# Distributed Clustering Using Collective Principal Component Analysis

Hillol Kargupta
Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
Baltimore, MD 21250
hillol@cs.umbc.edu

Weiyun Huang, Krishnamoorthy Sivakumar, Erik Johnson
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752
{whuang1, siva, erikj}@eecs.wsu.edu

30 July 1999

**Abstract**

This paper considers distributed clustering of high dimensional heterogeneous data using a distributed Principal Component Analysis (PCA) technique called the Collective PCA. It presents the Collective PCA technique that can be used independent of the clustering application. It shows a way to integrate the Collective PCA with a given off-the-shelf clustering algorithm in order to develop a distributed clustering technique. It also presents experimental results using different test data sets including an application for web mining.

## 1 Introduction

Clustering of large data sets is a common practice in data mining applications. The success of a clustering application usually depends critically on the representation of the data. Clustering without proper feature selection and feature construction may not produce desirable data clusters. Moreover, this is important for the scalability of the clustering algorithms. Principal component analysis (PCA) [27, 28] is a popular technique to construct a representation of the data that capture maximally variant dimensions of the data. It computes a representation with a set of basis vectors that are the dominant eigenvectors of the covariance matrix generated by the data. Clustering algorithms equipped with PCA-based representation are quite useful in many applications including knowledge discovery from databases (KDD) [6].

Both PCA and PCA-based clustering algorithms are reasonably well understood when the data sets are centrally located. However, the emergence of network-based computation has offered a new challenge to this traditional practice. This has introduced a new important dimension to both PCA and PCA-based clustering — distributed sources of data. This involves two difficult problems: (1) performing distributed PCA and (2) clustering distributed data using the principal components (PCs) computed during the previous step. To the best of our knowledge, there does not exist any technique that can perform distributed PCA from heterogeneous data sets (with different tables storing data for different features) with limited communication of raw data. This paper presents the Collective PCA (CPCA) that offers one solution to this problem. It also offers a distributed clustering technique that integrates the CPCA algorithm. It documents the performance of the proposed techniques for different data sets including a web-mining application.

Table 1: Homogeneous case: Site A with a table for credit card transaction records.

| Account Number | Amount | Location | Previous record | Unusual transaction |
|---|---|---|---|---|
| 11992346 | -42.84 | Seattle | Poor | Yes |
| 12993339 | 2613.33 | Seattle | Good | No |
| 45633341 | 432.42 | Portland | Okay | No |
| 55564999 | 128.32 | Spokane | Okay | Yes |

Table 2: Homogeneous case: Site B with a table for credit card transaction records.

| Account Number | Amount | Location | Previous record | Unusual transaction |
|---|---|---|---|---|
| 87992364 | 446.32 | Berkeley | Good | No |
| 67845921 | 978.24 | Orinda | Good | Yes |
| 85621341 | 719.42 | Walnut | Okay | No |
| 95345998 | -256.40 | Francisco | Bad | Yes |

Section 2 offers a brief review of PCA techniques and discusses the relevance to existing work on parallel/distributed PCA. Section 3 presents the Collective PCA (CPCA) algorithm and the error analysis of the proposed technique. Section 4 documents the performance of the CPCA algorithm for different test cases. Section 5 presents the distributed clustering technique that integrates the CPCA. Section 6 presents experimental results for the distributed clustering algorithm. Section 7 discusses the future work. Finally Section 8 concludes this paper.

## 2 Background

The field of KDD emerged in the recent past as a result of the dramatic evolution of the technology for information storage, access, and analysis. The ability of various organizations to collect, store, and retrieve huge amounts of data has necessitated the development of algorithms that can extract useful information from these databases. KDD addresses this issue.

Distributed knowledge discovery (DKD) [10, 13, 20, 22, 25, 32, 46, 37, 51, 56] takes KDD to a new platform. It embraces the growing trend of merging computation with communication and explores all facets of the KDD process in the context of the emerging distributed computing environments. DKD accepts the fact that data may be inherently distributed among different loosely coupled sites connected by a network and the sites may have heterogeneous data. It offers techniques to discover new knowledge through distributed data analysis and modeling using minimal communication of data. DKD must deal with different possibilities of data distribution. Different sites may contain data for a common set of features of the problem domain. In case of relational data this would mean a consistent database schema across all the sites. This is the homogeneous case. Tables 1 and 2 illustrate this case using an example from a hypothetical credit card transaction domain.[1] There are two data sites A and B, connected by a network. The KDD-objective in such a domain may be to find patterns of fraudulent transactions. Note that both the tables have the same schema. The underlying distribution of the data may or may not be identical across different data sites.

In the general case the data sites may be *heterogeneous*. In other words, sites may contain tables with different schemata. Different features are observed at different sites. Let us illustrate this case with relational data. Table 3 shows two data-tables at site X. The upper table contains weather-related data and the lower one contains demographic data. Table 4 shows the content of site Y, which contains holiday toy sales data. The objective of the KDD process may be detecting relations between the toy sales, the demographic and weather related features. In the general heterogeneous case the tables may be related through different sets of key indices. For example, Tables 3(upper) and (lower) are related through the key feature *City*; on the other hand Table 3 (lower) and Table 4 are related through key feature *State*.

[1] Please note that the credit card domain may not always have consistent schema. The domain is used just for illustration.

Table 3: Heterogeneous case: Site X with two tables, one for weather and the other for demography.

| City | Temp. | Humidity | Wind Chill |
|---|---|---|---|
| Boise | 20 | 24% | 10 |
| Spokane | 32 | 48% | 12 |
| Seattle | 63 | 88% | 4 |
| Portland | 51 | 86% | 4 |
| Vancouver | 47 | 52% | 6 |

| City | State | Size | Average earning | Proportion of small businesses |
|---|---|---|---|---|
| Boise | ID | Small | Low | 0.041 |
| Spokane | WA | Medium | Medium | 0.022 |
| Seattle | WA | Large | High | 0.014 |
| Portland | OR | Large | High | 0.017 |
| Vancouver | BC | Medium | Medium | 0.031 |

Table 4: Heterogeneous case: Site Y with one table containing holiday toy sales data.

| State | Best Selling Item | Price ($) | Number Items Sold (In thousands) |
|---|---|---|---|
| WA | Snarc Action Figure | 47.99 | 23 |
| ID | Power Toads | 23.50 | 2 |
| BC | Light Saber | 19.99 | 5 |
| OR | Super Squirter | 24.99 | 142 |
| CA | Super Fun Ball | 9.99 | 24 |

When the data sets are large, possibly embedded within some DBMS, downloading the data sets from different sites for constructing a single table may be difficult if not impossible from a logistic point of view. It may also demand large bandwidth for better response time. If the data is sensitive, security is also a major issue in downloading large data sets. Clearly, for large distributed environments, data analysis techniques that require minimal communication of raw data are preferable over techniques that require central collection of data sets. Particularly, the role of DKD becomes even more critical in the emerging *wireless computing* domain where the bandwidth is very limited.

This paper considers the PCA and distributed PCA-based clustering of heterogeneous and distributed data. Earlier efforts on classifier and hierarchical cluster learning from heterogeneous distributed data can be found elsewhere [25, 30, 32, 51]. We will not assume any restriction on the number of data sites. By definition, we will assume that there exists at least one key feature (e.g. the feature "City" in Tables 3 & 4) associated with the tables that can be used to link the information across different tables. This paper considers unsupervised analysis of data. Therefore, we will not require the data to come with any classification label. The following section presents a brief review of PCA.

## 2.1 PCA: A Brief Review

Principal Component Analysis (PCA) is a statistical technique for analyzing multivariate data [47, 27, 28]. It involves linear transformation of a collection of related (statistically correlated) variables into a set of transformed variables — usually referred to as principal components. All the principal components are statistically uncorrelated and individual principal components are ordered with respect to the statistical variance of that component. In the following, we provide a brief derivation of principal components, mainly to establish our notation and conventions. A number of excellent references maybe consulted for a thorough exposition [28].

Consider the random vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$[2] with mean $E[\mathbf{X}] = \mathbf{0}$ (if the data has non-zero mean, we first "center" the data by subtracting the mean) and (a symmetric and positive semi-definite) covariance matrix $Cov[\mathbf{X}] = E[\mathbf{X}'\mathbf{X}] = \Sigma_x$.[3]

The $i^{\text{th}}$ principal component of $\mathbf{X}$ is a linear combination $\mathbf{Y}_i = \mathbf{X}\mathbf{a}_i'$, where $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{in}]$, that is uncorrelated with the previous principal components $(\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_{i-1})$ and has maximum variance. The

[2] We denote our vectors as row vectors.

