Statistics and Its Interface Volume 11 (2018) 295–306

Dimension reduction for big data

Tonglin Zhang[*] and Baijian Yang

Dimension reduction is aimed at reducing the dimension of a high dimensional vector-valued explanatory variables and simultaneously preserves its relationship with a univariate or low-dimensional real-valued response. As one of the oldest and most well-known dimension reduction approaches, principal component analysis (PCA) has been extensively used in high dimensional data analysis in applications. Classical PCA approaches cannot be applied to big data because of memory and storage barriers. Using a technique called scanning data by rows, the article proposes a new PCA approach. It shows that the proposed PCA approach can provide exact solutions when the size of observed data exceeds the memory size of a computing system.

AMS 2000 subject classifications: Primary 62H25, 00K01; secondary 62J12.
Keywords and phrases: Big data, Dimension reduction, Generalized linear models, Parallel computation, Principal component analysis, Scanning data by rows.

1. INTRODUCTION

Dimension reduction, pioneered by many authors [6, 18, 19, e.g.], is aimed at reducing the dimension of a high dimensional vector-valued explanatory variables and simultaneously preserves its relationship with a univariate or low-dimensional real-valued response. In the past a few decades, statistical approaches in dimension reduction have gained considerable attention due to the rapidly increasing data volume and dimension [7, 8, 17, 31, e.g.]. As sizes of data grow extremely fast in recent years, a new topic, called *big data*, appears. In the era of big data, the usage of traditional dimension reduction approaches is difficult because the size of big data is usually beyond the ability of their basic techniques. The goal of the present article is to develop a new dimension reduction approach to overcoming this difficulty.

The basic idea of our approach is developed based on the technique of scanning data by rows [35]. The technique only needs to load individual rows sequentially from the hard disk to the memory of a computing system. Suppose observations of explanatory variables are expressed by a matrix. If the matrix cannot be loaded to the memory of a single computer, then any dimension reduction approach with the entire matrix in its numerical algorithms cannot be used. To solve the problem, we provide a new dimension reduction approach which does not have the entire matrix in its numerical algorithms. As we attempt to provide exact solutions, we do not adopt the sampling technique in our approach [21]. Instead, we develop our approach based on the technique of scanning data by rows. Using the technique of scanning data by rows, we obtain exact solutions in a dimension reduction problem even if the size of observed data exceeds the memory size of a computing system. We believe that the technique of scanning data by rows will have great impacts on the development of general statistical approaches to big data.

Recently, the term *big data* has spread quickly in both statistics and computer sciences. When dealing with statistical approaches to big data, optimization of target functions via traditional algorithms is often impossible because of the memory and computational efficiency barriers [20, 23]. The memory barrier can be partially solved using an external memory algorithm (EMA) [32]. The size of big data can be much higher than the storage volume of a single computer. The input data must be stored in multiple disks and the computations have to be distributed across many processors such that the whole job can be finished in a reasonable amount of time [23], where the technique of MapReduce is important [9]. MapReduce was originally developed at Google and has been widely accepted for big data analysis. An open source framework made of MapReduce, called *Hadoop*, is popular not only among the academic institutions but also in many areas in industries [11, 24]. From the view of computational perspectives, if an observed data set is successfully saved to the hard disk of a computer, then it is possible to consider methods based on a single processor. Otherwise, parallel computation is recommended. Therefore for big data, algorithms based on a single processor and a cluster of processors are both important.

We apply the technique of scanning data by rows to the dimension reduction problem. As principal component analysis (PCA) is one of the oldest and well-known dimension reduction approaches, we decide to focus on PCA in the dimension reduction problem. The major issue is to provide a way to derive the exact solution to PCA for explanatory variables. We note that PCA is often the first step of data analysis. It may be followed by linear regression [28], generalized linear regression [22], discriminant analysis [5, 36], cluster analysis [12], image analysis [4, 10, 16, 34], functional data analysis [13], and many others [1, 3]. It is therefore important to take a few follow-up analysis approaches into account. Because of the importance of normal, binomial, and Poisson data in applications, we focus on the linear regres-

[*]Corresponding author.

sion approach for the normal data, the logistic regression approach for binomial data, and the loglinear regression approach for Poisson data in the follow-up analysis of our PCA approach.

The remainder of this article is organized as follows. In Section 2, we briefly review the classical PCA approach. In Section 3, we introduce the technique of scanning data by rows. In Section 4, we provide a new PCA approach based on the technique of scanning data by rows. In Section 5, we evaluate the numerical and computational performance of our PCA approach based on three typical follow-up analyses: the linear regression for normal data, the logistic regression for binomial data, and the loglinear regression for Poisson data. In Section 6, we apply our approach to a real example. In Section 7, we provide a discussion.

2. CLASSICAL PCA

Principal Component Analysis (PCA) [15], which was first invented by [26] and later independently developed by [14], is an important data-processing and dimension reduction technique. PCA is a standard and popular statistical approach that tries to explain a large number of highly correlated explanatory variables by a small number of components. Each component is a linear combination of the explanatory variables. Information of explanatory variables is often reflected by a small number of components. Final results of PCA are often easy to interpret and better to understand than results using original explanatory variables.

Let $\mathbf{X}$ be an $n \times p$ matrix of explanatory variables. The first step in classical PCA is to standardize its columns such that their means are zero and their sum of squares of errors are one. If $\mathbf{X}$ is not standardized, then one can compute its standardization $\mathbf{X}_s$ by letting its (i,j)th entry be

$$(1) \qquad x_{ij,s} = \frac{x_{ij} - \bar{x}_j}{b_j}, i = 1, \cdots, n, j = 1, \cdots, p,$$

where x_{ij} is the (i,j)th entry of $\mathbf{X}$, $\bar{x}_j = \sum_{i=1}^n x_{ij}/n$ is the mean and $b_j^2 = \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2$ is the sum of square of errors of the jth column.

Let the singular value decomposition (SVD) of $\mathbf{X}_s$ be

$$(2) \qquad \mathbf{X}_s = \mathbf{U}\mathbf{D}\mathbf{V}',$$

where $\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_p)$ is an $n \times p$ orthogonal matrix satisfying $\mathbf{U}'\mathbf{U} = \mathbf{I}_p$, $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_p)$ is a $p \times p$ orthogonal matrix for loadings satisfying $\mathbf{V}'\mathbf{V} = \mathbf{I}_p$, and $\mathbf{D} = \text{diag}(d_1, \cdots, d_p)$ is a $p \times p$ diagonal matrix for singular values. The singular values are assumed to be ordered such that $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$. If $\mathbf{X}$ is full rank, then $d_p > 0$; otherwise $d_p = 0$. The columns of $\mathbf{Z} = \mathbf{U}\mathbf{D}$ are the principal components (PCs) and the columns of $\mathbf{V}$ are the corresponding loadings. The jth PC is $PC_j = d_j\mathbf{u}_j$ and its sample variance is d_j^2/n. For any integer $k \leq p$, let

$$(3) \qquad \mathbf{X}_{s,k} = \sum_{j=1}^k d_j \mathbf{u}_j \mathbf{v}_j' = \mathbf{U}_k \mathbf{D}_k \mathbf{V}_k',$$

where $\mathbf{U}_k = (\mathbf{u}_1, \cdots, \mathbf{u}_k)$, $\mathbf{D}_k = \text{diag}(d_1, \cdots, d_k)$, and $\mathbf{V}_k = (\mathbf{v}_1, \cdots, \mathbf{v}_k)$. The variation of $\mathbf{X}_{s,k}$ is $\sum_{j=1}^k d_j^2/n$ and its proportion to the total variation is

$$(4) \qquad \lambda_k = \frac{\sum_{j=1}^k d_j^2}{\sum_{j=1}^p d_j^2}.$$

If there is a small k such that $\lambda_k \approx 1$, then the dimension of the data matrix is successfully reduced from p to k with most variations contained in $\mathbf{X}_{s,k}$. It is adequate to use $\mathbf{U}_k\mathbf{D}_k = (PC_1, \cdots, PC_k)$ in the follow up analysis.

We have identified at least three difficulties if we want to apply the classical PCA approach to big data. The first difficulty is the derivation of $\mathbf{X}_s$. Note that $\mathbf{X}_s$ is not available at the beginning of the analysis. It is important to compute $\mathbf{X}_s$ in order to apply the classical PCA. If the data set is small or moderate, then the computation of $\mathbf{X}_s$ is trivial, but this is a major concern if the data set is massive. At least two problems are found. The first problem is the size of $\mathbf{X}$: the size of $\mathbf{X}$ is too large to be loaded to the memory of a personal computer, implying that (1) is hard to be applied. The second problem is the size of $\mathbf{X}_s$: the size of $\mathbf{X}_s$ is almost identical to the size of $\mathbf{X}$, implying that it is hard to save the result. The second difficulty is the application of the SVD to $\mathbf{X}_s$. Suppose $\mathbf{X}_s$ is provided and already saved to the hard disk of a computer. To apply (2), one needs to load $\mathbf{X}_s$ to the memory of the computer, which is a concern if $\mathbf{X}_s$ is massive. The concern is caused by the memory barrier of big data. The third difficulty is caused by the size of $\mathbf{U}$. In (2), the size of $\mathbf{U}$ is identical to the size of $\mathbf{X}_s$, which is also $n \times p$; the size of $\mathbf{D}$ is $p \times p$ but it can be reduced to p; the size of $\mathbf{V}$ is $p \times p$ which can rarely be reduced. If p is moderate or large but n is extremely large, then the size of $\mathbf{U}$ is much higher than the size of $\mathbf{D}$ and $\mathbf{V}$, implying that the result of the SVD in (2) cannot be stayed in memory. In the following of this article, we attempt to provide a new PCA approach which does not have those difficulties.

