Acceleration of Convergence of the Alternating Least Squares Algorithm for Nonlinear Principal Components Analysis

Masahiro Kuroda¹, Yuichi Mori¹, Masaya Iizuka² and Michio Sakakihara¹

1 Okayama University of Science
2 Okayama University
Japan

1. Introduction

Principal components analysis (PCA) is a popular descriptive multivariate method for handling quantitative data. In PCA of a mixture of quantitative and qualitative data, it requires quantification of qualitative data to obtain optimal scaling data and use ordinary PCA. The extended PCA including such quantification is called nonlinear PCA, see Gifi [Gifi, 1990]. The existing algorithms for nonlinear PCA are PRINCIPALS of Young et al. [Young et al., 1978] and PRINCALS of Gifi [Gifi, 1990] in which the alternating least squares (ALS) algorithm is utilized. The algorithm alternates between quantification of qualitative data and computation of ordinary PCA of optimal scaling data.

In the application of nonlinear PCA for very large data sets and variable selection problems, many iterations and much computation time may be required for convergence of the ALS algorithm, because its speed of convergence is linear. Kuroda et al. [Kuroda et al., 2011] proposed an acceleration algorithm for speeding up the convergence of the ALS algorithm using the vector ε (εv) algorithm of Wynn [Wynn, 1962]. During iterations of the εv accelerated ALS algorithm, the εv algorithm generates an accelerated sequence of optimal scaling data estimated by the ALS algorithm. Then the εv accelerated sequence converges faster than the original sequence of the estimated optimal scaling data. In this paper, we use PRINCIPALS as the ALS algorithm for nonlinear PCA and provide the εv acceleration for PRINCIPALS (εv-PRINCIPALS). The computation steps of PRINCALS are given in Appendix A. As shown in Kuroda et al. [Kuroda et al., 2011], the εv acceleration is applicable to PRINCALS.

The paper is organized as follows. We briefly describe nonlinear PCA of a mixture of quantitative and qualitative data in Section 2, and describe PRINCIPALS for finding least squares estimates of the model and optimal scaling parameters in Section 3. Section 4 presents the procedure of εv-PRINCIPALS that adds the εv algorithm to PRINCIPALS for speeding up convergence and demonstrate the performance of the εv acceleration using numerical experiments. In Section 5, we apply εv-PRINCIPALS to variable selection in nonlinear PCA. Then we utilize modified PCA (M.PCA) approach of Tanaka and Mori [Tanaka and Mori, 1997] for variable selection problems and give the variable selection procedures in M.PCA of qualitative data. Numerical experiments examine the the performance and properties of εv-PRINCIPALS. In Section 6, we present our concluding remarks.
2. Nonlinear principal components analysis

PCA transforms linearly an original data set of variables into a substantially smaller set of uncorrelated variables that contains much of the information in the original data set. The original data matrix is then replaced by an estimate constructed by forming the product of matrices of component scores and eigenvectors.

Let $X = (X_1 \, X_2 \, \cdots \, X_p)$ be an $n \times p$ matrix of $n$ observations on $p$ variables and be columnwise standardized. In PCA, we postulate that $X$ is approximated by the following bilinear form:

$$
\hat{X} = ZA^\top,
$$

(1)

where $Z = (Z_1 \, Z_2 \, \cdots \, Z_r)$ is an $n \times r$ matrix of $n$ component scores on $r$ ($1 \leq r \leq p$) components, and $A = (A_1 \, A_2 \, \cdots \, A_r)$ is a $p \times r$ matrix consisting of the eigenvectors of $X'X/n$ and $A^\top A = I_r$. Then we determine model parameters $Z$ and $A$ such that

$$
\theta = \text{tr}(X - \hat{X})^\top(X - \hat{X}) = \text{tr}(X - ZA^\top)^\top(X - ZA^\top)
$$

(2)

is minimized for the prescribed $r$ components.

Ordinary PCA assumes that all variables are measured with interval and ratio scales and can be applied only to quantitative data. When the observed data are a mixture of quantitative and qualitative data, ordinary PCA cannot be directly applied to such data. In such situations, optimal scaling is used to quantify the observed qualitative data and then ordinary PCA can be applied.

To quantify $X_j$ of qualitative variable $j$ with $K_j$ categories, the vector is coded by using an $n \times K_j$ indicator matrix $G_j$ with entries $g_{(j)ik} = 1$ if object $i$ belongs to category $k$, and $g_{(j)ik'} = 0$ if object $i$ belongs to some other category $k'(\neq k)$, $i = 1, \ldots, n$ and $k = 1, \ldots, K_j$. Then the optimally scaled vector $X_j^*$ of $X_j$ is given by $X_j^* = G_j \alpha_j$, where $\alpha_j$ is a $K_j \times 1$ score vector for categories of $X_j$. Let $X^* = (X_1^* \, X_2^* \, \cdots \, X_p^*)$ be an $n \times p$ matrix of optimally scaled observations to satisfy restrictions

$$
X^*^\top 1_n = 0_p \quad \text{and} \quad \text{diag} \left[ \frac{X^*^\top X^*}{n} \right] = I_p,
$$

(3)

where $1_n$ and $0_p$ are vectors of ones and zeros of length $n$ and $p$ respectively. In the presence of nominal and/or ordinal variables, the optimization criterion (2) is replaced by

$$
\theta^* = \text{tr}(X^* - \hat{X})^\top(X^* - \hat{X}) = \text{tr}(X^* - ZA^\top)^\top(X^* - ZA^\top).
$$

(4)

In nonlinear PCA, we determine the optimal scaling parameter $X^*$, in addition to estimating $Z$ and $A$.