[3] $A'$ denotes the transpose of vector/matrix $A$.

coefficient vector $\mathbf{a}_i$ is usually constrained to be of unit length. A solution to this optimization problem:

$$\max\{\mathbf{a}_i \Sigma_x \mathbf{a}_i'\}, \quad \text{subject to } \mathbf{a}_i \mathbf{a}_i' = 1, \text{ and}$$

$$\mathbf{a}_i \Sigma_x \mathbf{a}_j' = 0, j = 1, 2, \ldots, i-1,$$

is obtained by choosing $\mathbf{a}_i$ to be a unit eigenvector of $\Sigma_x$ corresponding to the $i^{\text{th}}$ largest eigenvalue $\lambda_i$ of $\Sigma_x$.

The $n$ principal components $\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n]$ may be compactly represented as $\mathbf{Y} = \mathbf{X}A$. Here the columns of matrix $A$ consist of the unit eigenvectors of $\Sigma_x$, arranged in descending order based on the corresponding eigenvalues. In general, we are interested in representing $\mathbf{X}$ by means of a small set of principal components (dimensionality reduction). Let $\hat{\mathbf{Y}} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_k]$ be the first $k$ principal components of $\mathbf{X}$, where $k << n$. These principal components can be used to obtain a reasonable approximation of the original data as follows:

$$\hat{\mathbf{X}} = \hat{\mathbf{Y}} \hat{A}' \tag{1}$$

where $\hat{A}$ is the sub-matrix consisting of the first $k$ columns of $A$. The relative mean-squared error (RMSE) between $\mathbf{X}$ and $\hat{\mathbf{X}}$ can be expressed in terms of the eigenvalues of $\Sigma_x$ as follows:

$$\frac{E[(\mathbf{X} - \hat{\mathbf{X}})(\mathbf{X} - \hat{\mathbf{X}})']}{E[\mathbf{X}\mathbf{X}']} = \frac{\sum_{j=k+1}^{n} \lambda_j}{\sum_{j=1}^{n} \lambda_j}. \tag{2}$$

If the eigenvalues of $\Sigma_x$ are "spread-out" in the sense that $\lambda_{\max}/\lambda_{\min}$ is large, the data $\mathbf{X}$ maybe represented (with a small RMSE) by a small number of principal components.

In practice, the covariance of the random vector $\mathbf{X}$ is not given and has to be estimated from available data. Let $X$ be a $m \times n$ data matrix, where each row of $X$ represents a sample (or observation) of the random vector $\mathbf{X}$. We shall use the sample covariance $\frac{1}{m} X'X$ of the data to obtain the transformation matrix $A$ for PCA [3].[4]

In recent years, the QR algorithm has been the most widely used algorithm for calculating the complete set of eigenvalues of a matrix [52, 18]. Cyclic Jacobi methods are particularly suited for implementation in a parallel computer [52, 18]. The divide-and conquer method of Cuppen is a relatively new method for calculating the complete eigensystem of a symmetric, tridiagonal matrix [52]. The singular value decomposition (SVD) of a real, symmetric, positive semi-definite matrix (the matrix $\Sigma_x$ in our case), is equivalent to the orthogonal decomposition in terms of eigenvalues/eigenvectors [26]. Therefore, algorithms for computing the SVD can also be used for PCA. The power method and its variants are some of the simplest techniques for finding a few of the dominant eigenvalue/eigenvector of $\Sigma_x$ [40, 41]. Because of its ease of implementation, we have adopted this method in our experiments.

## 2.2 PCA and Data Analysis

Principal component analysis has found wide applications in various disciplines like psychology [43, 5], genetics [23], pattern recognition [55], remote sensing [35], and seismic data analysis [24, 29], among others.

PCA is also a popular choice for data mining applications. PCA has been used for detecting linear associative rules [17]. Application of PCA-based techniques for large scale text can be found in [6]. PCA has also found applications in ensemble learning and aggregation of multiple models. Merz and Pazzani [42] have reported a PCA-based technique for combining regression estimates. A maximum-likelihood-based framework for constructing mixture models of PCA is proposed by Tipping and Bishop [50]. The technique developed by Tippin and Bishop develops a collection of PCA models by analyzing different horizontal partitions of the data. The objective is to a generate better quality model that is only locally linear, unlike the single PCA model which is globally linear. Their approach offers a technique to combine such PCA models generated on different partitions of the data. Although this is related to our work in the current paper, there is a major difference. In our case the data sets are heterogeneous; in other words data

[4] The normalization factor $m$ is not crucial to our further discussion. In the sequel, for simplicity, we shall drop that factor.

partitions may not share the same feature set. Another approach for aggregating multiple data partitions into a single hierarchical PCA model is developed by Westerhuis, Kourti, and Macgregor [53]. This technique was primarily developed for process control applications in chemical engineering where data sets are collected periodically. This approach iteratively extracts one dominant eigenvector at a time and assumes central storage of data sets. In a distributed environment with limited bandwidth this algorithm requires heavy communication.

The following section presents the overall algorithm of Collective PCA (CPCA) and error analysis.

# 3 The Collective PCA

This section presents the Collective PCA (CPCA) technique for constructing global PCA model from distributed heterogeneous data with minimal communication overhead. It also presents the error analysis of the proposed technique.

## 3.1 Overall Algorithm

This subsection presents the CPCA algorithm from an abstract algorithmic perspective. For the sake of simplicity we consider only one table per site. However, the technique can be generalized to problems with multiple number of tables at each site.

In a distributed and heterogenous environment, the entire data matrix $X$ is comprised of different smaller tables stored at different sites. In particular, each site has data regarding a particular subset of the $n$ features. Let us assume that the data is distributed among $s$ sites and the entire data matrix $X$ can be partitioned as $X = [X_1, X_2, \ldots, X_s]$, where $X_i$ is a $m \times n_i$ submatrix of $X$ that is available at site $i$. A PCA of this distributed data in a centralized fashion would involve moving data to one central site and calculating the eigenvalues/eigenvectors of the covariance matrix $X'X$ of the global data matrix $X$. The amount of data to be moved is $O(mn)$, where $m$ is the number of data samples (rows of the global data table $X$) and $n$ is the total number of features (columns of the global data table $X$). For typical DKD-applications this amount of data communication is either prohibitive because of the limited bandwidth or impractical because of logistics and/or security related reasons.

Given the distributed nature of the data, it would be advantageous to perform the computations for PCA (to the extent possible) locally, thereby minimizing the amount of data communication and the computation at the central site. In the following, we describe the CPCA approach to this problem.

First we perform a PCA, locally, on the data partition $X_i$ at site $i$. Let $A_i$ be the $n_i \times k_i$ matrix whose columns are the $k_i$ eigenvectors corresponding to the $k_i$ largest eigenvalues of $X_i'X_i$. Matrix $A_k$ is computed based on the data partition $X_i$ and we retain only the first $k_i$ principal components at site $i$. Let

$$Y_i = X_i A_i \tag{3}$$

be the principal components computed at site $i$. The choice of $k_i$ will depend on the eigenvalues of the local covariance $X_i'X_i$ and would be dictated by our error tolerance. It is a tradeoff between dimensionality reduction and accuracy. In all our experiments, we chose an error tolerance (i.e. RMSE, see eq. (2)) of 0.1. The matrix $Y_i$ is representative of the data $X_i$. Since typically the number of rows in large tables ($m$) is very large, we select a subset of $c$ samples (rows) (where $c << m$), with uniform probability. With some abuse of notation, we will denote by $Y_i$ the $c \times k_i$ matrix consisting of the principal components and only $c$ selected samples. The individual $Y_i$ and $A_i$ from each site are then transmitted to a central site (this could be just one of the local sites or a different site facilitating the CPCA process). At the central site, a new $c \times k$ data matrix $Y = [Y_1, Y_2, \ldots, Y_s]$ is formed by putting together the data from the individual sites, where $k = \sum_{i=1}^{s} k_i$. The data communication involved here is $O(ck)$ for the $Y_i$'s and $O(\sum_i n_i k_i)$ for the $A_i$'s (compare with $O(mn)$ for the centralized PCA case, where $c << m$ and $k = \sum_{i=1}^{s} k_i << n$). Typically, the number of samples $c$ that are selected is much larger than the number of features $n_i$ at site $i$. Therefore, the overhead involved in transmitting $A_i$ would be negligible compared to the overall data transmission involved.

At the central site, in principle, we need to reconstruct the original data $X$, from the $Y_i$'s and $A_i$'s (see eq. (1)). We then have to perform a PCA based on this reconstructed data. Recall that $Y$ is a simple linear

transformation of $X$; indeed from eq. (3)

$$Y = XA, \quad \text{where} \quad A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_s \end{bmatrix}$$

is a block diagonal matrix. However, since the PCA is invariant to linear transformations [28], we can work with the $Y$ data instead of $X$. The primary advantage is that the size of $Y$ is much smaller than that of $X$. In other words, we can exploit the dimensionality reduction already achieved at each of the local sites. Let $v_i$, $i = 1, 2, \ldots, p$ be the eigenvectors, corresponding to the $p$ largest eigenvalues of the covariance matrix of $Y$. Then $w_i = Av_i$ are the required eigenvectors of the covariance $X'X$ of the original data $X$.