3. THE TECHNIQUE OF SCANNING DATA BY ROWS

The technique of scanning data by rows is basically different from classical data analysis techniques. In classical data analysis techniques, the first step is always reading the entire data set to the memory of a computing system. After that, a statistical model is considered. Based on the statistical model, a numerical algorithm is carried out in memory for the computation of estimates of model parameters and its variance-covariance matrix. The classical technique is efficient if the data set can be successfully loaded to the memory of the computing system, but it is hard in the analysis of big data. For example, if the data set has $n = 10^7$ rows and $p = 10^3$ columns, then a computer needs around $75GB$ memory size for loading the entire data to memory, which is a major concern in big data analysis. To solve the problem, the technique of scanning data by rows is proposed [35]. This technique only needs to load individual rows to the memory of a computer. After individual rows are loaded sequentially, a set of summary information is provided. The final result of the summary information is derived after the last row is loaded. Based on the final result of summary information, the estimate of model parameters and its variance-covariance matrix are provided. In the following of this section, we attempt to interpret the technique of scanning data by rows based on the linear regression approach.

Assume a data set is composed of a response and many explanatory variables. Let Z_i and $\mathbf{w}_i$ be the ith observed value of the response and the ith observed $(p+1)$-dimensional vector of explanatory variables for $i = 1, \cdots, n$, respectively, where n is the sample size and the first component of $\mathbf{w}_i$ is one, representing the intercept term. If the relationship between the response and explanatory variables is provided by a linear model as

$$Z_i = \mathbf{w}_i'\boldsymbol{\beta} + \epsilon_i, \epsilon_i \sim^{iid} N(0, \sigma^2), \tag{5}$$

for $i = 1, \cdots, n$, then a major interest is to estimate $\boldsymbol{\beta}$. If the maximum likelihood approach is adopted, then the major interest is to derive $\hat{\boldsymbol{\beta}}$, the MLE of $\boldsymbol{\beta}$, and $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}})$, the estimate of the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$.

The technique of scanning data by rows can be partially reflected by the properties of the loglikelihood function of (5) as

$$\begin{aligned}\ell(\boldsymbol{\beta}, \sigma^2) = & -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\left[\sum_{i=1}^n Z_i^2\right. \\ & \left. -2\left(\sum_{i=1}^n \mathbf{w}_i Z_i\right)'\boldsymbol{\beta} + \boldsymbol{\beta}'\left(\sum_{i=1}^n \mathbf{w}_i\mathbf{w}_i'\right)\boldsymbol{\beta}\right].\end{aligned} \tag{6}$$

Note that $\ell(\boldsymbol{\beta}, \sigma^2)$ only involves $s_{zz} = \sum_{i=1}^n Z_i^2$, $\mathbf{s}_{wz} = \sum_{i=1}^n \mathbf{w}_i Z_i$, and $\mathbf{S}_{ww} = \sum_{i=1}^n \mathbf{w}_i\mathbf{w}_i'$, which are a univariate value, a $(p+1)$-dimensional vectors, and a $(p+1)\times(p+1)$-dimensional matrix, respectively. Let

$$\mathcal{S} = (s_{zz}, \mathbf{s}_{wz}, \mathbf{S}_{ww}) \tag{7}$$

be an unstructured array of the sufficient statistics of (5). Then, $\mathcal{S}$ can be computed via the technique of scanning data by rows: if $s_{m,zz} = \sum_{i=1}^m Z_i^2$, $\mathbf{s}_{m,wz} = \sum_{i=1}^m \mathbf{w}_i Z_i$, and $\mathbf{S}_{m,ww} = \sum_{i=1}^m \mathbf{w}_i\mathbf{w}_i'$ are derived after the mth row is scanned, then $s_{m+1,zz} = s_{m,zz} + Z_{m+1}^2$, $\mathbf{s}_{m+1,wz} = \mathbf{s}_{m,wz} + \mathbf{w}_{m+1}Z_{m+1}$, and $\mathbf{S}_{m+1,ww} = \mathbf{S}_{m,ww} + \mathbf{w}_{m+1}\mathbf{w}_{m+1}'$ are their updated values after the $(m+1)$th row is scanned. The final $\mathcal{S}$ is derived after m reaches n. Based on $\mathcal{S}$, we obtain the MLEs $\hat{\boldsymbol{\beta}} = \mathbf{S}_{ww}^{-1}\mathbf{s}_{wz}$ and $\hat{\sigma}^2 = (s_{zz} - \mathbf{s}_{wz}'\mathbf{S}_{ww}^{-1}\mathbf{s}_{wz})/n$ as well as $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2\mathbf{S}_{ww}^{-1}$, the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$.

A great advantage of the technique of scanning data by rows is that it overcomes the memory barrier. It is one of the most important issues to be addressed in big data analysis. In the linear regression approach, the computation of $\mathcal{S}$ only needs $O((p+1)^2)$ memory size, which is irrelevant to n. The technique can be used even if n is extremely large. Since the time of the computation is a linear function of n, it may take long if the size of the observed data is huge. After $\mathcal{S}$ is derived, the rest computations are irrelevant to n, indicating that they are fast. According to the properties of $\mathcal{S}$, it is clear that the technique can be successfully applied to a method based on a single process if p is around a few thousand. For a large p (e.g., greater than a few hundred thousand), a technique using a cluster of processors is needed.

4. BIG DATA PCA

If data are small or moderate, then the whole data set can be loaded to the memory of a single computer and all computations can be carried out in its memory. However, this is not the correct way in computations for big data as we cannot assume that a big data set can always be loaded to the memory of a single computer. Two important scenarios must be addressed. In the first scenario, we assume that the size of big data is not higher than the size of the hard disk of a computer. We focus on methods and algorithms based on a single processor. In the second scenario, we assume that the size of big data is much higher than the size of the hard disk of a single computer, where multiple disks must be used. Then, we focus on methods and algorithms based on a cluster of processors. For big data, methods and algorithms based on a single processor and a cluster of processors are both important.

4.1 Approach based on a single processor

If a data set cannot be loaded to the memory but can be saved to the hard disk of a single computer, then we can consider PCA based on a single processor. We avoid using any standardization procedure of explanatory variables. Therefore, we cannot use $\mathbf{X}_s$ in our approach. We want to derive the value of λ_k for every $k \leq p$. For a given k, we want to provide $\mathbf{U}_k$, $\mathbf{D}_k$, and $\mathbf{V}_k$ for (3) such that a follow-up analysis can be conducted.

Since the classical SVD in (2) cannot be used, we introduce a new approach, called the *modified SVD* approach. The idea of the approach is to use the technique of scanning data by rows. Note that $\mathbf{U}$ is massive but $\mathbf{D}$ and $\mathbf{V}$ are not. We decide to use the technique twice. In the first round of scanning data by rows, we attempt to provide $\mathbf{D}$ and $\mathbf{V}$. After that, we compute λ_k for every $k \leq p$. Then, we provide an optimal k for $\mathbf{D}_k$ and $\mathbf{V}_k$. Using $\mathbf{D}_k$ and $\mathbf{V}_k$ in the second round of scanning data by rows, we provide $\mathbf{U}_k$, which is $\mathbf{U}$ if we purposely choose $k = p$. Since parallel computation may be involved, we decide to introduce the version of the approach based on a single processor in this subsection and the version of the approach based on a parallel computation system in the next subsection.

The modified SVD is developed based on a modified version of the classical SVD in (2). It is expressed as

$$\mathbf{S}_{s,xx} = \mathbf{X}_s'\mathbf{X}_s = \mathbf{V}\mathbf{D}^2\mathbf{V}'. \tag{8}$$

Observing (8), we find that $\mathbf{V}$ is the matrix of eigenvectors and $\mathbf{D}^2$ is the diagonal matrix composed of eigenvalues of $\mathbf{S}_{s,xx}$. Therefore, we can obtain $\mathbf{D}$ and $\mathbf{V}$ if $\mathbf{S}_{s,xx}$ is available. Since the computation of (8) only involves a $p\times p$ matrix, it can be used if p is not very large (e.g., a few thousand). Since $\mathbf{X}_s$ is not available, we cannot directly compute $\mathbf{S}_{s,xx}$ from the data. We propose an indirect way. It is motivated from the technique of scanning data by rows for linear models in Section 3.

Let Y_i be the response variable and $\mathbf{x}_i = (x_{i1},\cdots,x_{ip})'$ be the p-dimensional vector of explanatory variables provided by the ith row of the data. Let $\mathbf{w}_i = (1, x_{i1},\cdots,x_{ip})'$. Then, a linear model for big data is derived if we set $Z_i = Y_i$ in (5). As (5) can only be used for a normally distributed Y_i, the approach introduced in Section 3 cannot be used if Y_i follows other distributions. However, we can still use $\mathbf{S}_{ww} = \sum_{i=1}^n \mathbf{w}_i\mathbf{w}_i'$ in our approach.