3. Alternating least squares algorithm for nonlinear principal components analysis

A possible computational algorithm for estimating simultaneously $Z$, $A$ and $X^*$ is the ALS algorithm. The algorithm involves dividing an entire set of parameters of a model into the model parameters and the optimal scaling parameters, and finds the least squares estimates
for these parameters. The model parameters are used to compute the predictive values of the model. The optimal scaling parameters are obtained by solving the least squares regression problem for the predictive values. Krijnen [Krijnen, 2006] gave sufficient conditions for convergence of the ALS algorithm and discussed convergence properties in its application to several statistical models. Kiers [Kiers, 2002] described setting up the ALS and iterative majorization algorithms for solving various matrix optimization problems.

PRINCIPALS

PRINCIPALS proposed by Young et al. [Young et al., 1978] is a method for utilizing the ALS algorithm for nonlinear PCA of a mixture of quantitative and qualitative data. PRINCIPALS alternates between ordinary PCA and optimal scaling, and minimizes $\theta^*$ defined by Equation (4) under the restriction (3). Then $\theta^*$ is to be determined by model parameters $Z$ and $A$ and optimal scaling parameter $X^*$, by updating each of the parameters in turn, keeping the others fixed.

For the initialization of PRINCIPALS, we determine initial data $X^*(0)$. The observed data $X$ may be used as $X^*(0)$ after it is standardized to satisfy the restriction (3). For given initial data $X^*(0)$ with the restriction (3), PRINCIPALS iterates the following two steps:

- **Model parameter estimation step**: Obtain $A^{(t)}$ by solving

$$\begin{bmatrix} X^{(t)^\top} X^{(t)} \\ n \end{bmatrix} A = AD_r,$$

where $A^\top A = I_r$ and $D_r$ is an $r \times r$ diagonal matrix of eigenvalues, and the superscript $(t)$ indicates the $t$-th iteration. Compute $Z^{(t)}$ from $Z^{(t)} = X^{*(t)} A^{(t)}$.

- **Optimal scaling step**: Calculate $\hat{X}^{(t+1)} = Z^{(t)} A^{(t)^\top}$ from Equation (1). Find $X^{*(t+1)}$ such that

$$X^{*(t+1)} = \arg\min_{X^*} \text{tr}(X^* - \hat{X}^{(t+1)})^\top (X^* - \hat{X}^{(t+1)})$$

for fixed $\hat{X}^{(t+1)}$ under measurement restrictions on each of the variables. Scale $X^{*(t+1)}$ by columnwise centering and normalizing.

4. The $v\varepsilon$ acceleration of the ALS algorithm

We briefly introduce the $v\varepsilon$ algorithm of Wynn [Wynn, 1962] used in the acceleration of the ALS algorithm. The $v\varepsilon$ algorithm is utilized to speed up the convergence of a slowly convergent vector sequence and is very effective for linearly converging sequences. Kuroda and Sakakihara [Kuroda and Sakakihara, 2006] proposed the $v\varepsilon$-accelerated EM algorithm that speeds up the convergence of the EM sequence via the $v\varepsilon$ algorithm and demonstrated that its speed of convergence is significantly faster than that of the EM algorithm. Wang et al. [Wang et al., 2008] studied the convergence properties of the $v\varepsilon$-accelerated EM algorithm.

Let $\{Y^{(t)}\}_{t \geq 0} = \{Y^{(0)}, Y^{(1)}, Y^{(2)}, \ldots\}$ be a linear convergent sequence generated by an iterative computational procedure and let $\{\hat{Y}^{(t)}\}_{t \geq 0} = \{\hat{Y}^{(0)}, \hat{Y}^{(1)}, \hat{Y}^{(2)}, \ldots\}$ be the accelerated
sequence of \{Y^{(t)}\}_{t \geq 0}. Then the \(v\varepsilon\) algorithm generates \{\dot{Y}^{(t)}\}_{t \geq 0} by using
\[
\dot{Y}^{(t-1)} = Y^{(t)} + \left[ (Y^{(t-1)} - Y^{(t)})^{-1} + (Y^{(t+1)} - Y^{(t)})^{-1} \right]^{-1},
\]
where \(Y^{-1} = Y / ||Y||^2\) and \(||Y||\) is the Euclidean norm of \(Y\). For the detailed derivation of Equation (6), see Appendix B. When \{\dot{Y}^{(t)}\}_{t \geq 0} converges to a limit point \(Y^{(\infty)}\) of \{\dot{Y}^{(t)}\}_{t \geq 0}, it is known that, in many cases, \(\dot{Y}^{(t)}_{t \geq 0}\) generated by the \(v\varepsilon\) algorithm converges to \(Y^{(\infty)}\) faster than \(\{Y^{(t)}\}_{t \geq 0}\).

We assume that \{\(X^{*(t)}\)\}_{t \geq 0} generated by PRINCIPALS converges to a limit point \(X^{*\infty}\). Then \(v\varepsilon\)-PRINCIPALS produces a faster convergent sequence \{\(X^{*(t)}\)\}_{t \geq 0} of \{\(X^{*(t)}\)\}_{t \geq 0} by using the \(v\varepsilon\) algorithm and enables the acceleration of convergence of PRINCIPALS. The general procedure of \(v\varepsilon\)-PRINCIPALS iterates the following two steps:

- **PRINCIPALS step:** Compute model parameters \(A^{(t)}\) and \(Z^{(t)}\) and determine optimal scaling parameter \(X^{*(t+1)}\).
- **Acceleration step:** Calculate \(\dot{X}^{*(t-1)}\) using \(\{X^{*(t-1)}, X^{*(t)}, X^{*(t+1)}\}\) from the \(v\varepsilon\) algorithm:

\[
\text{vec}\dot{X}^{*(t-1)} = \text{vec}X^{*(t)} + \left[ (\text{vec}(X^{*(t-1)} - X^{*(t)}))^{-1} + (\text{vec}(X^{*(t+1)} - X^{*(t)}))^{-1} \right]^{-1},
\]
where \(\text{vec}X^* = (X_1^T, X_2^T, \ldots, X_p^T)^T\), and check the convergence by

\[
||\text{vec}(\dot{X}^{*(t-1)} - \dot{X}^{*(t-2)})||^2 < \delta,
\]
where \(\delta\) is a desired accuracy.