The overall CPCA algorithm is summarized in the following:

1. Perform local PCA at each site; select dominant eigenvectors and project the data along them.
2. Send a sample of the projected data along with the eigenvectors.
3. Combine the projected data from all the sites.
4. Perform PCA on the global data set and identify the dominant eigenvectors and transform them back to the original space.

The following subsection presents an analysis of the approximation errors involved in our proposed CPCA.

## 3.2 Error Analysis of the proposed CPCA

The Collective PCA technique introduces approximations at two steps. These steps contribute to the overall error in the estimated covariance matrix at the central site, which affects the subsequent computation of eigenvectors and the clustering steps. Both factors are discussed below in some detail.

1. *Selection of only the first few dominant principal components at each of the local sites*: The error contributed by this step can be quantified as follows.

   Consider the random vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ with mean $E[\mathbf{X}] = \mu$ and (a symmetric and positive definite) covariance matrix $Cov[\mathbf{X}] = E[(\mathbf{X} - \mu)'(\mathbf{X} - \mu)] = \Sigma_x$. Let $\mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_k]$ be the first $k$ principal components of $\mathbf{X}$, where $k << n$. These principal components can be used to obtain a reasonable approximation to the original data — $\hat{\mathbf{X}} = \mathbf{Y}\hat{A}'$ — where $\hat{A}$ is the submatrix consisting of the first $k$ columns of $A$. The mean-squared error (MSE) between $\mathbf{X}$ and $\hat{\mathbf{X}}$ can be expressed in terms of the eigenvalues of $\Sigma_x$ as follows: $E[(\mathbf{X} - \hat{\mathbf{X}})(\mathbf{X} - \hat{\mathbf{X}})'] = \sum_{j=k+1}^{n} \lambda_j$. This captures the error introduced by neglecting the "insignificant" eigenvectors. Equivalently, we can consider the projected and reconstructed data at the central site to be a "perturbed" version of the actual data, where the amount of perturbation is exactly quantified by the eigenvalues associated with the principal components that were not transmitted.

2. *Transmitting only a subset of the (projected) observations to the central site*: The error contributed by this step can be quantified statistically. In particular, we assume that all the observations are statistically independent and are drawn from a common underlying probability distribution. It then follows from the law of large numbers and the central limit theorem [3, 4] that the covariance matrix $\hat{\Sigma}_x$ estimated from data is related to the true covariance matrix $\Sigma_x$ by

   $$\text{Var}[\|\hat{\Sigma}_x - \Sigma\|] = O\Big(\frac{1}{m}\Big),$$

   where $m$ is the number of observations transmitted to the central site. In simple words, transmitting only a fraction $\alpha$ of the total number of observations results in an increase by a factor $\alpha$ in error associated with estimating the covariance matrix.[5] If the total number of available observations is large, which is usually the case, this increase in estimation error may not be significant.

[5] We are ignoring the error contributed by the first step here.

Table 5: Size of data set, number of sites, and number of selected local principal components.

| Data | No. of rows | No. of features | No. of sites | Total No. of selected local PCs |
|---|---|---|---|---|
| Bodyfat | 252 | 15 | 2 | 4 |
| Bodyfat | 252 | 15 | 3 | 7 |
| Housing | 506 | 14 | 2 | 4 |
| Move | 6129 | 36 | 2 | 9 |
| Move | 6129 | 36 | 6 | 11 |
| Quest | 60000 | 200 | 2 | 161 |
| Quest | 60000 | 200 | 10 | 164 |

The following section presents experimental results comparing performance of the CPCA and the centralized PCA performed after downloading all the tables to a single site.

# 4 Experiments with the CPCA

The CPCA technique is applied to analyze different data sets. This section presents results of the application of CPCA to three experimental test suites. The first experimental suite is comprised of several publicly available relatively small data sets frequently used in machine learning literature. The suite is used to prove the feasibility of the CPCA. The second suite tests the scalability of CPCA to large data sets. The final experimental suite considers the application of CPCA in a real-life application scenario.

## 4.1 Experiment Suite: I

In this section we apply the CPCA technique to three different data sets, in order to demonstrate the method and compare the results with those of centralized principal component analysis. We show that CPCA achieves highly accurate result compared to the centralized approach and it does so with little communication overhead.

### 4.1.1 Bodyfat data:

This data set[6] is comprised of 252 observations and 15 features. We vertically partitioned the data set into two subsets with one set containing the first seven data columns and the other set containing the rest (eight) of the data columns. Next we construct two data sites with each of them storing only one of the two data sets. The row indices are used as the key linking the rows from different sites.

Following the CPCA approach, we perform PCA in each of these two sites, setting the principal component selection threshold as 0.90 (i.e., an error of at most 0.10). For these data sets only two local principal components are sufficient to satisfy the chosen selection threshold at each of the sites. Then the data sets are projected to these locally computed principal components. The projection for each of these partitions can be viewed as a $252 \times 2$ data matrix. Then the rows are sampled from the projected data. Rows corresponding to the same keys are sampled from each site. The sampled rows and the local principal component for each partition are then sent to a central site. Next global PCA is performed on the aggregated data at the central site in order to produce the global principal components.

Figure 1 (left) shows the variation of the angles (in radians) between the approximation of the first two dominant principal components generated by CPCA and the corresponding ones obtained by a centralized PCA with respect to the number of rows that are sampled from local projections. The results show that for these data sets sampling about 20% of the rows in the projected space is sufficient for highly accurate result. Figure 1 (right) shows the result of CPCA on the same data set when partitioned among three data sites.

[6]http: //lib.stat.cmu.edu/datasets/bodyfat

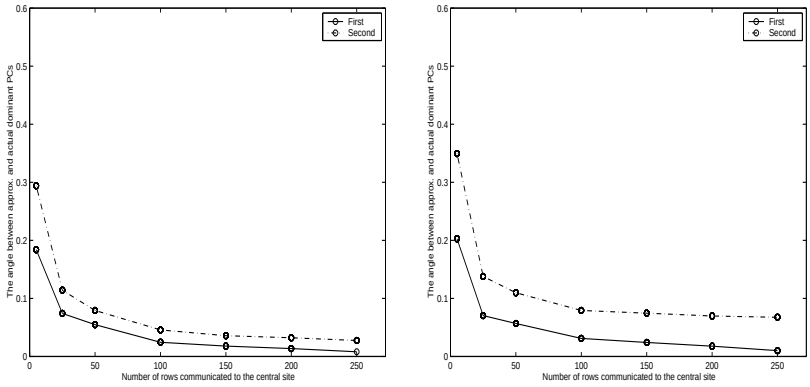


Figure 1: (Left)Performance of CPCA for the *bodyfat data* distributed among two sites.(Right)Performance of CPCA for the *bodyfat data* distributed among three sites. The graphs are averages of five independent runs.

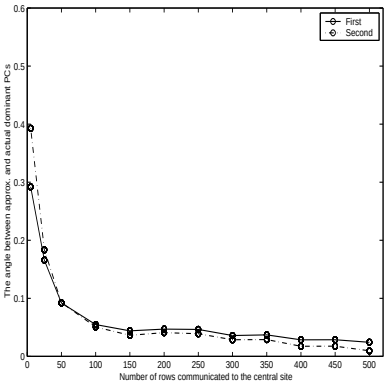


Figure 2: Performance of CPCA for the Housing data distributed among two sites. This is an average of five independent runs.

#### 4.1.2 Boston Housing data and Robot Movement data:

Similar experiments were performed on Boston Housing data[7] (Figure 2) and Robot Move data[8] (Figure 3) (only numeric features are used). The size of the data sets, number of partitions, and the total number of selected local principal components for all the experiments are listed in Table 5.

### 4.2 Experiment Suite: II

This section tests the overall performance including the scalability of the CPCA. We used the Quest Synthetic Data Generation Code [9] to generate a data set with 60,000 observations and 200 features. Although this code generates labeled data, we did not use the class-label for the unsupervised CPCA approach.

We partitioned the data set into two subsets, each containing 100 features. The PC selection threshold is set to 0.90 (i.e., we select the set of PCs that gives a RMSE of 0.1). Next we present the results comparing the two dominant principle components obtained using the centralized approach and the CPCA. Figure 4 (left) shows two curves, each representing the variation of the angle (in radian) between a CPCA-generated dominant principal component and the corresponding principal direction generated from the centralized data. Note that the angle between two unit vectors represents the distance between them and therefore it is a good measure of the accuracy of the estimated PCs.