We provide a way to compute $\mathbf{S}_{s,xx}$ from $\mathbf{S}_{ww}$. Let $c_{j_1j_2,ww}$ and $c_{j_1j_2,s,xx}$ be the (j_1,j_2)th entries of $\mathbf{S}_{ww}$ and $\mathbf{S}_{s,xx}$, respectively. Since $\mathbf{S}_{ww}$ is a $(p+1)\times(p+1)$ matrix but $\mathbf{S}_{s,xx}$ is a $p\times p$ matrix, we use $j_1,j_2 = 0,1,\cdots,p$ in the definition of $c_{j_1j_2,ww}$ and $j_1,j_2 = 1,\cdots,p$ in the definition of $c_{j_1j_2,s,xx}$. Then, $c_{00,ww} = n$, $c_{0j,ww} = c_{j0,ww} = \sum_{i=1}^n x_{ij}$ for $j = 1,\cdots,p$, and $c_{j_1j_2,ww} = \sum_{i=1}^n x_{ij_1}x_{ij_2}$ for $j_1,j_2 = 1,\cdots,p$, implying that

$$\bar{x}_j = c_{0j,ww}/c_{00,ww} \tag{9}$$

and

$$b_j^2 = c_{jj,ww} - c_{0j,ww}^2/c_{00,ww}. \tag{10}$$

Comparing the relationship between $c_{j_1j_2,ww}$ and $c_{j_1j_2,s,xx}$ for every $j_1,j_2 = 1,\cdots,p$, there is

$$c_{j_1j_2,s,xx} = \frac{c_{j_1j_2,ww} - c_{0j_1,ww}c_{0j_2,ww}/c_{00,ww}}{b_{j_1}b_{j_2}} \tag{11}$$

for $j_1,j_2 = 1,\cdots,p$. Therefore, we can obtain $\mathbf{S}_{s,xx}$ once $\mathbf{S}_{ww}$ is available. Its computational burden is independent of n. Using this idea, we provide an algorithm for $\mathbf{D}^2$ and $\mathbf{V}$ below.

Algorithm 1 has three major computational stages. The first stage is the computation of $\mathbf{S}_{ww}$ given by Step 3. It needs $O((p+1)^2)$ memory size. The time of the computation is $O(n(p+1)^2)$, which is proportional to n. The second stage is the computation of $\mathbf{S}_{s,xx}$ using $\mathbf{S}_{ww}$ given by Step 5, which also needs $O((p+1)^2)$ memory size. The time of the computation is independent of n. The third stage is the computation of $\mathbf{D}^2$ and $\mathbf{V}$ using $\mathbf{S}_{s,xx}$ given by Step 6. It still needs $O((p+1)^2)$ memory size and the time of the computation is irrelevant to n either. Therefore, the most time-consuming stage in Algorithm 1 is the computation of $\mathbf{S}_{ww}$,

Algorithm 1 The First Round Computation Based on A Single Processor

```
   Input: row-by-row of the data from a hard disk
   Output: $\mathbf{S}_{ww}$, $\mathbf{D}$, and $\mathbf{V}$
1: procedure ALGORITHM FOR $\lambda_k$, $\mathbf{D}_k$, AND $\mathbf{V}_k$ FOR EVERY $k \le p$
2:     Let $\mathbf{S}_{ww}$ be a $(p+1)\times(p+1)$ matrix with all entries equal to zero
3:     for the $i$th row of the data do update $\mathbf{S}_{ww} = \mathbf{S}_{ww} + \mathbf{w}_i\mathbf{w}_i'$ until the last row is scanned
4:     end for
5:     Compute $\mathbf{S}_{s,xx}$ using (11)
6:     Compute eigenvalues and eigenvectors of $\mathbf{S}_{s,xx}$ for $\mathbf{D}^2$ and $\mathbf{V}$, respectively
7:     Output
8: end procedure
```

which is the first stage. Once $\mathbf{S}_{ww}$ is available, the rest computations are fast. The entire algorithm needs $O((p+1)^2)$ memory size and $O(n(p+1)^2)$ computational time.

Note that $\mathbf{D}^2 = \mathrm{diag}(d_1^2,\cdots,d_p^2)$ with $d_1 \ge d_2 \ge \cdots d_p \ge 0$. Once $\mathbf{D}^2$ is obtained, we can easily compute λ_k by (4) for every $k \le p$. After that, an optimal k, denoted by k_{opt}, is recommended. It is expected that k_{opt} is small if columns of $\mathbf{X}$ are highly correlated, which means that the dimension of explanatory variables is significantly reduced. As dimension reduction is often the first step of data analysis, it is necessary to provide both the loadings $\mathbf{V}_{k_{opt}}$ and the principal components $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ for the follow-up analysis. Since $\mathbf{V}_{k_{opt}}$ has already been derived in Algorithm 1, we focus on the derivation of $\mathbf{U}_{k_{opt}}$ and $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ when we use the technique of scanning data by rows in the second times.

Note that

$$\begin{aligned}\mathbf{U}_k\mathbf{D}_k &= \mathbf{U}_k\mathbf{D}_k\mathbf{V}_k'\mathbf{V}_k \\ &= \mathbf{X}_{s,k}\mathbf{V}_k \\ &= \left(\sum_{j=1}^k d_j\mathbf{u}_j\mathbf{v}_j'\right)\mathbf{V}_k \\ &= \left(\sum_{j=1}^p d_j\mathbf{u}_j\mathbf{v}_j'\right)\mathbf{V}_k = \mathbf{X}_s\mathbf{V}_k\end{aligned} \tag{12}$$

for any $k \le p$. The lth column of $\mathbf{U}_k\mathbf{D}_k$ is

$$PC_l = \mathbf{X}_s\mathbf{v}_l = \sum_{j=1}^p v_{jl}\mathbf{x}_{j,s}, \tag{13}$$

where $\mathbf{x}_{j,s}$ is the jth column of $\mathbf{X}_s$ and v_{jl} is the (j,l)th entry of $\mathbf{V}$. The ith element of PC_l is

$$pc_{il} = \sum_{l=1}^p v_{jl}x_{ij,s}.$$

Using (1) for $x_{ij,s}$ in above, we obtain

$$pc_{il} = \sum_{j=1}^{p} v_{jl}(x_{ij} - \bar{x}_j)/b_j, \tag{14}$$

implying that the ith element of $\mathbf{u}_l$ is

$$u_{il} = (1/d_l)\sum_{j=1}^{p} v_{jl}(x_{ij} - \bar{x}_j)/b_j. \tag{15}$$

As v_{jl}, $\bar{x}_j$, and b_j are contained in Algorithm 1, we only need to read x_{ij} from the data in the computation of pc_{il}. This is a major issue in the second round usage of the technique of scanning data by rows. Since a response vector $\mathbf{y} = (y_1, \cdots, y_n)'$ may be involved in the follow-up analysis, we also consider $\mathbf{y}$ in the second round usage.

Algorithm 2 The Second Round Computation Based on A Single Processor

```
   Input: row-by-row of the data from a hard disk with $\mathbf{S}_{ww}$, $k_{opt}$, $\mathbf{D}_{k_{opt}}$, and $\mathbf{V}_{k_{opt}}$ from memory
   Output: $PC_1, \cdots, PC_{k_{opt}}$, $\mathbf{u}_1, \cdots, \mathbf{u}_{k_{opt}}$, and $\mathbf{y}$
1: procedure ALGORITHM FOR $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ AND $\mathbf{U}_{k_{opt}}$
2:     for the $i$th row of the data do compute $pc_{i1}, \cdots, pc_{ik_{opt}}$ using (14) and $u_{i1}, \cdots, u_{ik_{opt}}$ using (15)
3:         write $pc_{i1}, \cdots, pc_{ik_{opt}}$, $u_{i1}, \cdots, u_{ik_{opt}}$ and $y_i$ to hard disk
4:         until the last row is scanned
5:     end for
6: end procedure
```

The only task in Algorithm 2 is the computation of principal components. It does not need to load the entire observed data from hard disk to memory using the technique of scanning data by rows. The algorithm needs $O(k_{opt}(p+1))$ memory size and $O(nk_{opt}(p+1))$ computational time, which are lower than the memory size and computational time requested by Algorithm 1. Therefore, we expect that the second round computation is faster than the first round computation. In an extreme case, if one wants to compute all of the p principal components, then the person can purposely choose $k_{opt} = p$ in Algorithm 2 such that the output file contains the response variable $\mathbf{y}$ and all of the principal components, which is $\mathbf{U}$.

4.2 Parallel computation with distributed systems

Due to the reason that data grow extremely fast by daily collection, big data are often measured in tera and petabytes, implying that the whole data cannot be saved to a single hard disk. Therefore, multiple disks are used. To analyze this kind of data, a new concept of cluster computing is becoming popular, in which data-parallel computations are executed on clusters of processors by a distributed system. The pioneer work under this concept is MapReduce [9]. In MapReduce, same computations are applied over a large number of records by many processors [11]. To apply the MapReduce, one must specify the Map and Reduce functions within a job. The job usually divides the input data into independent small data sets that are processed in parallel by the Map tasks. Different outputs of the Map tasks become the inputs of the Reduce task for final results. MapReduce can handle failures of computation tasks by assigning pairs of jobs to different processors. An open source framework made of MapReduce, called *Hadoop*, is popular not only among the academic institutions but also in many areas in industries [11, 24]. Besides MapReduce, a new cluster computing framework called Spark appears [33]. Spark can be deployed in a Hadoop cluster [27]. It is based on the concept of maintaining data in memory rather than in disk. The main abstraction is that Spark introduces an abstraction called resilient distributed datasets (RDDs), which is a read-only collection of objects partitioned across a set of computers that can be rebuilt if the partition is lost.

