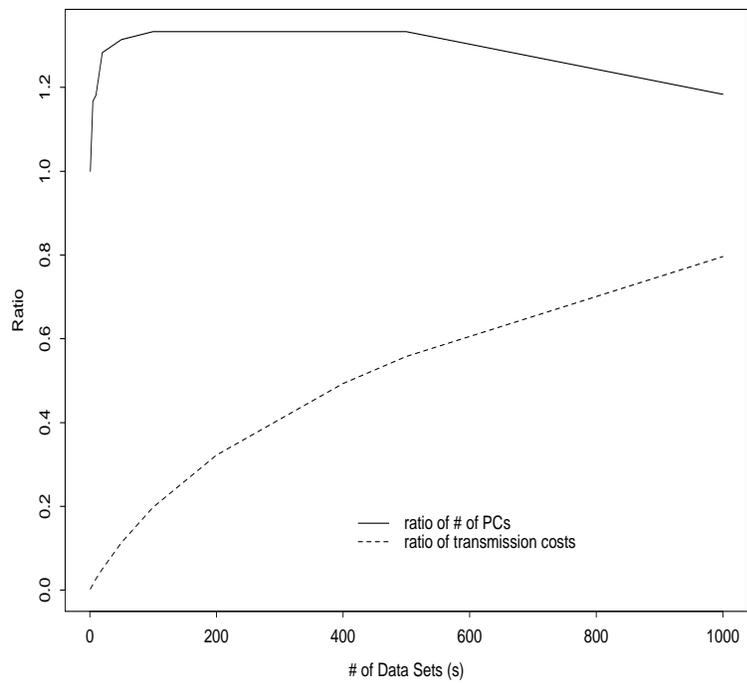


Figure 2: DPC number of principal components relative to CPC and DPC Transmission costs relative to CPC as a function of  $s$  for a fixed proportion of variation explained. ( $\sigma = 0.2$ )

Table 2: Comparison of DPC and CPC for a Fixed Number of Components, 10 Simulations each ( $n = 5,000, p = 20, k = 2, \sigma = 0.2$ )

	$s$	1	5	10	20	50	100	200	400	500	1000
$V_{ae}$	mean	1.000	1.000	1.000	1.000	1.000	1.000	.999	.998	.997	.991
	s.d.	.000	.000	.000	.000	.000	0.000	.000	.000	.000	.001
$T_{ae}$	mean	.002	.007	.014	.027	.059	.107	.189	.325	.390	.662
	s.d.	.000	.000	.000	.000	.001	0.001	.002	.002	.004	.002
$d_a$	mean	.205	.204	.205	.206	.205	.206	.205	.204	.206	.205
	s.d.	.002	.002	.002	.001	.002	.003	.002	.002	.002	.003

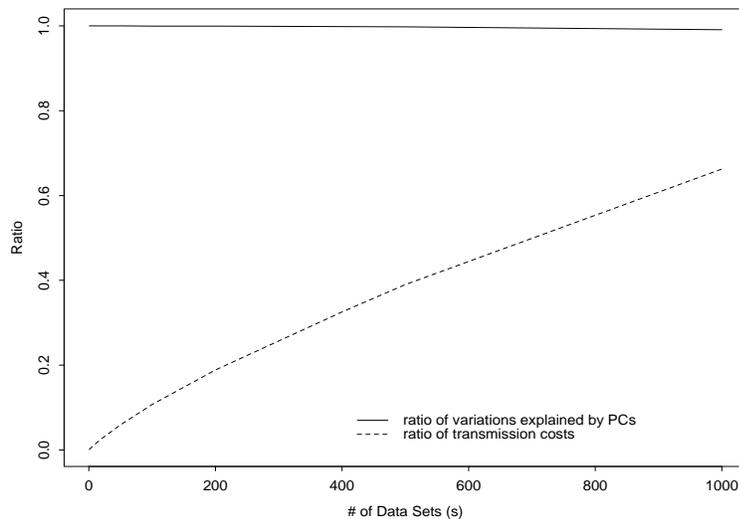


Figure 3: DPC Variation Explained relative to CPC and DPC Transmission Costs relative to CPC as a function of  $s$  for a fixed number of components. ( $\sigma = 0.2$ )

### 5.3 Experiment II

In the second experiment, the number of DPC and CPC components are both fixed at  $k = 2$ . The proportion of variation explained by local PCA is fixed at 0.90. Effectively, we require that the local results are more precise than the global requirement. We run 10 simulations to see the effect of  $s$  on the performance of distributed principal component analysis (DPC). The means and standard errors (s.e.) of  $V_{ae}, T_{ae}, d_a, d_e$  are shown in Table 2, where  $V_{ae}$  is the ratio of the variation explained by the first  $k$  DPC over the variation explained by the first  $k$  CPC, and the rest are defined as in Section 5.2.