Before starting the iteration, we determine initial data \(X^{*(0)}\) satisfying the restriction (3) and execute the PRINCIPALS step twice to generate \(\{X^{*(0)}, X^{*(1)}, X^{*(2)}\}\).

\(v\varepsilon\)-PRINCIPALS is designed to generate \(\{\dot{X}^{*(t)}\}_{t \geq 0}\) converging to \(X^{*\infty}\). Thus the estimate of \(X^*\) can be obtained from the final value of \(\{\dot{X}^{*(t)}\}_{t \geq 0}\) when \(v\varepsilon\)-PRINCIPALS terminates. The estimates of \(Z\) and \(A\) can then be calculated immediately from the estimate of \(X^*\) in the Model parameter estimation step of PRINCIPALS.

Note that \(\dot{X}^{*(t-1)}\) obtained at the \(t\)-th iteration of the Acceleration step is not used as the estimate \(X^{*(t+1)}\) at the \((t + 1)\)-th iteration of the PRINCIPALS step. Thus \(v\varepsilon\)-PRINCIPALS speeds up the convergence of \(\{X^{*(t)}\}_{t \geq 0}\) without affecting the convergence properties of ordinary PRINCIPALS.

**Numerical experiments 1: Comparison of the number of iterations and CPU time**

We study how much faster \(v\varepsilon\)-PRINCIPALS converges than ordinary PRINCIPALS. All computations are performed with the statistical package \(R\) [R Development Core Team, 2008] executing on Intel Core i5 3.3 GHz with 4 GB of memory. CPU times (in seconds) taken are measured by the function proc.time\(^1\). For all experiments, \(\delta\) for convergence

\(^1\)Times are typically available to 10 msec.
of ve-PRINCIPALS is set to $10^{-8}$ and PRINCIPALS terminates when $|\theta^{(t+1)} - \theta^{(t)}| < 10^{-8}$, where $\theta^{(t)}$ is the $t$-th update of $\theta$ calculated from Equation (4). The maximum number of iterations is also set to 100,000.

We apply these algorithms to a random data matrix of 100 observations on 20 variables with 10 levels and measure the number of iterations and CPU time taken for $r = 3$. The procedure is replicated 50 times.

Table 1 is summary statistics of the numbers of iterations and CPU times of PRINCIPALS and ve-PRINCIPALS from 50 simulated data. Figure 1 shows the scatter plots of the number of iterations and CPU time. The values of the second to fifth columns of the table and the figure show that PRINCIPALS requires more iterations and takes a longer computation time than ve-PRINCIPALS. The values of the sixth and seventh columns in the table are summary statistics of the iteration and CPU time speed-ups for comparing the speed of convergence of PRINCIPALS with that of ve-PRINCIPALS. The iteration speed-up is defined as the number of iterations required for PRINCIPALS divided by the number of iterations required for ve-PRINCIPALS. The CPU time speed-up is calculated similarly to the iteration speed-up. We can see from the values of the iteration and CPU time speed-ups that ve-PRINCIPALS converges 3.23 times in terms of the mean number of iterations and 2.92 times in terms of the mean CPU time faster than PRINCIPALS. Figure 2 shows the boxplots of the iteration and CPU time speed-ups. Table 1 and Figure 2 show that ve-PRINCIPALS well accelerates the convergence of $\{X_t^{(t)} \}_{t \geq 0}$.

![Fig. 1. Scatter plots of the number iterations and CPU time from 50 simulated data.](image1)

Figure 3 is the scatter plots of iteration and CPU time speed-ups for the number of iterations of PRINCIPALS. The figure demonstrates that the ve acceleration speeds up greatly the convergence of $\{X_t^{(t)} \}_{t \geq 0}$ and its speed of convergence is faster for the larger number of iterations of PRINCIPALS. For more than 400 iterations of PRINCIPALS, the speed of the ve acceleration is faster 3 times more than that of PRINCIPALS and the maximum values of both speed-ups are for around 1,000 iterations of PRINCIPALS. The advantage of the ve acceleration is very obvious.
Fig. 2. Boxplots of iteration and CPU time speed-ups from 50 simulated data.

Fig. 3. Scatter plots of iteration and CPU time speed-ups for the number of iterations of PRINCIPALS from 50 simulated data.
Table 1. Summary statistics of the numbers of iterations and CPU times of PRINCIPALS and \(\varepsilon\)-PRINCIPALS and iteration and CPU time speed-ups from 50 simulated data.

<table>
<thead>
<tr>
<th></th>
<th>PRINCIPALS</th>
<th>(\varepsilon)-PRINCIPALS</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration CPU time</td>
<td>Iteration CPU time</td>
<td>Iteration CPU time</td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td>136.0</td>
<td>46.0</td>
<td>1.07</td>
</tr>
<tr>
<td>1st Quartile</td>
<td>236.5</td>
<td>85.0</td>
<td>1.81</td>
</tr>
<tr>
<td>Median</td>
<td>345.5</td>
<td>137.0</td>
<td>2.72</td>
</tr>
<tr>
<td>Mean</td>
<td>437.0</td>
<td>135.0</td>
<td>2.70</td>
</tr>
<tr>
<td>3rd Quartile</td>
<td>573.2</td>
<td>171.2</td>
<td>3.40</td>
</tr>
<tr>
<td>Maximum</td>
<td>1564.0</td>
<td>348.0</td>
<td>5.71</td>
</tr>
</tbody>
</table>

Table 2. Rates of convergence \(\tau\) and \(\dot{\tau}\) of PRINCIPALS to \(\varepsilon\)-PRINCIPALS.