Suppose a data set is stored in H multiple disks. Let $\mathcal{D}_h$, $h = 1, \cdots, H$, be the portion of the data stored in the hth disk, which has n_h rows. Let $\mathbf{y}_h$ be the n_h-dimensional vector of the response variable and $\mathbf{X}_h$ be the $n_h \times p$-dimensional matrix of explanatory variables in $\mathcal{D}_h$. Then, $\mathbf{y} = (\mathbf{y}_1', \cdots, \mathbf{y}_H')'$ is the n-dimensional vector of the response and $\mathbf{X}$ is the $n \times p$-dimensional matrix of explanatory variables, where $n = \sum_{h=1}^{H} n_h$ is the sample size of the entire data. Using

$$\mathbf{S}_{ww} = \sum_{h=1}^{H} \mathbf{S}_{h,ww}, \tag{16}$$

where $\mathbf{S}_{h,ww} = \mathbf{W}_h'\mathbf{W}_h$ and $\mathbf{W}_h = (\mathbf{1}, \mathbf{X})$, a parallel computation algorithm for $\mathbf{D}$ and $\mathbf{V}$ based on a MapReduce distributed system is proposed.

Algorithm 3 The First Round Computation in A Distributed System

```
   Input: row-by-row of individual $\mathcal{D}_h$
   Output: $\mathbf{S}_{ww}$, $\mathbf{D}$, and $\mathbf{V}$ of the entire data
1: procedure PARALLEL COMPUTATION ALGORITHM FOR $\lambda_k$, $\mathbf{D}_k$, AND $\mathbf{V}_k$ FOR EVERY $k \leq p$
2:     Map tasks: Compute $\mathbf{S}_{h,ww}$ using Step 2 to Step 4 of Algorithm 1 for $h = 1, \cdots, H$ individually
3:     Reduce task: Compute $\mathbf{S}_{ww}$ using (16)
4:     Conduct Step 5 to Step 6 of Algorithm 1 for $\mathbf{D}$ and $\mathbf{V}$
5:     Output
6: end procedure
```

The main task in parallel computation (i.e., Step 2 to Step 3) of Algorithm 3 is the derivation of $\mathbf{S}_{ww}$. Once it is derived, the major task of the parallel computation is over. The computational burden of the rest computations is independent of n, which are possibly carried out in a single processor. A parallel computation algorithm is also needed if p is extremely large (e.g., p is over a million). In this article, we focus on the case when the computation of Step

4 in Algorithm 3 can be carried out in a single processor. Then, a parallel computation version for principal components can be proposed in Algorithm 4, where we denote $\mathbf{U}_{kh}\mathbf{D}_k$ as the portion of $\mathbf{U}_k\mathbf{D}_k$ in $\mathcal{D}_h$ such that we can express $\mathbf{U}_k\mathbf{D}_k = ((\mathbf{U}_{k1}\mathbf{D}_k)', \cdots, (\mathbf{U}_{kH}\mathbf{D}_k)')'$ for the first k principal components of the entire data.

Algorithm 4 The Second Round Computation in A Distributed System

```
   Input: row-by-row of individual $\mathcal{D}_h$ with $\mathbf{S}_{ww}$, $k_{opt}$, $\mathbf{D}_{k_{opt}}$, and $\mathbf{V}_{k_{opt}}$ derived in Algorithm 3
   Output: $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$, $\mathbf{U}_{k_{opt}}$, and $\mathbf{y}$
1: procedure ALGORITHM FOR PRINCIPAL COMPONENTS OF THE ENTIRE DATA
2:     Map tasks: Compute $\mathbf{U}_{k_{opt}h}\mathbf{D}_{k_{opt}}$ using Step 2 to Step 4 of Algorithm 2 for each $\mathcal{D}_h$
3:     Reduce task: Organize the results of map tasks for $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$, $\mathbf{U}_{k_{opt}}$, and $\mathbf{y}$
4: end procedure
```

There are two scenarios in the result of Algorithm 4. In the first scenario, the size of $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ and $\mathbf{y}$ (or $\mathbf{U}_{k_{opt}}$ and $\mathbf{y}$) is not significantly reduced by the PCA approach. A large number of disks must be used to store $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ and $\mathbf{y}$. It is almost equally difficult to use $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ and $\mathbf{y}$ in the follow-up analysis comparing to the method using the original observed data. Then, the PCA approach is not helpful. In the second scenario, the size of $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ and $\mathbf{y}$ is much lower than the size of the entire data. If it is possible to store $\mathbf{U}_{k_{opt}}\mathbf{D}_{k_{opt}}$ and $\mathbf{y}$ in a single or a small number of hard disks, then one can consider to store them in these disks, which makes the follow-up analysis easier than the method using the original observed data. In practice, the two scenarios can be determined by the property of $\mathbf{D}$ in Algorithm 3.

4.3 Relation to classical PCA

Although classical PCA cannot be applied to big data, it is still important to understand the theoretical relation between our big data PCA and the classical PCA approaches. The main issue is to find their difference mathematically. We assume that the result of our big data PCA is provided by the algorithms given in Sections 4.1 and 4.2 but the result of classical PCA is provided by a super computer such that we can assume both results are available in the problem. This is important in understanding the theoretical property of our approach. We show that our approach is theoretically equivalent to the classical PCA approach under a certain equivalence relationship, indicating that we can classify our big data PCA as an exact dimension reduction approach.

We use $\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_p)$, $\mathbf{D} = \text{diag}(d_1, \cdots, d_p)$, and $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_p)$ to denote the result of our big data PCA. We use $\mathbf{U}_c = (\mathbf{u}_{c1}, \cdots, \mathbf{u}_{cp})$, $\mathbf{D}_c = \text{diag}(d_{c1}, \cdots, d_{cp})$, and $\mathbf{V}_c = (\mathbf{v}_{c1}, \cdots, \mathbf{v}_{cp})$ to denote the result of the classical PCA. Clearly, we cannot conclude that the two results are identical as $(\delta_1\mathbf{v}_1, \cdots, \delta_p\mathbf{v}_p)$ for any $\delta_1, \cdots, \delta_p \in \{-1, 1\}$ is also a solution of eigenvector matrix of $\mathbf{S}_{s,xx}$ in (8). Therefore, we provide the following theorems to describe the mathematical relation between our big data PCA and the classical PCA approaches.

Theorem 4.1. $\mathbf{D} = \mathbf{D}_c$.

Theorem 4.2. *If $d_1 > d_2 > \cdots > d_p$ then there exist $\delta_1, \cdots, \delta_p \in \{-1, 1\}$ such that $\mathbf{v}_j = \delta_j\mathbf{v}_{cj}$ and $\mathbf{u}_j = \delta_j\mathbf{u}_{cj}$ for all $j = 1, \cdots, p$.*

Theorem 4.3. *If $d_{\gamma_1-1} > d_{\gamma_1} = d_{\gamma_1+1} = \cdots = d_{\gamma_2-1} > d_{\gamma_2}$ for some $\gamma_1, \gamma_2 \in \{1, \cdots, p\}$ with $\gamma_2 - \gamma_1 \geq 2$ where $d_0 = \infty$ and $d_{p+1} = -\infty$, then there exists an orthogonal matrix $\mathbf{Q}$ on $\mathbb{R}^{\gamma_2-\gamma_1}$ such that $(\mathbf{v}_{\gamma_1}, \cdots, \mathbf{v}_{\gamma_2}) = (\mathbf{v}_{c\gamma_1}, \cdots, \mathbf{v}_{c\gamma_2})\mathbf{Q}$ and $(\mathbf{u}_{\gamma_1}, \cdots, \mathbf{u}_{\gamma_2}) = (\mathbf{u}_{c\gamma_1}, \cdots, \mathbf{u}_{c\gamma_2})\mathbf{Q}$.*

Corollary 4.1. *Assume $d_1, \cdots, d_p$ have q distinct values $d_{01}, \cdots, d_{0q}$ given by $d_{j_1} = d_{j_2}$ if $\gamma_{l-1} \leq j_1, j_2 < \gamma_l$ and $d_{j_1} > d_{j_2}$ otherwise for $l = 1, \cdots, q$, where $1 = \gamma_0 < \gamma_1 < \cdots < \gamma_q = p$ provides a partition of $\{1, \cdots, p\}$. Then there exists a blocked orthogonal matrix $\mathbf{Q} = diag(\mathbf{Q}_1, \cdots, \mathbf{Q}_q)$ such that $\mathbf{V} = \mathbf{V}_c\mathbf{Q}$ and $\mathbf{U} = \mathbf{U}_c\mathbf{Q}$, where $\mathbf{Q}_l$ with $l = 1, \cdots, q$ is an orthogonal matrix on $\mathbb{R}^{\gamma_l-\gamma_{l-1}}$.*

As PCA is often the first step in a data analysis procedure, it is important to understand the impact of the scenarios reflected by Theorem 4.2 and Corollary 4.1 in the follow-up analysis. Suppose a generalized linear model (GLM) is considered in the follow-up analysis. The relation between Y_i and $\mathbf{x}_i$ is modeled by

$$g(\mu_i) = \eta_i = \alpha + \mathbf{x}_i'\boldsymbol{\beta}, i = 1, \cdots, n, \tag{17}$$

where $\mu_i = \mathbb{E}(Y_i)$, g is an link function, α represents the coefficient for the intercept, and $\boldsymbol{\beta}$ represents the vector of coefficients for explanatory variables. If PCA is used, then the relationship is reduced to

$$g(\mu_i) = \eta_i = \alpha + \sum_{j=1}^{k} pc_{ij}\beta_j, i = 1, \cdots, n, \tag{18}$$

where $k \leq p$ and pc_{ij} is the ith element of PC_j, either provided by our big data PCA or classical PCA. Then in (18), $PC_j = \mathbf{X}_s\mathbf{v}_j$ if our big data PCA is used and $PC_j = \mathbf{X}_s\mathbf{v}_{cj}$ if classical PCA is used.