[7] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/housing/
[8] http: //kdd.ics.uci.edu/databases/pioneer/pioneer.html
[9] http: //www.almaden.ibm.com/cs/quest/syndata.html

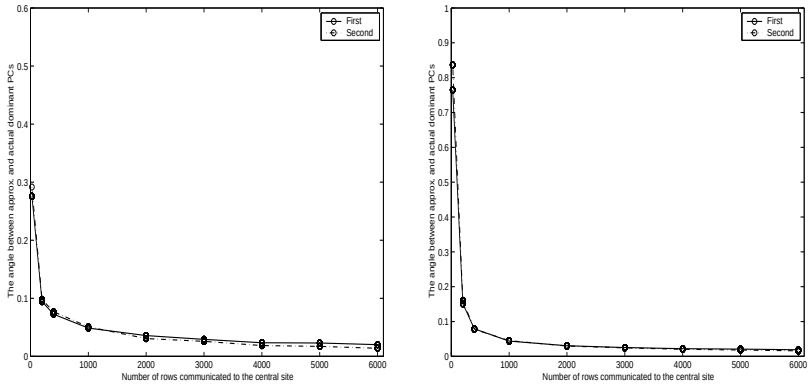


Figure 3: (Left)Performance of CPCA for the robot move data distributed among two sites. (Right) Performance of CPCA for the robot move data distributed among six sites. The graphs are averages of five independent runs.

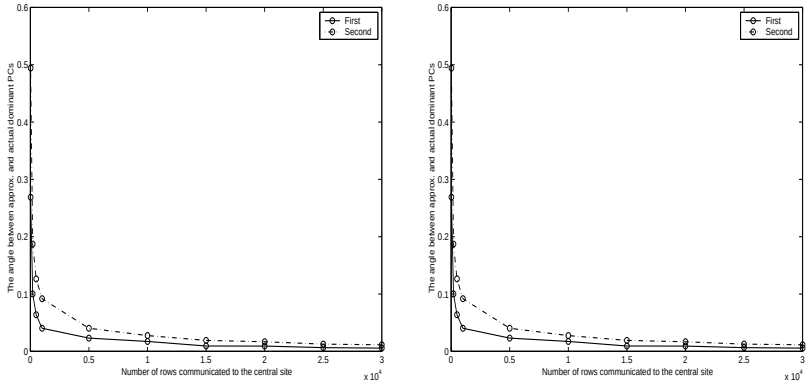


Figure 4: (Left)Performance of CPCA for the Quest Synthetic data with two data sites.(Right)Performance of CPCA for the Quest Synthetic data with ten data sites. The graphs are averages of five independent runs.

Figure 4 (right) shows similar performance when the data set is distributed among ten sites. Table 5 shows the size of the global data set and the total number of selected local principal components from all the sites.

We see from the figures that the CPCA achieves good compression. Suppose our global data set is $m$ by $n$, and is distributed among $s$ sites, such that each site contains $n_i$ features. Number of selected local principal component is $k_i$, and the total number of selected local principal components is $k = \sum_{i=1}^{s} k_i$. Usually when the PC selection threshold is set to 0.90 or 0.95, $k$ is quite small compared to $n$. Let us denote the number of sampled rows by $c$; for our experiments we chose a value of $c$ such that $c < 0.2m$. The total communication cost is $O(ck + \sum_{i=1}^{s} n_i k_i)$, which is very small compared to $O(mn)$, the cost to move the whole data set to one site. The following section describes a distributed clustering algorithm that makes use of the CPCA.

# 5 Distributed Clustering Using the CPCA

Clustering is an important technique that is often used in data mining applications [8, 21, 44, 57]. Clustering high dimensional data often requires application of PCA-like techniques for constructing features that capture the maximum variance in the data in a small number of components. When the data is centralized, application of PCA followed by clustering is a normal practice. However, doing that in a distributed environment with minimal communication of raw data is a challenge. The previous sections presented the CPCA technique for performing PCA from distributed and heterogeneous data. This section demonstrates that the CPCA can be easily integrated with standard off-the-shelve clustering algorithms in order to generate a distributed clustering technique.

There are numerous recent efforts directed towards scaling up clustering algorithms. In [45], the author shows an adaptation of the SLINK [48] and other agglomerative hierarchical clustering algorithms to a multiprocessor environment to parallelize the clustering process. The PADMA system [31] offers a distributed clustering system for homogeneous text data. In [15], the authors adapt the K-Means algorithm to run in a parallel/distributed environment. The Collective Hierarchical Clustering algorithm was proposed elsewhere [30] for generating hierarchical clusters from distributed and heterogeneous data. To the best of our knowledge there does not exist any known technique for PCA-based clustering of distributed, *heterogeneous* data. The following discussion presents a technique to do that.

## 5.1 CPCA-based Distributed Clustering

The proposed distributed clustering approach respects the user's choice of any specific local clustering algorithm and works using a given module of centralized clustering algorithm $\mathcal{C}$. It executes the following steps:

1. Performs local PCA at each site.
2. Projects the local data on the local PCs and applies the given clustering algorithm $\mathcal{C}$ at each site.
3. Selects a set of representative points from each cluster at every site. Let $S_i$ be the set of indices at site $i$ corresponding to the chosen representative points.
4. All sites communicate the projected data rows corresponding to all indices in $\cup_i S_i$ to the central site.
5. The central site performs the global PCA on this collected data set and broadcasts the global PCs to each site.
6. Each site projects the local data on the global PCs and performs clustering using the given algorithm $\mathcal{C}$.
7. Each site communicates a description of the locally constructed clusters (using a graph structure) to the central site.

8. The central site combines the different graphs obtained from the local sites. The combination methods may vary. Here we present one method: since we have already obtained the information about representative points, we can have each site send every point's "nearest neighbor" index to central site, i.e., the central site will know that in each local site, which representative point is the closest to each point. If there are $n$ data points in the global data set, it will take $O(n)$ communications to send the indices information. Then we could use that representative point's local information to approximate the point's local information, so as to approximate the global information and get the global clusters. We will discuss this approximation in detail in next section.

The foundation of this algorithm is discussed in the following section.

## 5.2 Theoretical Foundation

The rationale behind this algorithm is explained in the following. The local PCs are computed for compressing the local data—the same reason that we had for the initial step in the plain vanilla flavored CPCA. In case of CPCA the next step was the uniform sampling of the local data rows. In case of the distributed clustering we take advantage of the given local clustering technique in order to select representatives of all the different clusters. This reduces the possibility of choosing a sample set that completely neglects some clusters present in the data set. The idea of using representative points in clustering is not new. Centroids and other statistical entities have long been used for representing clusters. However, this approach may not work when clusters are of different shapes and sizes. Alternate choices for representative points have also been proposed. For example, the CURE algorithm [21] selects a set of data points for each cluster as representative points, so that clusters of arbitrary shapes can be discovered. The algorithm proposed here shares the latter perspective.

Next the global site computes the global PCs and broadcasts them to each site. This step is needed since we would like to construct the final clusters based on the global PCs. Note that the local PCs are useful only in the local context. They may not have any implication in the global sense.

Step six of the proposed clustering algorithm involves projection of the local data along the global eigenvectors. In steps 7 and 8, we perform global clustering by decomposing it into local cite clusterings and combining them at the central site. In other words, we are claiming that the global distance[10] between any two points $\mathbf{x}$ and $\mathbf{y}$ can be accurately approximated by adding the local distances between these two points. The following discussion justifies this approach for the case of Euclidean distance metric, since the underlying representation is orthogonal.

Let $V$ be an $n \times p$ matrix such that its columns are the global PCs, i.e., the $p$ global eigenvectors corresponding to the $p$ dominant eigenvalues (computed in step 5). The projection of the complete data set $X$ along $V$ is $XV$. Unfortunately, the data set $X$ is distributed among different sites, i.e. $X = [X_1, X_2, \ldots, X_s]$. Note that $X_i$ (the data set at site $i$) has $m$ rows and $n_i$ columns. Moreover, we can write $V = [V_1', V_2', \ldots, V_s']'$, where $V_i$ is an $n_i \times p$ sub-matrix. Therefore, $XV = [X_1V_1 + X_2V_2 \cdots + X_sV_s]$ and the $i$-th site computes only $X_iV_i$.

Let us consider two data points $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s]$ and $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_s]$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are $1 \times n_i$ dimensional row matrices. Computing the distance between $\mathbf{x}$ and $\mathbf{y}$ requires central collection of the $\mathbf{x}_1V_1$, $\mathbf{x}_2V_2$, and $\mathbf{x}_sV_s$. However in the following we show that sum of the distances between $\mathbf{x}_iV_i$ and $\mathbf{y}_iV_i$ for all $i$ is a good approximation of the overall distance between $\mathbf{x}$ and $\mathbf{y}$.