<table>
<thead>
<tr>
<th>(r)</th>
<th>(\tau)</th>
<th>(\dot{\tau})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.060</td>
<td>0.001</td>
</tr>
<tr>
<td>2</td>
<td>0.812</td>
<td>0.667</td>
</tr>
<tr>
<td>3</td>
<td>0.489</td>
<td>0.323</td>
</tr>
<tr>
<td>4</td>
<td>0.466</td>
<td>0.257</td>
</tr>
<tr>
<td>5</td>
<td>0.493</td>
<td>0.388</td>
</tr>
<tr>
<td>6</td>
<td>0.576</td>
<td>0.332</td>
</tr>
<tr>
<td>7</td>
<td>0.473</td>
<td>0.372</td>
</tr>
<tr>
<td>8</td>
<td>0.659</td>
<td>0.553</td>
</tr>
<tr>
<td>9</td>
<td>0.645</td>
<td>0.494</td>
</tr>
<tr>
<td>10</td>
<td>0.678</td>
<td>0.537</td>
</tr>
<tr>
<td>11</td>
<td>0.592</td>
<td>0.473</td>
</tr>
<tr>
<td>12</td>
<td>0.648</td>
<td>0.465</td>
</tr>
</tbody>
</table>

Table 2. Rates of convergence \(\tau\) and \(\dot{\tau}\) of PRINCIPALS to \(\varepsilon\)-PRINCIPALS.

Acceleration of Convergence of the Alternating Least Squares Algorithm for Nonlinear Principal Components Analysis

We introduce the result of studies of convergence of \(\varepsilon\)-PRINCIPALS from Kuroda et al. [Kuroda et al., 2011]. The data set used in the experiments is obtained in teacher evaluation by students and consists of 56 observations on 13 variables with 5 levels each; the lowest evaluation level is 1 and the highest 5.

The rates of convergence of these algorithms are assessed as

\[
\tau = \lim_{t \to \infty} \frac{||X^*(t) - X^*(t-1)||}{||X^*(t-1) - X^*(t-2)||}
\]

for PRINCIPALS,

\[
\dot{\tau} = \lim_{t \to \infty} \frac{||\dot{X}^*(t) - \dot{X}^*(t-1)||}{||\dot{X}^*(t-1) - \dot{X}^*(t-2)||}
\]

for \(\varepsilon\)-PRINCIPALS.

If the inequality \(0 < \dot{\tau} < \tau < 1\) holds, we say that \(\{\dot{X}^*(t)\}_{t \geq 0}\) converges faster than \(\{X^*(t)\}_{t \geq 0}\). Table 2 provides the rates of convergence \(\tau\) and \(\dot{\tau}\) for each \(r\). We see from the table that \(\{\dot{X}^*(t)\}_{t \geq 0}\) converges faster than \(\{X^*(t)\}_{t \geq 0}\) in comparison between \(\tau\) and \(\dot{\tau}\) for each \(r\) and thus conclude that \(\varepsilon\)-PRINCIPALS significantly improves the rate of convergence of PRINCIPALS. The speed of convergence of \(\varepsilon\)-PRINCIPALS is investigated by

\[
\dot{\rho} = \lim_{t \to \infty} \frac{||\dot{X}^*(t) - X^*(\infty)||}{||X^*(t+2) - X^*(\infty)||} = 0.
\]
If \( \{\dot{X}^*(t)\}_{t \geq 0} \) converges to the same limit point \( X^*(\infty) \) as \( \{X^*(t)\}_{t \geq 0} \) and Equation (7) holds, we say that \( \{\dot{X}^*(t)\}_{t \geq 0} \) accelerates the convergence of \( \{X^*(t)\}_{t \geq 0} \). See Brezinski and Zaglia [Brezinski and Zaglia, 1991]. In the experiments, \( \{\dot{X}^*(t)\}_{t \geq 0} \) converges to the final value of \( \{X^*(t)\}_{t \geq 0} \) and \( \dot{\rho} \) is reduced to zero for all \( r \). We see from the results that \( \psi \)-PRINCIPALS accelerates the convergence of \( \{X^*(t)\} \).

5. Variable selection in nonlinear PCA: Modified PCA approach

In the analysis of data with large numbers of variables, a common objective is to reduce the dimensionality of the data set. PCA is a popular dimension-reducing tool that replaces the variables in the data set by a smaller number of derived variables. However, for example, in PCA of a data set with a large number of variables, the result may not be easy to interpret. One way to give a simple interpretation of principal components is to select a subset of variables that best approximates all the variables. Various variable selection criteria in PCA have been proposed by Jolliffe [Jolliffe, 1972], McCabe [McCabe, 1984], Robert and Escoufier [Robert and Escoufier, 1976], Krzanowski [Krzanowski, 1987]. Al-Kandari et al. [Al-Kandari et al., 2001; Al-Kandari et al., 2005] gave guidelines as to the types of data for which each variable selection procedure is useful. Cadima et al. [Cadima et al., 2004] reported computational experiments carried out with several heuristic algorithms for the optimization problems resulting from the variable selection criteria in PCA found in the above literature.