Theorem 4.4. *If $d_k > d_{k+1}$ then the MLEs of $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_n)$ using $PC_j = \mathbf{X}_s\mathbf{v}_j$ or $PC_j = \mathbf{X}_s\mathbf{v}_{cj}$ in (18) are equal.*

The conclusion of Theorem 4.4 does not hold if the condition $d_k > d_{k+1}$ is violated. It can be shown using Corollary 4.1. If $d_k = d_{k+1}$, then there exist γ and γ' satisfying $\gamma \leq k \leq \gamma' - 2$ such that $d_{\gamma-1} > d_\gamma = \cdots = d_{\gamma'-1} > d_{\gamma'}$. The dimension of span$\{\mathbf{v}_\gamma, \cdots, \mathbf{v}_k\}$ is $k-\gamma+1$. The dimension of the eigenvector space corresponding to d_k is $\gamma' - \gamma$, which is greater than $k-\gamma+1$. Therefore, span$\{\mathbf{v}_\gamma, \cdots, \mathbf{v}_k\}$ is a real subspace of the eigenvector space corresponding to

d_k, implying that $\text{span}\{PC_1, \cdots, PC_k\}$ may be differently provided between our big data PCA and classical PCA approaches. However, if all of the eigenvalues are different, then results of the two approaches are identical for every $k \in \{1, \cdots, p\}$.

4.4 Specification in linear regression

Assume the linear regression approach is used in the follow-up analysis. Let the regression model be expressed as

$$\mathbf{y} = \mathbf{1}\alpha + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim N(0, \sigma^2\mathbf{I}), \tag{19}$$

where $\mathbf{1}$ is an n-dimensional vector with all elements equal to one, $\mathbf{y}$ is an n-dimensional response vector, $\boldsymbol{\beta}$ is a p-dimensional vector of regression coefficients for explanatory variables, and $\boldsymbol{\epsilon}$ is an n-dimensional error term. After PCA approach is applied, the principal component matrix $\mathbf{U}_k\mathbf{D}_k$ is derived and Model (19) is approximated by

$$\mathbf{y} = \mathbf{1}\alpha_s + \mathbf{U}_k\mathbf{D}_k\boldsymbol{\beta}_s + \boldsymbol{\epsilon}_s, \boldsymbol{\epsilon}_s \sim N(0, \sigma_s^2\mathbf{I}), \tag{20}$$

where $\boldsymbol{\beta}_s$ is a k-dimensional vector of regression coefficients for principal components. The least square estimator (LSE) of α_s is

$$\hat{\alpha}_s = \bar{y} \tag{21}$$

and its variance is $\mathbb{V}(\hat{\alpha}_s) = \sigma_s^2/n$. The LSE of $\boldsymbol{\beta}_s$ is

$$\hat{\boldsymbol{\beta}}_s = \mathbf{D}_k^{-1}\mathbf{U}_k'\mathbf{y}$$

and its variance-covariance matrix is $\mathbb{V}(\hat{\boldsymbol{\beta}}_s) = \sigma_s^2\mathbf{D}_k^{-2}$. As $\mathbf{X}_s$ is standardized, there is $\mathbb{C}\text{ov}(\hat{\alpha}_s, \hat{\boldsymbol{\beta}}_s) = 0$. The mean square error (MSE) of Model (20) is

$$\hat{\sigma}_s^2 = \mathbf{y}'(\mathbf{I} - \mathbf{1}\mathbf{1}'/n - \mathbf{U}_k\mathbf{U}_k')\mathbf{y}/(n-k-1).$$

Let $s_{yy} = \|\mathbf{y}\|^2$ and $\mathbf{s}_{s,xy} = \mathbf{X}_s'\mathbf{y}$. Then

$$\begin{aligned}\mathbf{V}_k\mathbf{s}_{s,xy} =& \mathbf{V}_k'\mathbf{X}_s'\mathbf{y} \\ =& \mathbf{V}_k'\sum_{j=1}^{p} d_j\mathbf{v}_j\mathbf{u}_j'\mathbf{y} \\ =& \mathbf{V}_k'\sum_{j=1}^{k} d_j\mathbf{v}_j\mathbf{u}_j'\mathbf{y} = \mathbf{D}_k\mathbf{U}_k'\mathbf{y}\end{aligned}$$

and

$$\mathbf{U}_k'\mathbf{y} = \mathbf{D}_k^{-1}\mathbf{V}_k\mathbf{s}_{s,xy}.$$

Thus,

$$\hat{\boldsymbol{\beta}}_s = \mathbf{D}_k^{-2}\mathbf{s}_{s,xy} \tag{22}$$

and

$$\hat{\sigma}_s^2 = (s_{yy} - n\bar{y}^2 - \mathbf{s}_{s,xy}'\mathbf{V}_k'\mathbf{D}_k^{-2}\mathbf{V}_k\mathbf{s}_{s,xy})/(n-k-1), \tag{23}$$

implying that $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}_s^2\mathbf{D}_k^{-2}$.

Theorem 4.5. *It is enough to use $\mathcal{S}$ in the computation of $\hat{\boldsymbol{\beta}}_s$, $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}}_s)$, and $\hat{\sigma}_s^2$.*

Theorem 4.5 concludes that solutions to a regression model in the follow-up analysis can be completely obtained once $\mathcal{S}$ is available, implying that we do not need to scan the data set twice. Therefore, the computation of dimension reduction via PCA in linear regression can be extremely efficient in practice.

5. NUMERICAL EVALUATION

We evaluated the computational advantages of our big data PCA via simulated numerical examples. All of the computations were carried out by a third generation Intel core-i7 2.8GHz processor with 16GB DDR3 memory. All of the algorithms in our big data PCA were written in **C++**. We focused on the evaluation of the performance of our approach based on a single processor. If a parallel algorithm is used, then its performance can be reflected by the performance of algorithms carried out by individual processors. Therefore, the evaluation of algorithms based on a single processor is basic and important in the understanding of our entire approach.

The primary interest was to demonstrate the feasibility of the algorithms provided in Sections 4.1 and 4.2 when they were applied to a massive data set which could not be loaded to memory of a computing system. We assumed that the massive data set was saved to the hard disk of the computing system. It had n observations (i.e., n rows) and many columns. The columns contained information of one response and p explanatory variables. We were interested in the case when both n and p were large. We wanted to show that our big data PCA approach could be used if the size of a data set was a few hundred gigabytes (GB). The data set was generated by **R**, but the PCA analysis was based on **C++**. The goal of the PCA analysis was to provide a file on the hard disk such that the number of columns was significantly reduced.

As dimension reduction is often the first step of data analysis, we considered three approaches in the follow-up analysis: the linear regression approach for normal data, the logistic regression for binomial data, and the loglinear regression for Poisson data. Since response variables are not involved in PCA for explanatory variables, it is possible to carry out the identical computational method for PCA in the three approaches. The follow-up analysis is applied after the PCA analysis is over. As linear, logistic, and loglinear regression models are important and often used in practice, it is important for us to evaluate the impact of our PCA approach based on all of the three models.

5.1 Linear regression

It is common to use the linear regression approach when the response variable is continuous, where one can simply assume the response variable is normally distributed. If the relation between the response and explanatory variables is linear, then the linear regression approach can be applied. In the simulated example for normal data, we assumed that the true relation between $\mathbf{y}$ and $\mathbf{X}$ is described by Model (19) but it was approximated by (20) via PCA for a certain $k \ll p$. Based on the ith row of the data, the true model was expressed by (5) with $Z_i = Y_i$ and $\mathbf{w}_i' = (1, \mathbf{x}_i)'$, where Y_i was the ith element of $\mathbf{y}$ and $\mathbf{x}_i$ was the ith row of $\mathbf{X}$.

As dimension reduction is often used to highly correlated explanatory variables, we used

$$\mathbf{x}_i \sim^{iid} N(\boldsymbol{\nu}, \boldsymbol{\Sigma}) \tag{24}$$

to generate explanatory variables. We chose $\boldsymbol{\nu}$ as a 1000-dimensional vector with all of its elements equal to 0.1 and $\boldsymbol{\Sigma} = 0.05\mathbf{R}$, where $\mathbf{R}$ was the correlation matrix of $\mathbf{x}_i$. To derive a highly correlated $\mathbf{R}$, we first generated 1000 points in $[0, 1]$ and then defined $r_{ij} = (e^{-d_{ij}} + e^{-2d_{ij}})/2$, where d_{ij} was the distance between the points. To derive the regression coefficients, we generated $\boldsymbol{\beta}$ independently from $N(0.02, 0.01^2)$. After $\boldsymbol{\nu}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{\beta}$ were derived, we fixed their values in the simulation of $\mathbf{x}_i$ and Y_i. We generated $\mathbf{x}_i$ identically and independently from $N(\boldsymbol{\nu}, \boldsymbol{\Sigma})$. For each i, we generated Y_i independently from Model (5) with $\alpha = 5.0$ and $\sigma^2 = 4$. We generated $n = 10^7$ samples from the model. Finally, we obtained a data set with $n = 10^7$ and $p = 10^3$. With the response variable, the data set contained 1001 columns and 10^7 rows. Its size was around $173GB$. If each floating number needed 8 bytes, a computer should have at least $75GB$ memory size to load the entire data set to its memory, which was greater than the memory size of the computer that we used.