Define $\mathbf{x} - \mathbf{y} = \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_s]$. Let $U = [U_1', U_2', \ldots, U_s']'$, where $U_i$ is an $n_i \times (n-p)$ sub-matrix whose columns correspond to the $(n-p)$ global eigenvectors corresponding to the $(n-p)$ smallest global eigenvalues. Recall that, in practice, these eigenvectors are not computed and correspond to the $(n-p)$ PCs that are ignored. It is well known that the $n \times n$ matrix $[V\ U]$ is orthogonal. In particular, its rows form an orthonormal basis.

For notational simplicity, we will consider a two site case (i.e., $s = 2$). The extension to the general case is straight-forward. Using *all* the global eigenvectors, the projection of $\mathbf{z}$ can be written as

[10] assuming that the clustering algorithm makes use of a distance metric.

$$\begin{bmatrix} \hat{\mathbf{z}} & \hat{\mathbf{w}} \end{bmatrix} = \mathbf{z} \begin{bmatrix} V & U \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \begin{bmatrix} V_1 & U_1 \\ V_2 & U_2 \end{bmatrix}$$
$$= \begin{bmatrix} \hat{\mathbf{z}}_1 & \hat{\mathbf{w}}_1 \end{bmatrix} + \begin{bmatrix} \hat{\mathbf{z}}_2 & \hat{\mathbf{w}}_2 \end{bmatrix}.$$

The vectors $\hat{\mathbf{z}}_1 = \mathbf{z}_1 V_1$ and $\hat{\mathbf{z}}_2 = \mathbf{z}_2 V_2$ are the local PCs computed at sites 1 and 2, respectively, whereas the vectors $\hat{\mathbf{w}}_1 = \mathbf{z}_1 U_1$ and $\hat{\mathbf{w}}_2 = \mathbf{z}_2 U_2$ are the PCs that are ignored. Since the rows of matrix $[V\ U]$ form an orthogonal basis, vectors $[\hat{\mathbf{z}}_1\ \hat{\mathbf{w}}_1]$ and $[\hat{\mathbf{z}}_2\ \hat{\mathbf{w}}_2]$ are orthogonal. Therefore,

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{z}\|^2 &= \left\| \begin{bmatrix} \hat{\mathbf{z}} & \hat{\mathbf{w}} \end{bmatrix} \right\|^2 \\ &= \left\| \begin{bmatrix} \hat{\mathbf{z}}_1 & \hat{\mathbf{w}}_1 \end{bmatrix} \right\|^2 + \left\| \begin{bmatrix} \hat{\mathbf{z}}_2 & \hat{\mathbf{w}}_2 \end{bmatrix} \right\|^2 \\ &= \|\hat{\mathbf{z}}_1\|^2 + \|\hat{\mathbf{w}}_1\|^2 + \|\hat{\mathbf{z}}_2\|^2 + \|\hat{\mathbf{w}}_2\|^2. \end{aligned}$$

It follows that

$$\|\mathbf{z}\|^2 - (\|\hat{\mathbf{z}}_1\|^2 + \|\hat{\mathbf{z}}_2\|^2) = \|\hat{\mathbf{w}}_1\|^2 + \|\hat{\mathbf{w}}_2\|^2,$$

where $\hat{\mathbf{z}}_i = \hat{\mathbf{x}}_i - \hat{\mathbf{y}}_i = \mathbf{x}_i V_i - \mathbf{y}_i V_i$, for $i = 1, 2$.

It is now easy to see that the error in approximating the (squared) distance between data points $\mathbf{x}$ and $\mathbf{y}$ by the sum of the (squared) distances between the appropriate local PCs is relatively small. Indeed, it is directly related to the norm of the principal components that have been ignored, which is usually quite small.

This gives us a simple way to approximate the global distance among a pair of points by computing the local distances and adding them accordingly in order to get the global distance. Each site runs the clustering algorithm $\mathcal{C}$ on $X_i V_i$ and sends the local distance information among the representative points of the clusters to the global site. The global site sums up the pair-wise distances between any two points computed by each site. As noted earlier, this serves as a good approximation to the global pair-wise Euclidean distance between the two points. As a result the generated global clusters offer good approximation of the clusters that can be obtained by the corresponding centralized technique. The following section presents some experimental results documenting the performance of the CPCA-based distributed clustering algorithm.

# 6 Experiments with CPCA-Based Distributed Clustering

This section presents the experimental results with CPCA-based clustering technique for two experimental test suites. In our experiments we use K-means clustering algorithm [16] as the clustering module. We compare the results of centralized clustering with those of the CPCA-based distributed clustering. In the centralized case, we also use PCA to extract the features and perform clustering on the projections. Since the result of K-means clustering algorithm depends heavily on the selection of starting points, we randomly choose a set of starting points and use them for both the centralized and distributed algorithms. Our experiments follow the general steps of the algorithm presented in section 5. However, the implementation of some of the steps depends on the clustering algorithm module, as we explain below.

1. In the third step, when we select the representative points, we make sure that the starting points are selected, so that finally we could use the same set of starting points to combine the local clusters in central site.

2. In the sixth step, each site projects the local data on the global PCs. Since the data are heterogeneous, the local data are actually projected to a horizontal partition of the global PCs. Let's call the projection "partial projection". We perform a "single-iteration K-means" clustering on the partial projections. In other words, each site uses the partial projections of the representative points as centroids and computes the distance between the projected data points and these centroids. We label each point with the index of the closest centroid, i.e. the index of a representative point which has the closest partial projection to it.

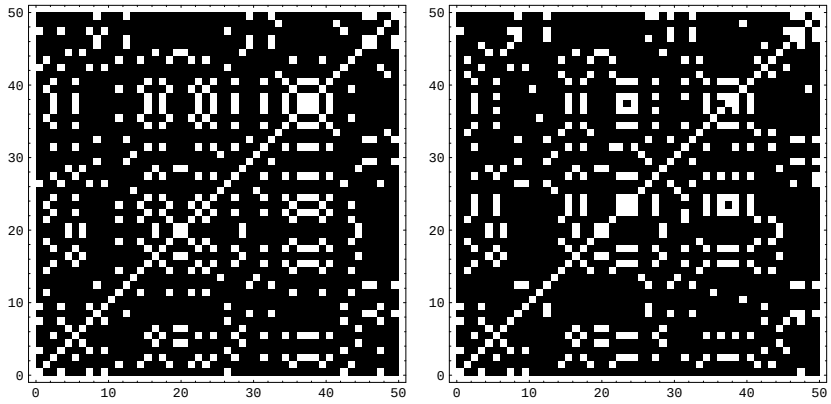


Figure 5: (Left) Performance of centralized clustering for the Quest Synthetic data . (Right) Performance of CPCA-based distributed clustering for the Quest Synthetic data with two data sites. The plots are averages of five independent runs.

3. In the seventh step, we represent the locally constructed clusters with the labels associated with the data points. These labels are communicated to the central site.

4. In the last step, the central site combines the local cluster results. As discussed in section 5.2, we approximate the projections of the global data on the global principal directions by adding the partial projections obtained from local sites. Our previous analytical result shows that this is indeed a good approximation. Then we run the K-means algorithm on the projections of the global data to get the final clusters.

The following sections describe the performance of the above algorithm for two experimental suites.

## 6.1 Experiment Suite: I

This suite uses a data set with 10,000 observations and 100 features, generated by the Quest Synthetic Data Generation Code. As in section 4, we did not use the class labels.

We partitioned the data set into two subsets, each containing 50 features. Threshold is set to 0.90 and the desired number of clusters is set to 8. 10% of points are sampled as representative points. After obtaining the clusters from both centralized and distributed techniques, we randomly sampled 50 points from the data set and constructed an adjacency matrix (if the $i^{th}$ point and the $j^{th}$ point are in the same cluster, then the entry (i,j) of the matrix is set to one, otherwise it is set to zero). Then we draw density plot based on the adjacency matrix. Figure 5 (left) shows the density plot obtained by the centralized algorithm. Figure 5 (right) shows performance of our CPCA-based clustering algorithm. Since we use random sampling in the algorithm, here we present the average of five independent runs as the result of the distributed algorithm. We take the average of the five adjacency matrices obtained by five runs and convert the numeric values in the average matrix into boolean values by rounding off the floating point numbers to the nearest integers.

We also tested the performance of the algorithm with different number of sites. Figure 6 shows the comparison of the sampled results of centralized and distributed algorithm when the data set is equally distributed among ten sites. Figure 7 shows the difference, i.e., misclassifications in the CPCA-based clustering as compared to the centralized PCA-based clustering. The percentage of misclassifications (i.e., the percentage of "white dots" in the density plots) for two-site experiment is 7.44%, while the one for ten-site experiment is 7.60%. We also noticed that with this data set,we didn't get much compression in the CPCA phase (for two-site experiment, we selected 71% of local PCs, and for the ten-site experiment, we selected 83%. This was necessary for the required RMSE of 0.1). We will see that in the next experiment suite, we can get more compression and the accuracy is higher.