Tanaka and Mori [Tanaka and Mori, 1997] proposed modified PCA (M.PCA) for deriving principal components which are computed by using only a selected subset of variables but which represent all the variables including those not selected. Since M.PCA includes variable selection procedures in the analysis, its criteria can be used directly to find a reasonable subset of variables. Mori et al. [Mori et al., 1998; Mori et al., 2006] extended M.PCA to qualitative data and provided variable selection procedures, in which the ASL algorithm is utilized.

5.1 Formulation of modified PCA

M.PCA derives principal components which are computed as linear combinations of a subset of variables but which can reproduce all the variables very well. Let \( X \) be decomposed into an \( n \times q \) submatrix \( X_{V_1} \) and an \( n \times (p - q) \) remaining submatrix \( X_{V_2} \). Then M.PCA finds \( r \) linear combinations \( Z = X_{V_1} A \). The matrix \( A \) consists of the eigenvectors associated with the largest \( r \) eigenvalues \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \) and is obtained by solving the eigenvalue problem:

\[
[(S_{11}^2 + S_{12} S_{21}) - DS_{11}] A = 0,
\]

where \( S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \) is the covariance matrix of \( X = (X_{V_1}, X_{V_2}) \) and \( D \) is a \( q \times q \) diagonal matrix of eigenvalues. A best subset of \( q \) variables has the largest value of the proportion \( P = \sum_{j=1}^{r} \lambda_j / \text{tr}(S) \) or the RV-coefficient \( RV = \left\{ \sum_{j=1}^{r} \lambda_j^2 / \text{tr}(S^2) \right\}^{1/2} \). Here we use \( P \) as variable selection criteria.

5.2 Variable selection procedures

In order to find a subset of \( q \) variables, we employ Backward elimination and Forward selection of Mori et al. [Mori et al., 1998; Mori et al., 2006] as cost-saving stepwise selection procedures in which only one variable is removed or added sequentially.
[Backward elimination]

**Stage A:** *Initial fixed-variables stage*

A-1 Assign $q$ variables to subset $X_{V_1}$, usually $q := p$.

A-2 Solve the eigenvalue problem (8).

A-3 Look carefully at the eigenvalues, determine the number $r$ of principal components to be used.

A-4 Specify kernel variables which should be involved in $X_{V_1}$, if necessary. The number of kernel variables is less than $q$.

**Stage B:** *Variable selection stage (Backward)*

B-1 Remove one variable from among $q$ variables in $X_{V_1}$, make a temporary subset of size $q - 1$, and compute $P$ based on the subset. Repeat this for each variable in $X_{V_1}$, then obtain $q$ values on $P$. Find the best subset of size $q - 1$ which provides the largest $P$ among these $q$ values and remove the corresponding variable from the present $X_{V_1}$. Put $q := q - 1$.

B-2 If $P$ or $q$ is larger than preassigned values, go to B-1. Otherwise stop.

[Forward selection]

**Stage A:** *Initial fixed-variables stage*

A-1 $\sim$ A-3 Same as A-1 to A-3 in Backward elimination.

A-4 Redefine $q$ as the number of kernel variables (here, $q \geq r$). If you have kernel variables, assign them to $X_{V_1}$. If not, put $q := r$, find the best subset of $q$ variables which provides the largest $P$ among all possible subsets of size $q$ and assign it to $X_{V_1}$.

**Stage B:** *Variable selection stage (Forward)*

B-1 Adding one of the $p - q$ variables in $X_{V_2}$ to $X_{V_1}$, make a temporary subset of size $q + 1$ and obtain $P$. Repeat this for each variable in $X_{V_2}$, then obtain $p - q$ $P$s. Find the best subset of size $q + 1$ which provides the largest (or smallest) $P$ among the $p - q$ $P$s and add the corresponding variable to the present subset of $X_{V_1}$. Put $q := q + 1$.

B-2 If the $P$ or $q$ are smaller (or larger) than preassigned values, go back to B-1. Otherwise stop.

In Backward elimination, to find the best subset of $q - 1$ variables, we perform M.PCA for each of $q$ possible subsets of the $q - 1$ variables among $q$ variables selected in the previous selection step. The total number of estimations for M.PCA from $q = p - 1$ to $q = r$ is therefore large, i.e., $p + (p - 1) + \cdots + (r + 1) = (p - r)(p + r + 1)/2$. In Forward selection, the total number of estimations for M.PCA from $q = r$ to $q = p - 1$ is $pC_r + (p - r) + (p - (r + 1)) + \cdots + 2 = pC_r + (p - r - 1)(p - r + 2)/2$.

**Numerical experiments 3: Variable selection in M.PCA for simulated data**

We apply PRINCIPALS and ve-PRINCIPALS to variable selection in M.PCA of qualitative data using simulated data consisting of 100 observations on 10 variables with 3 levels.

Table 3 shows the number of iterations and CPU time taken by two algorithms for finding a subset of $q$ variables based on $3 (= r)$ principal components. The values of the second to fifth columns in the table indicate that the number of iterations of PRINCIPALS is very large and a long computation time is taken for convergence, while ve-PRINCIPALS converges considerably faster than PRINCIPALS. We can see from the sixth and seventh columns in the
table that \( v \varepsilon \)-PRINCIPALS requires the number of iterations 3 - 5 times smaller and CPU time 2 - 5 times shorter than \( v \varepsilon \)-PRINCIPALS. In particular, the \( v \varepsilon \) acceleration effectively works to accelerate the convergence of \( \{X^*(t)\}_{t \geq 0} \) for the larger number of iterations of PRINCIPALS.