According to Theorem 4.5, we did not consider the second round usage of scanning data by rows reflected by Algorithm 2. To apply Theorem 4.5, we computed $\mathcal{S} = (s_{yy}, \mathbf{s}_{wy}, \mathbf{S}_{ww})$ using the technique of scanning data by rows, where we modified Step 3 of Algorithm 1 by updating $\mathcal{S} = \mathcal{S} + (Y_i^2, \mathbf{w}_i Y_i, \mathbf{w}_i \mathbf{w}_i')$. After $\mathcal{S}$ was derived, we first computed $\mathbf{S}_{s,xx}$ using (11) and then computed $\mathbf{D}$ and $\mathbf{V}$ via its eigenvalue and eigenvector decomposition. After that, we obtained λ_k for every $k \in \{1, \cdots, p\}$. To carry out the dimension reduction strategy, we chose the optimal value k_{opt} of k such that $\lambda_{k_{opt}-1} < 0.95$ but $\lambda_{k_{opt}} \geq 0.95$. It meant that the previous k principal components contained at least 95% total variations of explanatory variables. We had $k_{opt} = 6$. Therefore, we decided to consider the model with the previous six principal components. Finally, we applied (22) and (23) to compute $\hat{\boldsymbol{\beta}}_s$, $\hat{\sigma}_s^2$, and $\mathbb{V}(\hat{\boldsymbol{\beta}}_s)$. We had $R^2 = 0.7670$, $\hat{\sigma}_s^2 = 2.00091^2$, $\hat{\boldsymbol{\beta}}_s = (447.145, 2.369, 41.082, 6.111, 10.768, -7.782)'$ with standard error vector $s(\hat{\boldsymbol{\beta}}_s) = (0.0780, 0.1554, 0.2559, 0.3646, 0.4693, 0.5876)'$, indicating that all of the six principal components were significant. Since $\hat{\sigma}_s^2$ was close to the true σ^2 value, we concluded that the PCA approach successfully reduced the dimension of explanatory variables from one thousand to six. The time taken of the computations provided by **C++** showed that the computation of $\mathcal{S}$ cost 129,318 seconds (around 35.92 hours), the computation of $\mathbf{S}_{s,xx}$ from $\mathbf{S}_{ww}$ cost less than one second, the computation of eigenvalue and eigenvector decomposition of $\mathbf{S}_{ww}$ cost about ten seconds, and the rest computations cost less than one second.

5.2 Logistic regression

Logistic regression is popular in the analysis of binomial data. Suppose the response variable is provided by (Y_i, m_i), where Y_i is the number of successes and m_i is the number of trials provided in the ith row of the data. If $Y_i \sim Bin(m_i, \pi_i)$ independently, then a logistic regression model is

$$\log[\pi_i/(1-\pi_i)] = \alpha + \mathbf{x}_i'\boldsymbol{\beta}, i = 1, \cdots, n. \tag{25}$$

To generate binomial data, we also used (24) to generate explanatory variables in (25). We chose $\boldsymbol{\nu}$ as a 1000-dimensional vector with all of its elements equal to 0.0005 and $\Sigma = 0.00025\mathbf{R}$, where the correlation matrix $\mathbf{R}$ was generated in the same way as we did in Section 5.1. We generated $\boldsymbol{\beta}$ independently from $N(0.03, 0.003^2)$. After $\boldsymbol{\nu}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{\beta}$ were derived, we fixed their values. For each i, we generated $\mathbf{x}_i$ and then computed π_i by (25) with $\alpha = 0.1$. Next, we generated m_i from $Poisson(1000)$ and then generated the binomial response Y_i. All of the rows were generated independently. We still chose $n = 10^7$ and finally we obtained a data set with $n = 10^7$ rows and $p + 2 = 1002$ columns, where the response variable was represented by two columns. The size of the data set was around $173GB$. A computing system needed at least $75GB$ to load the entire data set to its memory.

As Theorem 4.5 could not be used to binomial data, we decided to consider both Algorithms 1 and 2 in our PCA approach, where k_{opt} was still selected by the minimum k containing at least 95% total variations of the explanatory variables. In our approach, we computed $\mathbf{S}_{ww}$ using Step 3 of Algorithm 1. We derived $\mathbf{D}$ and $\mathbf{V}$ by computing $\mathbf{S}_{s,xx}$ and its eigenvalue and eigenvector decomposition. We obtained $k_{opt} = 6$. Using $k = k_{opt} = 6$, we obtained the formulae of the first six principal components, which were provided by the first six columns of $\mathbf{V}$. After that, we carried out Algorithm 2 for the second usage of the technique of scanning data by rows. The algorithm provided a file in hard disk. It had two columns for the response variable and six columns for the principal components. The size of the file was about $0.81GB$. To load the file, a computing system needed about $0.6GB$ memory size. After columns were reduced by PCA, the size of the data was not large. We decided to carry out a further analysis for (18) via $\mathbf{R}$. The result provided $\hat{\boldsymbol{\beta}}_s = (47.1968, 0.8605, 4.3765, 0.2027, -1.1300, 0.02854)'$ with its standard error vector $s(\hat{\boldsymbol{\beta}}_s) = (0.0026, 0.0051, 0.0081, 0.0151, 0.0150, 0.0188)'$. The residual deviance of the model was $G^2 = 10{,}010{,}607$ and the residual degree of freedom was 9,999,993, implying that the model fitted the data. The time taken of Algorithm 1 was 134,076 seconds (about

37.24 hours). The time taken of Algorithm 2 was 70,358 seconds (about 19.54 hours).

5.3 Loglinear regression

Loglinear regression is the most popular approach in the analysis of Poisson data when observations are independently collected. If the ith response variable Y_i follows a Poisson distribution with mean λ_i, then a loglinear regression can be proposed as

$$\log \lambda_i = \alpha + \mathbf{x}_i'\boldsymbol{\beta}, i = 1, \cdots, n. \tag{26}$$

To generate the Poisson data, we still used (24) to generate explanatory variables in Model (26), where we chose $\boldsymbol{\nu}$ as a 1000-dimensional vector with all of its elements equal to 0.03 and $\Sigma = 0.001\mathbf{R}$, where the correlation matrix $\mathbf{R}$ was generated in the same way as before. We generated $\boldsymbol{\beta}$ independently from $N(0.003, 0.006^2)$. After $\boldsymbol{\nu}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{\beta}$ were derived, we fixed their values. We generated $\mathbf{x}_i$ from $N(\boldsymbol{\nu}, \boldsymbol{\Sigma})$ and then computed λ_i with $\alpha = 5$ via (26). After that, we generated Y_i. All of those were derived independently. We also used $n = 10^7$ and finally we obtained a data set with 10^7 rows and 1001 columns, where the response was represented by one column. The size of the data was also around $173GB$. A computing system also needed at least $75GB$ to load the entire data set to its memory.

We carried out exactly the same procedure as we did for the binomial data, where k_{opt} also was selected by the minimum k containing at least 95% total variations of the explanatory variables. We also considered both Algorithms 1 and 2. After Algorithm 1 was over, we obtained a data file on the hard disk, which had 10^7 rows and 7 columns, where one column represented the response and the other six columns represent the principal components. The size of the data was about $0.75GB$. We also carried out a further analysis for (18) via $\mathbf{R}$. The results provided $\hat{\boldsymbol{\beta}}_s = (95.2614, -0.7691, 9.1421, -0.0875, 1.3755, 0.4385)'$ and $s(\hat{\boldsymbol{\beta}}_s) = (0.0026, 0.0051, 0.0083, 0.0188, 0.0155, 0.0193)'$. The residual deviance was $G^2 =$ 10,000,825 and the residual degree of freedom was 9,999,993, indicating the model fitted the data. The time taken of Algorithm 1 was 130,767 seconds (about 36.32 hours). The time taken of Algorithm 2 was 71,565 seconds (about 19.88 hours).

6. APPLICATION

We applied our approach to the 1999 KDD (Knowledge Discovery in Databases) data set, which can be downloaded from the website of the University of California Irvine (UCI) Machine Learning Repository. The 1999 KDD data set, prepared by [29], was built based on data captured in an intrusion detection system (IDS) evaluation program. The IDS monitors the security status of networks and detects abnormal behaviors. The 1999 data set contains four different types of attacks. The DOS (denial of service) is a class of attacks in which an attacker makes computing or memory resources too busy to handle legitimate requests. It can cause denies of legitimate users. The U2R (user to root) is a class of attacks using normal accounts to access the root of the system. The R2L (remote to local) is a class of attacks for unauthorized accesses via remote machines. R2L attacks may contain guessing passwords. The probing is a class of attacks of gathering information for the purpose of circumventing security controls (e.g., port scanning). The 1999 data set was previously discussed by may authors [29, 30, e.g.]. It was one of the mostly widely used data sets for evaluation of anamaly detection methods in network security.

The 1999 KDD data set contained 4,898,431 records for types of attacks, including normal users. It had 3,883,370 (79.28%) DOS attacks, 52 (< 0.01%) U2R attacks, 1,126 (0.02%) R2L attacks, 41,102 (0.84%) probing attacks, and 972,781 (19.86%) normal users. As most of the attacks were DOS, we defined a binary variable for whether records were DOS or not. We chose the binary variable as the response variable. Besides the response, the data set also contained 41 explanatory variables. Thirty-eight of those were continuous and the other three were categorical. As four continuous explanatory variables were almost kept at same values, we decided to exclude them from our analysis. Then, at most 34 continuous explanatory variables could be used to account for the response variable. After continuous explanatory variables were determined, we also studied the three factor (i.e., categorical explanatory) variables. As the service type was more important and more interesting than the other two, we decided to use service type as the only factor in our approach. Therefore, our analysis involved 34 continuous explanatory variables and one factor variable.