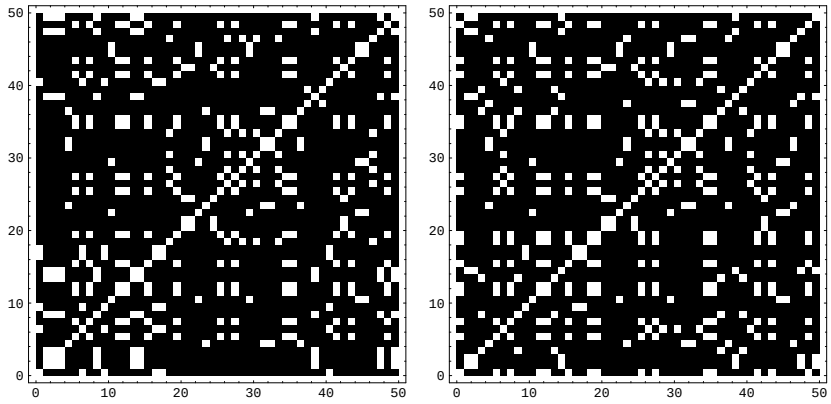


Figure 6: (Left) Performance of centralized clustering for the Quest Synthetic data with ten data sites. (Right) Performance of CPCA-based distributed clustering for the Quest Synthetic data with ten data sites. The plots are averages of five independent runs.

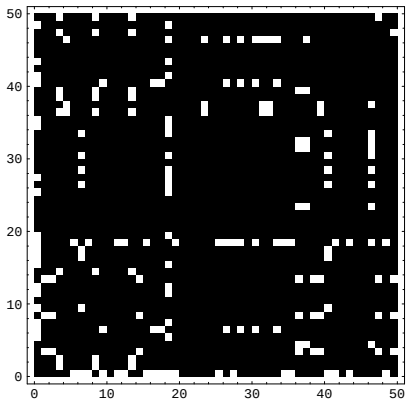


Figure 7: Difference between results of centralized clustering algorithm and CPCA-based distributed clustering algorithm for Quest data with ten data sites.

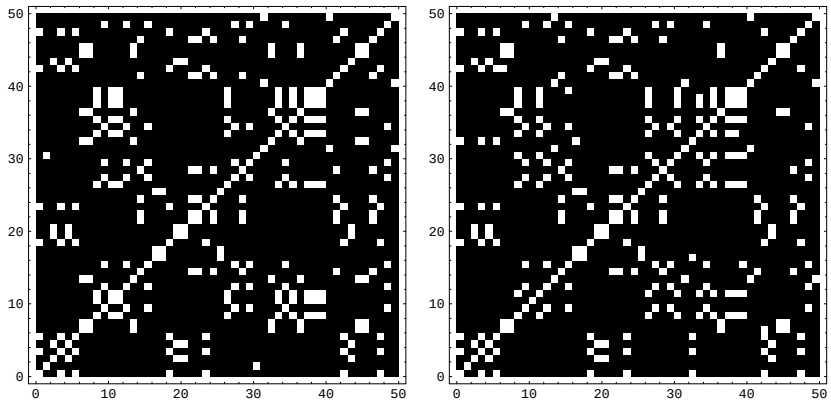


Figure 8: (Left) Performance of Centralized clustering for 3-site web data . (Right) Performance of CPCA-based distributed clustering for 3-site web data when 10% points are sampled as representative points. The plots are averages of five independent runs.

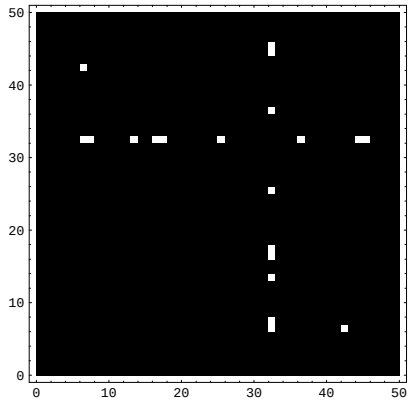


Figure 9: Difference between the results of centralized algorithm and CPCA-based clustering algorithm when 20% points are sampled as representative points .

## 6.2 Experiment Suite: II

The main purpose of this section is to illustrate an application scenario for the CPCA-based clustering technique. PCA is frequently used for high-dimensional text-analysis applications. Therefore text analysis should be an ideal candidate for applying the CPCA. However, in a distributed environment, applications may become more interesting and challenging when relevant data sets involve text, numeric, and other non-numeric features. In the following we describe one such case.

Consider the case of financial news stories regularly posted on the Internet. These news stories are often very useful for investors, portfolio managers, and others. Typically many of these stories are associated with some companies. Announcements regarding new products, quarterly revenue, legal battles, mergers, and partnerships of companies often dominate such business news. On the other hand, there exist many sites in the Internet (for example, Yahoo finance and CNN finance) that offer valuable information about the background and current financial profile about almost all major companies.The website of a company itself also provides quite useful information such as new developments, products, and others. Another important information source will be the stock quotes. We could easily access the real-time stock quotes (or quotes with short delay) online.

The experiment performed in this section considers such real-life applications. We collected two data sets (named as 3-site web data and 4-site web data respectively) from financial websites and did experiments.

First data set (3-site web data) contains three data tables. One table corresponds to news corpora. Each document in this corpora corresponds to a key, and thus associated with a company. For our application we used the abbreviated symbol (ticker symbol) of the corresponding company as the key. A second table is constructed that stores the financial and background feature values of a company, again indexed by the company symbol. A third data table is constructed using the data about the sector the company belongs to. This table is also indexed by the company symbol. We considered a data set involving 1027 companies and the three tables are located at three different sites. The CPCA technique is applied on this three data sets. The PC selection threshold is set to 0.90. Table 6 shows the number of features at each site and corresponding number of chosen PCs. As we see, the CPCA technique offers a big compression factor. We varied the number of representative points to compare the accuracy of our algorithm. Figure 8 (left) shows the density plot for centralized clustering result and figure 8 (right) shows that for distributed clustering result when 10% of points are sampled as representative points. Figure 9 shows the the density plot of the difference between centralized result and distributed result when 20% of points are sampled. When the number of representative points increases, the accuracy is better. The percentages of misclassifications corresponding to 10% and 20% communication of data are 3.28% and 0.8% respectively, according to the 50 points we randomly chosen to display. We selected 24% of local PCs in the CPCA phase.

We also studied the performance with different sample sizes using the 3-site web data. For each sample size, we run our algorithm 5 times and compute the average of the adjacency matrix generated from the clusters. The adjacency matrix is generated for all the data points, in this case it is 1027 by 1027. Then we compare the average adjacency matrix with the one obtained by centralized algorithm and compute the

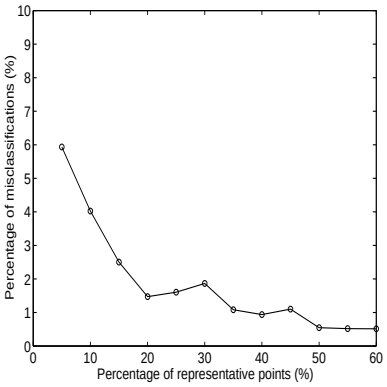


Figure 10: Misclassifications (%) vs. sample size (%): average performance for 3-site web data

Table 6: 3-site web data: data subsets, number of features and number of selected local PCs.

| Data subset | Number of features | Number of selected local PCs |
|---|---|---|
| Profile information | 73 | 2 |
| Sector data | 15 | 4 |
| News stories | 793 | 204 |

percentage of the different entries. Figure 10 shows the percentages of misclassifications when sample size varies from 5% to 60%.

We also tested our algorithm on another data set (called 4-site web data). It involves 4811 companies, and the data are distributed among four sites. Besides the three data sources mentioned above, we collected a fourth data subset, which is the quotes data set. We use the stock quotes on a certain day to simulate the real-time data. The data are time series, i.e., for each company, we record one quote every 15 minutes. For the news stories site, we collected the latest three pieces of news after Oct.1, 2000 for each company. However, some companies may only have one or two news stories in this period. Besides these two subsets, the data set still contains a profile data subset (63 features) and a sector data subset(13 features). The PC selection threshold is set to 0.90, which results in 53.86% of local PCs in the CPCA phase. We sampled 10% of the points as representative points. The percentages of misclassifications among the displayed data is 7.44%. Figure 11 shows the difference between centralized result and distributed result. As in other figures, 50 points are randomly chosen to display the density plots.