(a) Backward elimination

<table>
<thead>
<tr>
<th>( q )</th>
<th>PRINCIPALS</th>
<th>( v \varepsilon )-PRINCIPALS</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>iteration</td>
<td>CPU time</td>
<td>iteration</td>
<td>CPU time</td>
</tr>
<tr>
<td>10</td>
<td>141</td>
<td>1.70</td>
<td>48</td>
</tr>
<tr>
<td>9</td>
<td>1,363</td>
<td>17.40</td>
<td>438</td>
</tr>
<tr>
<td>8</td>
<td>1,620</td>
<td>20.19</td>
<td>400</td>
</tr>
<tr>
<td>7</td>
<td>1,348</td>
<td>16.81</td>
<td>309</td>
</tr>
<tr>
<td>6</td>
<td>4,542</td>
<td>53.72</td>
<td>869</td>
</tr>
<tr>
<td>5</td>
<td>13,735</td>
<td>159.72</td>
<td>2,949</td>
</tr>
<tr>
<td>4</td>
<td>41,759</td>
<td>482.59</td>
<td>12,521</td>
</tr>
<tr>
<td>3</td>
<td>124</td>
<td>1.98</td>
<td>44</td>
</tr>
<tr>
<td>Total</td>
<td>64,491</td>
<td>752.40</td>
<td>17,530</td>
</tr>
</tbody>
</table>

(b) Forward selection

<table>
<thead>
<tr>
<th>( q )</th>
<th>PRINCIPALS</th>
<th>( v \varepsilon )-PRINCIPALS</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>iteration</td>
<td>CPU time</td>
<td>iteration</td>
<td>CPU time</td>
</tr>
<tr>
<td>3</td>
<td>4,382</td>
<td>67.11</td>
<td>1442</td>
</tr>
<tr>
<td>4</td>
<td>154,743</td>
<td>1,786.70</td>
<td>26,091</td>
</tr>
<tr>
<td>5</td>
<td>13,123</td>
<td>152.72</td>
<td>3,198</td>
</tr>
<tr>
<td>6</td>
<td>3,989</td>
<td>47.02</td>
<td>1,143</td>
</tr>
<tr>
<td>7</td>
<td>1,264</td>
<td>15.27</td>
<td>300</td>
</tr>
<tr>
<td>8</td>
<td>340</td>
<td>4.38</td>
<td>108</td>
</tr>
<tr>
<td>9</td>
<td>267</td>
<td>3.42</td>
<td>75</td>
</tr>
<tr>
<td>10</td>
<td>141</td>
<td>1.73</td>
<td>48</td>
</tr>
<tr>
<td>Total</td>
<td>178,249</td>
<td>2,078.33</td>
<td>32,405</td>
</tr>
</tbody>
</table>

Table 3. The numbers of iterations and CPU times of PRINCIPALS and \( v \varepsilon \)-PRINCIPALS and their speed-ups in application to variable selection for finding a subset of \( q \) variables using simulated data.

The last row in Table 3 shows the total number of iterations and total CPU time for selecting 8 subsets for \( q = 3, \ldots, 10 \). When searching the best subset for each \( q \), PRINCIPALS requires 64,491 iterations in Backward elimination and 178,249 iterations in Forward selection, while \( v \varepsilon \)-PRINCIPALS finds the subsets after 17,530 and 32,405 iterations, respectively. These values show that the computation times by \( v \varepsilon \)-PRINCIPALS are reduced to only 28\% (= 1/3.52) and 19\% (= 1/5.16) of those of ordinary PRINCIPALS. The iteration and CPU time speed-ups given in the sixth and seventh columns of the table demonstrate that the \( v \varepsilon \) acceleration works well to speed up the convergence of \( \{X^*(t)\}_{t \geq 0} \) and consequently results in greatly reduced computation times in variable selection problems.

**Numerical experiments 4: Variable selection in M.PCA for real data**

We consider the variable selection problems in M.PCA of qualitative data to mild distribution of consciousness (MDOC) data from Sano et al. [Sano et al. 1977]. MDOC is the data matrix of 87 individuals on 23 variables with 4 levels. In the variable selection problem, we select a suitable subset based on 2 \((= r)\) principal components.

Table 4 summarizes the results of variable selection using Backward elimination and Forward selection procedures for finding a subset of \( q \) variables. We see from the last row of the table
### Table 4

The numbers of iterations and CPU times of PRINCIPALS and \( \ve^{-}\)PRINCIPALS, their speed-ups and \( P \) in application to variable selection for finding a subset of \( q \) variables using MDOC.