We used logistic regression to analyze the data. Assume the factor variable has J levels. Let Y_{ij} be the ith value of the response variable and $\mathbf{x}_{ij}$ be the ith vector of the continuous explanatory variables at the jth level of the factor variable, where $Y_{ij} = 1$ if the attack is DOS and $Y_{ij} = 0$ otherwise. If $Y_{ij} \sim Bernoulli(\pi_{ij})$ independently, then a logistic regression model with the interaction effect between the continuous and factor variables is expressed as

$$\log[\pi_{ij}/(1 - \pi_{ij})] = \beta_{0j} + \mathbf{x}_{ij}'\boldsymbol{\beta}_j, \tag{27}$$

for $i = 1, \cdots, n_{ij}$ and $j = 1, \cdots, J$, where n_{ij}, β_{0j}, and $\boldsymbol{\beta}_j$ represent the sample size, the intercept term, and the slope vector of the continuous explanatory variables at the jth level of the factor variable. The total sample size is $n = \sum_{i=1}^{J} n_{ij}$. A main effect model is derived if $\boldsymbol{\beta}_j$ are all equal.

We investigated possible ways for fitting (27) to the 1999 KDD data. We noted that the factor variable (i.e., service type) had 70 levels. Many of them contained a few hundred observations or less. After these levels were combined, the factor variable still had 16 levels (i.e., $J = 16$). To fit (27), one should open at least one matrix with $n =$ 4,898,431 rows and $16 \times 35 = 560$ columns. To load the matrix, a computer should have at least $20GB$ memory size (assuming each floating number is 8 bytes). As a fitting procedure often needs to open several matrices with this size, the memory need is often a few times higher than $20GB$, indicating that we cannot apply (27) to the original observed values of the 1999 KDD data.

We carried out a dimension reduction approach to reduce the number of columns in our analysis. The approach was only applied to the continuous explanatory variables. We considered Algorithms 1 and 2 in the same way as we did for the logistic regression in Section 5.2. Our results of Algorithm 1 indicated that the first six principal components could account for around 70% of the total variation. We decided to apply Algorithm 2 with $k_{opt} = 6$. We obtained a data set with 8 columns, including the response variable, the factor variable, and six principal components. The size of the data set was about $0.6GB$. The time taken of Algorithm 1 was 1,978 seconds (about 32.97 minutes). The time taken of Algorithm 2 was 1,600 seconds (about 26.67 minutes).

After results of dimension reduction were derived, we carried out a further analysis using the *glm* procedure in **R**. As a factor variable was involved, we did not use (18). Instead, we studied a few options of logistic regression models. If the interaction effect between the factor variable and principal components were involved, then an interaction effect model was obtained as

$$\log[\pi_{ij}/(1-\pi_{ij})] = \beta_{0j} + \sum_{k=1}^{K} pc_{ij,k}\beta_{jk}, \tag{28}$$

for $i = 1, \cdots, n_i$, $j = 1, \cdots, J$, and $K \le 6$, where $pc_{ij,k}$ was the ith value of the kth principal component at the jth level of the factor variable. We attempted to fit (28) with $K = 1, \cdots, 6$. We obtained results of (28) with $K = 1, 2, 3$. When we fitted (28) with $K = 4$, the *glm* procedure was out-of-memory. To check the reason, we monitored the memory record of the **R** procedure with $K = 3$ via the *Windows Task Manager*. It reported that the maximum memory size occupied by **R** was over $15GB$, which was just slightly lower than the memory size ($16GB$) of our machine. Therefore, we concluded that fitting (28) with four principal components was impossible in our machine.

We also studied the main effect model. We removed interaction effect from (28) when $K = 3$ and obtained the main effect model as

$$\log[\pi_{ij}/(1-\pi_{ij})] = \beta_{0j} + \sum_{k=1}^{3} pc_{ij,k}\beta_{k}, \tag{29}$$

for $i = 1, \cdots, n_i$ and $j = 1, \cdots, J$. The model contained the main effect of the factor variable (defined by β_{0j}) and the main effect of the first three principal components (defined by β_k with $k = 1, 2, 3$). It was derived by letting $\beta_{jk} = \beta_{j'k}$ for all $j, j' = 1, \cdots, J$ in (28). To compare, we considered two other models. In the first, we removed the factor variable from (29) by letting β_{0j} all equal and obtained the model with only the first three principal components. In the second, we removed the main effects of the principal components from (29) and obtained the model with only the factor variable. The residual deviance of the model with $k = 3$ in (28) was G^2_{Int} = 164,627. The value of its model degrees of freedom was 59. The value of its residual degrees of freedom was 4,898,317. Using the null deviance value G^2_{Null} = 4,998,871, we concluded that the model accounted for about 96.7% total deviance value using only 59 degrees of freedom (i.e., $96.7\% = (G^2_{Null} - G^2_{Int})/G^2_{Null}$). The residual deviance of (29) was G^2_{Main} = 195,305. It accounted for about 96.1% total deviance value using 17 degrees of freedom. In the comparison between the two models, we concluded that the interaction effects between the factor variable and the first three principal components were significant (p-value less than 10^{-16}). The residual deviance of the model with only the first three principal components was G^2_{Prin} = 460,245. It accounted for about 90.8% total deviance value using 3 degrees of freedom. The residual deviance of the model with only the factor variable was G^2_{Fac} = 789,376. It accounted for about 84.2% total deviance value with 15 degrees of freedom. Our results showed that all of the main and interaction effects were important. It was enough for us to use the first three principal components and service types to model whether the network attack was DOS or not.

7. DISCUSSION

As the first step of classical PCA approaches is to load the entire data set to memory, they cannot be applied if the size of data exceeds the memory size of a computer. To solve the problem, we propose a new PCA approach, where the basic idea is to use the technique of scanning data by rows. We demonstrate that our PCA approach can be applied even if the size of data is higher than the memory size of the computer. As the technique of scanning data by rows can be used either in an individual way or a parallel way, our PCA approach can be applied to a personal computer or a distributed computing system. If a personal computer is used, then all of the computations are carried out in a single processor. The approach can be applied if the observed data can be saved to the hard disk of a personal computer. If a distributed computing system is used, then the computations are carried out in many processors. The approach can be applied if the observed data cannot be saved to the hard disk of a personal computer. Therefore, the application of our PCA approach relies on the hard disk size but not the memory size of a computing system.

In classical PCA, loadings, singular values, and principal components are treated equally important. A classical PCA approach often attempts to provide all of these via singular value decomposition of a standardized matrix of explanatory variables. In big data analysis, standardization of a matrix itself is a concern and should be avoided in the computation, implying that many classical statistical approaches cannot be used, where PCA for dimension reduction is just one of those. In our PCA, we treat loadings and singular values more important than principal components. We show that it is not necessary to provide principal components if the linear regression approach is considered in the follow-up analysis. However, we should provide principal components if other approaches are considered. Therefore, in dimension reduction for big data, the choice of the follow-up analysis is also important.

More importantly, our research provides a novel way in understanding new statistical challenges in big data analysis. In classical statistics, computational efficiency is often studied after the entire data set is completely loaded to memory of a computer. Statistical and computational methods are compared according to their time usages in memory, where the way of reading data is often ignored. This is not a concern in the analysis of small or moderate data but it is a concern in the analysis of big data. Based on our research, we find that the way of reading data significantly affects the computational efficiency of the entire data analysis procedure. It is possible to provide exact solutions even if the data set is extremely large. This is an important property in PCA for big data when generalized linear models are used in the follow-up analysis. It will have great impacts on future statistical research on big data.

ACKNOWLEDGEMENT

The authors appreciate the comments from the editor, an associate editor, and three anonymous referees, that significantly improves the quality of the article.

APPENDIX A. PROOFS

Proof of Theorem 4.1: As $\mathbf{X}_s = \mathbf{U}_s\mathbf{D}_s\mathbf{V}_s'$ and $\mathbf{S}_{s,xx} = \mathbf{X}_s'\mathbf{X}_s = \mathbf{V}\mathbf{D}^2\mathbf{V}'$, there is $\mathbf{V}_s\mathbf{D}_s^2\mathbf{V}_s' = \mathbf{V}\mathbf{D}^2\mathbf{V}'$. Using the uniqueness of eigenvalues in the eigenvalue and eigenvector decomposition approach for a real symmetric matrix, there is $\mathbf{D}^2 = \mathbf{D}_c^2$. As both $\mathbf{D}$ and $\mathbf{D}_c$ are diagonal matrices composed of nonnegative real values, there is $\mathbf{D} = \mathbf{D}_c$. □

Proof of Theorem 4.2: If eigenvalues of $\mathbf{S}_{s,xx}$ are all different, then the dimension of eigenvector space corresponding to each eigenvalue of $\mathbf{S}_{s,xx}$ is one. Using the uniqueness of eigenvector space of a real symmetric matrix, we conclude $\mathbf{v}_j = \mathbf{v}_{cj}$ or $\mathbf{v}_j = -\mathbf{v}_{cj}$ for every $j = 1, \cdots, p$. By the definition of $\mathbf{u}_j$ given by (14), we further conclude $\mathbf{u}_j = \mathbf{u}_{cj}$ if $\mathbf{v}_j = \mathbf{v}_{cj}$ or $\mathbf{u}_j = -\mathbf{u}_j$ if $\mathbf{v}_j = -\mathbf{v}_{cj}$. □