From Table 2 and Figure 3,  $V_{ae}$  is uniformly decreasing as  $s$  increases and is close to 1 across all  $s$ . It implies that the DPC is very close to CPC. In order to highlight the difference between DPC and CPC, we increase the noise ( $\sigma^2$ ). The results are shown in Table 3. After comparing Table 2 and 3, we find that DPC losses slightly increase under larger noise. However, the losses are still very small.

Table 3: Comparison of DPC and CPC for a Fixed Number of Components, 10 Simulations each ( $\sigma = 0.5$ )

	$s$	1	5	10	20	50	100	200	400	500	1000
$V_{ae}$	mean	1.000	1.000	0.999	.999	.997	.994	.989	.982	.979	.977
	s.d.	.000	.000	.000	.000	.000	0.000	.001	.000	.000	.001
$T_{ae}$	mean	.003	.016	.031	.059	.131	.230	.372	.550	.614	.837
	s.d.	.000	.000	.000	.000	.001	0.001	.002	.002	.001	.002
$d_a$	mean	.463	.467	.466	.465	.466	.468	.466	.467	.467	.467
	s.d.	.005	.004	.005	.004	.003	.005	.005	.004	.004	.003

## 6 Summary

We present an algorithm for computing approximate principal component analysis of data distributed across several locations. Our algorithm does not require that the local data is transferred to a central location. Only an approximation of each local covariance matrix along with a vector of means is centralized.

The basis of our algorithm is that the global data covariance can be partitioned into a sum of "within" locations covariances and a "between" locations covariance (8). We approximate the "within" local covariances with principal components and compute the "between" covariance exactly (10). Each local approximation carries the linear optimality properties of principal components.

The advantages of our approach compared to out of core and parallel approaches are twofold. First, our algorithm requires substantially lower data transmission rates,  $O(sp)$  compared to  $O(np)$ , where  $n$  is the total number of items across all locations,  $p$  is the number of features, and  $s$  is the number of data locations. Clearly, data sets with many items will benefit the most. Second, because only a representation based on the first two moments of the data is centralized (the vector of means and an approximate covariance matrix), local control remains over local data and data privacy issues are addressed.

Our numerical experiments show that when principal components provide a good data representation, our distributed approximation suffers almost no losses in accuracy. When principal components do not provide a good representation, as in the case of approximating high-dimensional spherical noise, the distributed approximation required up to 33% more components than a centralized algorithm for the same quality of approximation. However, data transfers are reduced in both cases.

## References

- [1] Z. BAI, J. DEMMEL, J. DONGARRA, A. PETITET, H. ROBINSON, AND K. STANLEY, *The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers*, SIAM Journal on Scientific Computing, 18 (1997), pp. 1446–1461.
- [2] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, John’s Hopkins University Press, Baltimore, Maryland, third ed., 1996.
- [3] H. HOTELLING, *Analysis of a complex of statistical variables into principal components*, J. Educ. Psych., 24 (1933), pp. 417–441, 498–520.
- [4] R. IHAKA AND R. GENTLEMAN, *R: A language for data analysis and graphics*, Journal of Computational and Graphical Statistics, 5 (1996), pp. 299–314.
- [5] I. T. JOLIFFE, *Principal Component Analysis*, Springer-Verlag, 1986.
- [6] H. KARGUPTA, W. HUANG, S. KRISHNAMURTHY, AND E. JOHNSON, *Distributed clustering using collective principal component analysis*, in In Proceedings of the ACM SIGKDD Workshop on Distributed and Parallel Knowledge Discovery in Databases, August 2000, pp. 8–19.
- [7] G. OSTROUCHOV, *Parallel computing on a hypercube: an overview of the architecture and some applications*, in Proceedings of the 19th Symposium on the Interface of Computer Science and Statistics, R. M. Heiberger, ed., American Statistical Association, 1987, pp. 27–32.
- [8] K. PEARSON, *On lines and planes of closest fit to systems of points in space*, Phil. Mag., 2 (1901), pp. 559–572.
- [9] E. RABANI AND S. TOLEDO, *Out-of-core SVD and QR decompositions*, in Proceedings of the 10th SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, Virginia, March 2001.
- [10] D. W. SCOTT, *Multivariate Density Estimation: theory, practice, and visualization*, John Wiley & Sons, Inc., New York, 1992.
- [11] E. J. WEGMAN, *Huge data sets and the frontiers of computational feasibility*, Journal of Computational and Graphical Statistics, 4 (1995), pp. 281–295.
- [12] ———, *Visions: New techniques and technologies in statistics*, Computational Statistics, 15 (2000), pp. 133–144.