(a) Backward elimination

<table>
<thead>
<tr>
<th>( q )</th>
<th>PRINCIPALS Iteration</th>
<th>CPU time</th>
<th>( \ve^{-})PRINCIPALS Iteration</th>
<th>CPU time</th>
<th>Speed-up Iteration</th>
<th>CPU time</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>36</td>
<td>1.39</td>
<td>10</td>
<td>0.65</td>
<td>3.60</td>
<td>2.13</td>
<td>0.694</td>
</tr>
<tr>
<td>22</td>
<td>819</td>
<td>32.42</td>
<td>231</td>
<td>15.40</td>
<td>3.55</td>
<td>2.11</td>
<td>0.694</td>
</tr>
<tr>
<td>21</td>
<td>779</td>
<td>30.79</td>
<td>221</td>
<td>14.70</td>
<td>3.52</td>
<td>2.10</td>
<td>0.693</td>
</tr>
<tr>
<td>20</td>
<td>744</td>
<td>29.37</td>
<td>212</td>
<td>14.05</td>
<td>3.51</td>
<td>2.09</td>
<td>0.693</td>
</tr>
<tr>
<td>19</td>
<td>725</td>
<td>28.43</td>
<td>203</td>
<td>13.41</td>
<td>3.57</td>
<td>2.12</td>
<td>0.692</td>
</tr>
<tr>
<td>18</td>
<td>705</td>
<td>27.45</td>
<td>195</td>
<td>12.77</td>
<td>3.62</td>
<td>2.15</td>
<td>0.692</td>
</tr>
<tr>
<td>17</td>
<td>690</td>
<td>26.67</td>
<td>189</td>
<td>12.25</td>
<td>3.65</td>
<td>2.18</td>
<td>0.691</td>
</tr>
<tr>
<td>16</td>
<td>671</td>
<td>25.73</td>
<td>180</td>
<td>11.61</td>
<td>3.73</td>
<td>2.22</td>
<td>0.690</td>
</tr>
<tr>
<td>15</td>
<td>633</td>
<td>24.26</td>
<td>169</td>
<td>10.85</td>
<td>3.75</td>
<td>2.24</td>
<td>0.689</td>
</tr>
<tr>
<td>14</td>
<td>565</td>
<td>21.79</td>
<td>153</td>
<td>10.02</td>
<td>3.69</td>
<td>2.17</td>
<td>0.688</td>
</tr>
<tr>
<td>13</td>
<td>540</td>
<td>20.69</td>
<td>147</td>
<td>9.48</td>
<td>3.67</td>
<td>2.18</td>
<td>0.687</td>
</tr>
<tr>
<td>12</td>
<td>498</td>
<td>19.09</td>
<td>132</td>
<td>8.64</td>
<td>3.77</td>
<td>2.21</td>
<td>0.686</td>
</tr>
<tr>
<td>11</td>
<td>451</td>
<td>17.34</td>
<td>121</td>
<td>7.95</td>
<td>3.73</td>
<td>2.18</td>
<td>0.684</td>
</tr>
<tr>
<td>10</td>
<td>427</td>
<td>16.29</td>
<td>117</td>
<td>7.46</td>
<td>3.65</td>
<td>2.18</td>
<td>0.682</td>
</tr>
<tr>
<td>9</td>
<td>459</td>
<td>16.99</td>
<td>115</td>
<td>7.05</td>
<td>3.99</td>
<td>2.41</td>
<td>0.679</td>
</tr>
<tr>
<td>8</td>
<td>419</td>
<td>15.43</td>
<td>106</td>
<td>6.42</td>
<td>3.95</td>
<td>2.40</td>
<td>0.676</td>
</tr>
<tr>
<td>7</td>
<td>382</td>
<td>14.02</td>
<td>100</td>
<td>5.89</td>
<td>3.82</td>
<td>2.38</td>
<td>0.673</td>
</tr>
<tr>
<td>6</td>
<td>375</td>
<td>13.51</td>
<td>96</td>
<td>5.41</td>
<td>3.91</td>
<td>2.50</td>
<td>0.669</td>
</tr>
<tr>
<td>5</td>
<td>355</td>
<td>12.58</td>
<td>95</td>
<td>5.05</td>
<td>3.74</td>
<td>2.49</td>
<td>0.661</td>
</tr>
<tr>
<td>4</td>
<td>480</td>
<td>16.11</td>
<td>117</td>
<td>5.33</td>
<td>4.10</td>
<td>3.02</td>
<td>0.648</td>
</tr>
<tr>
<td>3</td>
<td>2,793</td>
<td>86.55</td>
<td>1,354</td>
<td>43.48</td>
<td>2.06</td>
<td>1.99</td>
<td>0.620</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>1.92</td>
<td>10</td>
<td>1.34</td>
<td>3.50</td>
<td>1.43</td>
<td>0.581</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>13,581</td>
<td>498.82</td>
<td>4,273</td>
<td>229.20</td>
<td>3.18</td>
<td>2.18</td>
</tr>
</tbody>
</table>