Proof of Theorem 4.3: The eigenvector space of $\mathbf{S}_{s,xx}$ corresponding to $d_{\gamma_1}, \cdots, d_{\gamma_2-1}$ is spanned by their eigenvectors $\mathbf{v}_{\gamma_1}, \cdots, \mathbf{v}_{\gamma_2-1}$. It is unique and its dimension is $\gamma_2 - \gamma_1$. For any orthogonal matrix $\mathbf{Q}$ on $\mathbb{R}^{\gamma_2-\gamma_1}$, there is $\text{span}\{\mathbf{v}_{\gamma_1}, \cdots, \mathbf{v}_{\gamma_2-1}\} = \text{span}\{(\mathbf{v}_{\gamma_1}, \cdots, \mathbf{v}_{\gamma_2-1})\mathbf{Q}\}$, implying that $(\mathbf{v}_{\gamma_1}, \cdots, \mathbf{v}_{\gamma_2-1})\mathbf{Q}$ also provides eigenvectors corresponding to those eigenvalues. Therefore, there exists an orthogonal matrix $\mathbf{Q}$ on $\mathbb{R}^{\gamma_2-\gamma_1}$ such that $(\mathbf{v}_{\gamma_1}, \cdots, \mathbf{v}_{\gamma_2}) = (\mathbf{v}_{c\gamma_1}, \cdots, \mathbf{v}_{c\gamma_2-1})\mathbf{Q}$. Using (14), we conclude $(\mathbf{u}_{\gamma_1}, \cdots, \mathbf{u}_{\gamma_2}) = (\mathbf{u}_{c\gamma_1}, \cdots, \mathbf{u}_{c\gamma_2-1})\mathbf{Q}$. □

Proof of Corollary 4.1: By applying the conclusion of Theorem 4.3 to each eigenvector space spanned by eigenvectors of distinct d_{0l}, $l = 1, \cdots, q$, we obtain $\mathbf{Q}_l$ for each $l = 1, \cdots, q$. Since eigenvector spaces of distinct eigenvalues of a real symmetric matrix are orthogonal, we conclude $\mathbf{Q} = \text{diag}(\mathbf{Q}_1, \cdots, \mathbf{Q}_q)$ is also an orthogonal matrix, implying the conclusion of the Corollary. □

Proof of Theorem 4.4: If $d_k > d_{k+1}$, then we can find an $a \in \{1, \cdots, q\}$ such that $d_k = d_{0a} > d_{k+1} = d_{0(a+1)}$, implying that $(\mathbf{X}_s\mathbf{v}_1, \cdots, \mathbf{X}_s\mathbf{v}_k) = (\mathbf{X}_s\mathbf{v}_{c1}, \cdots, \mathbf{X}_s\mathbf{v}_{ck})\mathbf{Q}_k$, where $\mathbf{Q}_k = \text{diag}(\mathbf{Q}_1, \cdots, \mathbf{Q}_a)$ is uniquely determined the eigenvector space spanned by eigenvectors corresponding to $d_{01}, \cdots, d_{0a}$. Therefore, the MLE of $\boldsymbol{\eta}$ is solution of $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_n)$ by maximizing the likelihood function of the data. Since the design matrix $(\mathbf{1}, PC_1, \cdots, PC_k)$ of (18) is full rank, the MLE of $\boldsymbol{\eta}$ is unique [25], implying the conclusion of the Corollary. □

Proof of Theorem 4.5: Let $s_{j,wy}$ and $s_{s,j,xy}$ be the jth element of $\mathbf{s}_{wy}$ and $\mathbf{s}_{s,xy}$, respectively, where we use $j = 0, 1, \cdots, p$ for $\mathbf{s}_{wy}$ and $j = 1, \cdots, p$ for $\mathbf{s}_{s,xy}$. Then, $s_{s,j,xy} = (s_{j,wy} - n\bar{x}_j\bar{y})/b_j$, implying that one can derive $\mathbf{s}_{s,xy}$ once $\mathcal{S}$ is available. The conclusion is drawn as $\mathbf{U}$ is not used in (22) or (23). □

Received 5 August 2016

REFERENCES

[1] Abdi, H. and Williams, L. (2010). Principal component analysis. *WIRES Computational Statistics*, **2**, 433–459.

[2] Bay, S. D., Kibler, D., Pazzani, M. J., and Smyth, P. (2000). The UCI KDD archive of large data sets for data mining research and experimentation. *ACM SIGKDD Explorations and Newsletter*, **2**, 81–85.

[3] Candés, E. and Li, X. (2011). Robust Principal Component analysis. *Journal of the ACM*, **58**, Article 11. MR2811000

[4] Celik, T. (2009). Unsupervised change detection in satellite images using principal component analysis and K-means clustering. *IEEE Geoscience and Remote Sensing Letters*, **6**, Article 4.

[5] Choi, S. W., Park, J. H., Lee, I. B. (2004). Process monitoring using a Gaussian mixture model via principal component analysis and discriminant analysis. *Computer and Chemical Engineering*, **28**, 1377–1387.

[6] Cook, R. D. and Nachtsheim, C. J. (1994). Re-weighting to archive elliptically contoured covariates in regression. *Journal of the American Statistical Association*, **89**, 592–599.

[7] Cook, R. D. and Li, B. (2002). Dimension reduction for conditional mean in regression. *Annals of Statistics*, **30**, 455–474. MR1902895

[8] Cook, R. D. (2007). Fisher lecture: dimension reduction in regression. *Statistical Science*, **22**, 1–26. MR2408655

[9] Dean, J. and Ghamawat, S. (2004). MapRedue: simplified data processing on large clusters. In *Proceeding of OSDI*, 137–150.

[10] Du, Q. (2007). Hyperspectral image compression using JPEG2000 and principal component analysis. *IEEE Geoscience and Remote Sensing Letters*, **4**, 201–205.

[11] Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M., Benítez, J. M., and Herrera, F. (2014). Big data with cloud computing: an insight on the computing environment, MapReduce, and programming frameworks. *WIREs Data Mining Knowledge Discovery*, **4**, 380–409.

[12] Gardner, J. W. (1991). Detection of vapours and odours from a multisensor array using pattern recognition Part I. principal

component and cluster analysis. *Sensors and actuators B*, **4**, 109–11.

[13] Hall, P., Müller, H. and Wang, J. (2006). Properties of principal component methods for functional and longitudal data analysis. *Annals of Statistics*, **3**, 1493–1517. MR2278365

[14] Hotelling, H. (1933). Analysis of a complex of statistical variables in principal components. *Journal of Educational Psychology*, **24**, 417–441.

[15] Jolliffe, I. (2002). *Principal Component Analysis, 2nd Edition.* New York: Springer-Verlag. MR2036084

[16] Kim, K. I., Franz, M. O., and Schölkopf, B. (2005). Iterative kernel principal component analysis for image modeling. *IEEE Transactions on pattern analysis and machine intelligence*, **27**, 1351–1366.

[17] Li, B. and Wang, S. (2007). On directional regression for dimension reduction. *Journal of the American Statistical Association*, **102**, 997–1008. MR2354409

[18] Li, K. C. and Duan, N. (1989). Regression analysis under link violation. *The Annals of Statistics*, **17**, 1009–1052. MR1015136

[19] Li, K. C. (1991). Sliced inverse regression for dimension reduction (with discussion). *Journal of the American Statistical Association*, **86**, 316–342. MR1137117

[20] Lin, N. and Xi, R. (2011). Aggregated estimating equation estimation. *Statistics and Its Interface*, **4**, 73–83. MR2775250

[21] Ma, P. and Sun, X. (2015). Leveraging for big data regression. *WIREs Computational Statistics*, **7**, 70–76. MR3348722

[22] Marx, B., Smith, E. P. (1990). Principal component estimation for generalized linear regression. *Biometrika*, **77**, 23–31. MR1049405

[23] Meeker, W. Q. and Hong, Y. (2014). Reliability meets big data: opportunities and challenges. *Qualify Engineering*, **26**, 102–116.

[24] Miner, D. and Shook, A. (2012). *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems. O'Reilly Media Inc*, Sebastpool, California.

[25] Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of Royal Statistical Society Series A*, **135**, 370–384.

[26] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, **2**, 559–572.

[27] Reyes-Ortiz, J. L., Oneto, L., and Anguita, D. (2015). Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf. *Precedia Computer Science*, **53**, 121–130.

[28] Sousa, S. I. V., Matins, F. G., Alvim-Ferraz, M. C. M., Pereira, M. C. (2007). Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. *Environmental Modelling & Software*, **22**, 97–103.

[29] Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., and Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: results from the JAM project. *Proceedings of DARPA Information Survivability Conference and Exposition*, **2**, 130–144.

[30] Tavallaee, M. Bagheri, E., Lu, W., Ghorbani, A. A. (2009). A detalied analysis of the KDD cup 99 data set. *Proceedings of the second IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA2009).* USA: IEEE Press 2009, 53–58.

[31] Xia, Y., Tong, H., Li, W. K., and Zhu, L. X. (2002). An adaptive estimation of dimension reduction space. *Journal of Royal Statistical Society*, **64**, 363–410. MR1924297

[32] Vitter, J. S. (2008). *Algorithms and Data Structures for External Memory*, Now Publication Inc, Hanover, MA, USA. MR2453149

[33] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: cluster computing with working sets. *HotCloud*, **10**, Article 10.

[34] Zhang, L., Dong, W., Zhang, D., and Shi, G. (2010). Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, **43**, 1531–1549.

[35] Zhang, T. and Yang, B. (2017). Box-Cox transformation in big data. *Technometrics*, **59**, 189–201. MR3635042

[36] Zhao, G. and Mactean, A. L. (2000). A comparison of canonical discriminant analysis and principal component analysis for spectral transformation. *Photogrammetric engineering & Remote Sensing*, **66**, 841–847.

Tonglin Zhang
Department of Statistics
Purdue University
250 North University Street
West Lafayette, IN 47907-2066
USA
E-mail address: tlzhang@purdue.edu

Baijian Yang
Department of Computer and Information Technology
Purdue University
401 North Grant Street
West Lafayette, IN 47907
USA
E-mail address: byang@purdue.edu