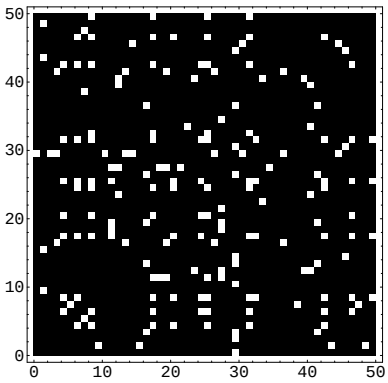


Figure 11: Difference between results of centralized clustering algorithm and CPCA-based distributed clustering algorithm for 4-site web data when 10% points are sampled as representative points.

# 7 Future Work

This paper documents our initial effort to perform distributed PCA-based clustering. It answers few questions. However, it raises even more questions. Several fundamental and applied issues need to be answered before this approach can be adopted in practice. Some of them are discussed below.

The problem of quantifying error in CPCA is one such issue. A natural choice is to compute the angle $\cos^{-1}(u_i' v_i)$ between the $i$th eigenvectors $u_i, v_i$, $i = 1, 2, \ldots, k$, computed by means of CPCA and the global PCA, respectively. This paper adopts this approach (see Figures 1, 2, 3, 4). However, this may not be the best way to do so.

A more appropriate way to quantify the error in CPCA is to compute the "distance" between the subspaces spanned by $\{u_1, \ldots, u_k\}$ and $\{v_1, \ldots, v_k\}$. In fact, for some cases with repeated eigenvalues, the former error maybe large even when the subspaces are close in some appropriate metric. To quote from Stewart [49]:

> "... one cannot expect the eigenvectors of nearby matrices to lie near one another when their corresponding eigenvalues belong to clusters of poorly separated eigenvalues."
>
> "Although the eigenvectors corresponding to a cluster of eigenvalues of a Hermitian matrix are sensitive to perturbations in the elements in the matrix, the subspace spanned by them is relatively insensitive."

Since we ultimately use the computed eigenvectors to project our data onto the subspace spanned by them, the distance between the subspaces is more relevant than the distance between the basis vectors (computed eigenvectors) of an orthonormal basis for these subspaces.

A detailed analysis of this error measure based on distance between subspaces is important. Indeed, let $\mathcal{U}, \mathcal{V} \subset R^n$ be two subspaces. The *gap* between $\mathcal{U}$ and $\mathcal{V}$ is defined as [49, 33]

$$\gamma(\mathcal{U}, \mathcal{V}) = \max\{\sup_{\|u\|=1, u \in \mathcal{U}} \inf_{v \in \mathcal{V}} \|u - v\|, \sup_{\|v\|=1, v \in \mathcal{V}} \inf_{u \in \mathcal{U}} \|u - v\|\},$$

where $\|.\|$ is a norm on $R^n$. The gap function is a metric for the important special case where $\|.\|$ is the Euclidean norm. The gap function has many useful properties:

$$\gamma(\mathcal{U}, \mathcal{V}) < 1 \Rightarrow \dim(\mathcal{U}) = \dim(\mathcal{V}), \quad \text{and} \quad \gamma(\mathcal{U}, \mathcal{V}) = \|P_{\mathcal{U}} - P_{\mathcal{V}}\|$$

where $P_{\mathcal{U}}$, $P_{\mathcal{V}}$ denote orthogonal projection operators onto subspaces $\mathcal{U}$, $\mathcal{V}$, respectively.

The relation between two subspaces $\mathcal{U}$, $\mathcal{V}$ can also be characterized by a set of suitably constructed orthonormal basis vectors for the respective subspaces. The angle $\theta_i$ between the $i$-th basis vectors of $\mathcal{U}$ and $\mathcal{V}$ is defined as the $i$-th canonical angle between the subspaces [14]. In this case, the sine of the largest canonical angle is the gap $\gamma(\mathcal{U}, \mathcal{V})$ between the subspaces.

We are actively investigating the idea of characterizing the CPCA error in terms of the relation between invariant subspaces and will report our results in a future publication.

Another important issue is the reliance of our proposed technique on Euclidean distance metric. While the decomposed evaluation of Euclidean distance works out fine, the same cannot be said for any general non-Euclidean metric. There are several practical domains (e.g. DNA sequences, web-log data) where Euclidean distance may not make sense. One possible way to handle this case is to efficiently construct an embedding of the given data in a non-Euclidean space to an Euclidean space. There exist several interesting results that support this possibility. Appreciating these results require a geometrical perspective of graphs.

Let $(\rho_x)$ be a metric that defines the distance between any two points in the given domain $(\mathcal{X}^n)$ that contains the data sets. An isometry is a mapping $\gamma$ from the metric space $(\mathcal{X}^n, \rho_x)$ to another metric space $(\mathcal{Y}^m, \rho_y)$ such that $\rho_x(x_1, x_2) = \rho_y(\gamma(x_1), \gamma(x_2))$. In other words $\gamma$ preserves the distance between points in the two spaces. We say that the mapping $\gamma$ is $\epsilon$-nearly isometric, if $\frac{\rho_x(x_1, x_2)}{\rho_y(\gamma(x_1), \gamma(x_2))} \leq \epsilon$. In this case we may say that the mapping has an $\epsilon$ distortion.

The following theorem developed elsewhere [7] provide an interesting result about near isometric mappings of a metric space to a Hilbert space.

**Theorem 1 ([7])** *Every $n$-point metric space of dimension $n$ can be mapped to a $O(\log n)$ Hilbert space with an $O(\log n)$ distortion.*

This result was further explored elsewhere [38] which produced the following theorem.

**Theorem 2 ([39])** *In random polynomial time, every $n$-point metric space of $n$ dimensions can be embedded in $\ell_p^{O(\log n)}$ (for any $p \geq 1$), with distortion $O(\log n)$, where $\ell_p^m$ is a norm in the Euclidean space $\Re^m$ defined by $\|(x_1, x_2, \cdots, x_n)\|_p = (\sum |x_i|^p)1/p$.*

The randomized algorithm proposed by Linial et al. [39] works by randomly choosing $O(\log n)$ number of subsets of the data and computing the minimum distance between the point being projected and the subsets. Related work for constructing projections of $n$ points in Euclidean space can be found elsewhere [9, 12]. We are currently exploring this possibility.

The assignment of the representative points to every member of the data set can be made more efficient by storing the data set using similarity preserving indices [54]. We are currently integrating such techniques with the distributed clustering technique.

# 8 Conclusions

Distributed data analysis is playing an increasingly important role in KDD applications from databases connected through large networks. Particularly, the growing popularity of mobile computing devices and wireless networks with limited bandwidth is fostering rapid development of this area. Unsupervised data analysis is an important part of data analysis and PCA plays a critical role in such analysis of high dimensional data. Development of distributed PCA algorithm is therefore important. This paper presented the Collective PCA technique which offers one solution to this problem. The experimental results demonstrated that the CPCA can be effectively used for analyzing high dimensional voluminous data with limited communication overhead.

This paper also showed that the CPCA algorithm can be integrated with off-the-shelf clustering modules for developing distributed clustering algorithms. In fact the integrated approach is likely to improve the performance of the CPCA technique itself. This is because selection of representative samples from data clusters is likely to perform better than the uniform sampling of data. This paper documents the result of our early effort in distributed clustering from heterogeneous data. We do need to pursue the research issues identified in the previous section in order to establish the proposed technique in practice. Nevertheless, it opens up several new possibilities for distributed data mining applications. Distributed web mining, identifying emerging patterns from a large network of sensors, exchange of "significant" activities among mobile devices through wireless networks are only a few scenarios where distributed PCA-based clustering can be useful. We hope that this work brings the field of DKD one more step closer to the ultimate objective — ubiquitous KDD.

# Acknowledgments

This research is supported by the United States National Science Foundation CAREER award IIS-0093353.

# References

[1] J. Aitchison. Multivariate statistical methods. *Biometrika*, 70(1):57–65, 1983.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide.* Society for Industrial & Applied Mathematics, third edition, 1999.

[3] P. J. Bickel and K. A. Doksum. *Mathematical Statistics.* Holden–Day, Oakland, California, 1977.

[4] P. Billingsley. *Probability and Measure.* John Wiley, New York, 1995.

[5] J. E. Birren and D. F. Morrison. Analysis of WAIS subtests in relation to age and education. *Journal of Gerontology*, 16:363–369, 1961.

[6] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, December 1998.

[7] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal Of Mathematics*, 52:46–52, 1985.

[8] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15. AAAI Press, September 1998.

[9] Aggarwal. C., Procopiuc. C., J. Wolf, and P. Yu. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD'99*. ACM, 1999.

[10] P. K. Chan and S. J. Stolfo. Sharing learned models among remote database partitions by local meta-learning. In E. Simoudis, J. Han, and U. Fayyad, editors, *The Second International Conference on Knowledge Discovery and Data Mining*, pages 2–7. AAAI Press, 1996.

[11] W. S. Cleveland and R. Guarino. Some robust statistical procedures and their application to air pollution data. *Technometrics*, 18:401–409, 1976.

[12] L. Cowen and C. Priebe. Randomized non-linear projections uncover high-dimensional structure. *Advances in Applied Math*, 19:319–331, 1997.

[13] V. Crestana and N. Soparkar. Mining decentralized data repositories. Technical Report CSE-TR-385-99, Ann Arbor, MI, 1999.

[14] C. Davis and W. Kahan. The rotation of eigenvectors by a perturbation III. *SIAM Journal of Numerical Analysis*, 7:1–46, 1970.

[15] I. Dhillon and D. Modha. A data clustering algorithm on distributed memory multiprocessors. In *Workshop on Large-Scale Parallel KDD Systems*, 1999.

[16] R. Dubes and A. Jain. Clustering methodologies in exploratory data analysis. *Advances In Computers*, 19:113–228, 1980.

[17] C. Faloutsos, F. Korn, A. Labrinidis, Y. Kotidis, A. Kaplunovich, and D. Perkovic. Quantifiable data mining using principal component analysis. Technical report, 1997. Institute for Systems Research, University of Maryland technical Report TR 97-25.

[18] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989.

[19] A. A. Green, M. Berman, P. Switzer, and M. D. Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Transactions on Geoscience and Remote Sensing*, 26(1):65–74, January 1988.

[20] R. Grossman, S. Bailey, S. Kasif, D. Mon, A. Ramu, and B. Malhi. The preliminary design of PAPYRUS: A system for high performance, distributed data mining over clusters, meta-clusters and super-clusters. Advances in Distributed and Parallel Knowledge Discovery, Eds: Hillol Kargupta and Philip Chan, AAAI/MIT Press, pages 259-276, 2000.

[21] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 73–84. ACM Press, 1998.

[22] Y. Guo and J. Sutiwaraphun. Distributed learning with knowledge probing: A new framework for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, 2000.

[23] S. Hashiguchi and H. Morishima. Estimation of genetic contribution of principal components to individual variates concerned. *Biometrics*, 25:9–15, 1969.

[24] C. H. Hermon and D. Mace. Use of Karhunen-Loeve transformation in seismic data processing. *Geophys. Prosp.*, 26:600–626, 1978.

[25] D. Hershberger and H. Kargupta. Distributed multivariate regression using wavelet-based collective data mining. *Journal of Parallel Distributed Computing*, 61(3):372–400, March 2001.

[26] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

[27] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24, 1933.

[28] J. E. Jackson. *A User's Guide to Principal Components.* John Wiley, 1991.

[29] I. F. Jones and S. Levy. Signal-to-noise ratio enhancement in multichannel seismic data via the Karhunen-Loeve transform. *Geophys. Prosp.*, 35:12–32, 1987.

[30] E. Johnson, H. Kargupta. *Collective, Hierarchical Clustering from Distributed, Heterogeneous Data*, volume 1759. Springer-Verlag, 1999.

[31] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining using an agent based architecture. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of Knowledge Discovery And Data Mining*, pages 211–214, Menlo Park, CA, 1997. AAAI Press.

[32] H. Kargupta, B. Park, D. Hershbereger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. Advances in Distributed and Parallel Knowledge Discovery, Eds: Hillol Kargupta and Philip Chan, AAAI/MIT Press, pages 133-184, 2000.

[33] T. Kato. *Perturbation theory for linear operators.* Springer, New York, 1966.

[34] A. Konig. A survey of methods for multivariate data projection, visualisation and interactive analysis. In *Proceedings of the 5th International Conference on Soft Computing and Information/Intelligent Systems, IIZUKA'98*, pages 55–59, 1998.

[35] J. B. Lee, A. S. Woodyatt, and M. Berman. Enhancement of high spectral resolution remote sending data by a noise-adjusted principal component transform. *IEEE Transactions on Geoscience and Remote Sensing*, 28(3):295–304, May 1990.

[36] J. N. Lee and S. J. Riederer. The contrast-to-noise in relaxation time, synthetic, and weighted-sum MR images. *Magnetic Resonance in Medicine*, 5:13–22, 1987.

[37] W. Lam and A. M. Segre. Distributed data mining of probabilistic knowledge. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 178–185, Washington, 1997. IEEE Computer Society Press.

[38] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings Of The 35th Annual Symposium On Foundations Of Computer Science*, pages 577–591, Los Alamitos, California, USA, 1994. IEEE Computer Society Press.

[39] N. Linial and O. Sasson. Non-expansive hashing. In *Journal Of ACM*, pages 509–518, 1996.

[40] M. L. Maron and R. J. Lopez. *Numerical Analysis: A Practical Approach.* Wadsworth Publishing Company, Belmont, California, 1992.

[41] J. H. Mathews. *Numerical Methods for Mathematics, Science, and Engineering.* Prentice Hall, 1992.

[42] C. Merz and M. Pazzani. A principal components approach to combining regression estimates. *Machine Learning*, 36:9–32, 1999.

[43] D. F. Morrison. *Multivariate statistical methods.* McGraw Hill, New York, 1976.

[44] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of 20th International Conference on Very Large Data Bases*, pages 144–155. Morgan Kaufmann, 1994.

[45] C. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 8:1313–1325, 1995.

[46] H. Park, R. Ayyagari and H. Kargupta, A Fourier Analysis-Based Approach to Learn Classifiers from Distributed Heterogeneous Data. Proceedings of the First SIAM International Conference on Data Mining, 2001.

[47] K. Pearson. On lines and planes of closest fit to systems of points in space. *Phil. Mag., Ser. B*, 2:559–572, 1901.

[48] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16:30–34, 1973.

[49] G. W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15(4):727–764, October 1973.

[50] M. Tipping. Probabilistic visualisation of high-dimensional binary data. To appear in Advances in Neural Information Processing Systems 11. Editors: Michael S. Kearns, Sara A. Solla and David A. Cohn., 1999.

[51] K. Tumer and J. Ghosh. Robust order statistics based ensembles for distributed data mining. To be published in the Advances in Distributed and Parallel Knowledge Discovery, Eds: Hillol Kargupta and Philip Chan, MIT Press, 1999.

[52] D. S. Watkins. *Fundamentals of Matrix Computations.* John Wiley, New York, 1991.

[53] J. Westerhuis, T. Kourti, and J. Macgregor. Analysis of multiblock and hierarchical PCA and PLS models. *Journal of Chemometrics*, 12:301–321, 1998.

[54] D. White and R. Jain. Similarity indexing: Algorithms and performance. In *Proceedings SPIE Vol. 2670, San Diego*, 1996.

[55] S. Wold. Pattern recognition by means of disjoint principal components models. *Pattern Recognition*, 8:127–139, 1976.

[56] K. Yamanishi. Distributed cooperative Bayesian learning strategies. In *Proceedings of COLT 97*, pages 250–262, New York, 1997. ACM.

[57] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114. ACM Press, 1996.

# Author Biographies

insert photo

**Dr. Kargupta** is an Assistant Professor in the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1996. His research interests include ubiquitous and distributed data mining, computation in gene expression, and genetic algorithms. His research is supported by several National Science Foundation grants including a CAREER award and NASA. He won the 1997 Los Alamos Award for Outstanding Technical Achievement. His dissertation earned him the 1996 SIAM (Society for Industrial and Applied Mathematics) annual best student paper prize. He has published more than fifty peer-reviewed articles in journals, conferences, and books. He has also co-edited a book, entitled "Advances in Distributed and Parallel Knowledge Discovery", MIT/AAAI Press.

insert photo

**Weiyun Huang** received the Bachelor of Engineering degree from Xian Jiaotong University, China, in 1996, and Master of Engineering degree from Nanjing University, China, in 1999, both in computer science. She is currently a graduate student in School of EECS, Washington State University. Her research interests include distributed data mining, database systems and distributed systems.

insert photo

**Krishnamoorthy Sivakumar** is an Assistant Professor in the School of EECS, Washington State University. His current research interests include statistical signal processing, Markov models and Bayesian inference for images, and application of signal processing techniques to distributed data mining. He received an M.S. in Mathematical Sciences and M.S. and Ph.D. in Electrical and Computer Engineering, in 1995, 1993, and 1997, respectively, all from The Johns Hopkins University. He has published about thirty technical articles in journals, international conferences, and book chapters, and has one pending patent application. He is a member of the editorial board of the Journal of mathematical imaging and vision and a co-chair of the Workshop on Ubiquitous Data Mining for Mobile and Distributed Environments in the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01).

**Erik Johnson** is a graduate student of the School of Electrical Engineering and Computer Science, Washington State University.