(b) Forward selection

<table>
<thead>
<tr>
<th>( q )</th>
<th>PRINCIPALS Iteration</th>
<th>CPU time</th>
<th>( \ve^{-})PRINCIPALS Iteration</th>
<th>CPU time</th>
<th>Speed-up Iteration</th>
<th>CPU time</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3,442</td>
<td>176.76</td>
<td>1,026</td>
<td>119.07</td>
<td>3.35</td>
<td>1.48</td>
<td>0.597</td>
</tr>
<tr>
<td>3</td>
<td>5,389</td>
<td>170.82</td>
<td>1,189</td>
<td>44.28</td>
<td>4.53</td>
<td>3.86</td>
<td>0.633</td>
</tr>
<tr>
<td>4</td>
<td>1,804</td>
<td>60.96</td>
<td>429</td>
<td>20.27</td>
<td>4.21</td>
<td>3.01</td>
<td>0.650</td>
</tr>
<tr>
<td>5</td>
<td>1,406</td>
<td>48.53</td>
<td>349</td>
<td>17.41</td>
<td>4.03</td>
<td>2.79</td>
<td>0.662</td>
</tr>
<tr>
<td>6</td>
<td>1,243</td>
<td>43.25</td>
<td>305</td>
<td>15.75</td>
<td>4.08</td>
<td>2.75</td>
<td>0.668</td>
</tr>
<tr>
<td>7</td>
<td>1,114</td>
<td>39.03</td>
<td>278</td>
<td>14.61</td>
<td>4.01</td>
<td>2.67</td>
<td>0.674</td>
</tr>
<tr>
<td>8</td>
<td>871</td>
<td>31.35</td>
<td>221</td>
<td>12.39</td>
<td>3.94</td>
<td>2.53</td>
<td>0.677</td>
</tr>
<tr>
<td>9</td>
<td>789</td>
<td>28.57</td>
<td>202</td>
<td>11.52</td>
<td>3.91</td>
<td>2.48</td>
<td>0.680</td>
</tr>
<tr>
<td>10</td>
<td>724</td>
<td>26.32</td>
<td>187</td>
<td>10.74</td>
<td>3.87</td>
<td>2.45</td>
<td>0.683</td>
</tr>
<tr>
<td>11</td>
<td>647</td>
<td>23.69</td>
<td>156</td>
<td>9.39</td>
<td>4.15</td>
<td>2.52</td>
<td>0.685</td>
</tr>
<tr>
<td>12</td>
<td>578</td>
<td>21.30</td>
<td>142</td>
<td>8.60</td>
<td>4.07</td>
<td>2.48</td>
<td>0.687</td>
</tr>
<tr>
<td>13</td>
<td>492</td>
<td>18.39</td>
<td>125</td>
<td>7.76</td>
<td>3.94</td>
<td>2.37</td>
<td>0.688</td>
</tr>
<tr>
<td>14</td>
<td>432</td>
<td>16.23</td>
<td>110</td>
<td>6.94</td>
<td>3.93</td>
<td>2.34</td>
<td>0.689</td>
</tr>
<tr>
<td>15</td>
<td>365</td>
<td>13.91</td>
<td>95</td>
<td>6.13</td>
<td>3.84</td>
<td>2.27</td>
<td>0.690</td>
</tr>
<tr>
<td>16</td>
<td>306</td>
<td>11.80</td>
<td>80</td>
<td>5.30</td>
<td>3.83</td>
<td>2.22</td>
<td>0.691</td>
</tr>
<tr>
<td>17</td>
<td>267</td>
<td>10.32</td>
<td>71</td>
<td>4.66</td>
<td>3.76</td>
<td>2.21</td>
<td>0.691</td>
</tr>
<tr>
<td>18</td>
<td>226</td>
<td>8.77</td>
<td>60</td>
<td>3.96</td>
<td>3.77</td>
<td>2.21</td>
<td>0.692</td>
</tr>
<tr>
<td>19</td>
<td>193</td>
<td>7.48</td>
<td>51</td>
<td>3.39</td>
<td>3.78</td>
<td>2.21</td>
<td>0.692</td>
</tr>
<tr>
<td>20</td>
<td>152</td>
<td>5.91</td>
<td>40</td>
<td>2.65</td>
<td>3.80</td>
<td>2.23</td>
<td>0.693</td>
</tr>
<tr>
<td>21</td>
<td>108</td>
<td>4.26</td>
<td>30</td>
<td>2.00</td>
<td>3.60</td>
<td>2.13</td>
<td>0.693</td>
</tr>
<tr>
<td>22</td>
<td>72</td>
<td>2.85</td>
<td>20</td>
<td>1.33</td>
<td>3.60</td>
<td>2.14</td>
<td>0.694</td>
</tr>
<tr>
<td>23</td>
<td>36</td>
<td>1.39</td>
<td>10</td>
<td>0.66</td>
<td>3.60</td>
<td>2.11</td>
<td>0.694</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>20,686</td>
<td>771.88</td>
<td>5,176</td>
<td>328.81</td>
<td>3.99</td>
<td>2.35</td>
</tr>
</tbody>
</table>
that the iteration speed-ups are 3.18 in Backward elimination and 3.99 in Forward selection and thus \( ve \)-PRINCIPALS well accelerates the convergence of \( \{X^*(t)\}_{t \geq 0} \). The CPU time speed-ups are 2.18 in Backward elimination and 2.35 in Forward selection, and are not as large as the iteration speed-ups. The computation time per iteration of \( ve \)-PRINCIPALS is greater than that of PRINCIPALS due to computation of the Acceleration step. Therefore, for the smaller number of iterations, the CPU time of \( ve \)-PRINCIPALS is almost same as or may be longer than that of PRINCIPALS. For example, in Forward selection for \( q = 2 \), PRINCIPALS converges in almost cases after less than 15 iterations and then the CPU time speed-up is 1.48.

The proportion \( P \) in the eighth column of the table indicates the variation explained by the first 2 principal components for the selected \( q \) variables. Iizuka et al. [Iizuka et al., 2003] selected the subset of 6 variables found by either procedures as a best subset, since \( P \) slightly changes until \( q = 6 \) in Backward elimination and after \( q = 6 \) in Forward selection.

6. Concluding remarks

In this paper, we presented \( ve \)-PRINCIPALS that accelerates the convergence of PRINCIPALS by using the \( ve \) algorithm. The algorithm generates the \( ve \) accelerated sequence \( \{\dot{X}^*(t)\} \) using \( \{X^*(t)\}_{t \geq 0} \) but it does not modify the estimation equations in PRINCIPALS. Therefore the algorithm enables an acceleration of the convergence of PRINCIPALS, while still preserving the stable convergence property of PRINCIPALS. The \( ve \) algorithm in itself is a fairly simple computational procedure and, at each iteration, it requires only \( O(np) \) arithmetic operations. For each iteration, the computational complexity of the \( ve \) algorithm may be less expensive than that for computing a matrix inversion and for solving the eigenvalue problem in PRINCIPALS.

The most appealing points of the \( ve \) algorithm are that, if an original sequence converges to a limit point then the accelerated sequence converges to the same limit point as the original sequence and its speed of convergence is faster than the original sequence. In all the numerical experiments, the \( ve \) accelerated sequence \( \{\dot{X}^*(t)\}_{t \geq 0} \) converges to the final value of \( \{X^*(t)\}_{t \geq 0} \) after the significantly fewer number of iterations than that of PRINCIPALS.

The numerical experiments employing simulated data in Section 4 demonstrated that \( ve \) acceleration for PRINCIPALS significantly speeds up the convergence of \( \{X^*(t)\}_{t \geq 0} \) in terms of the number of iterations and the computation time. In particular, the \( ve \) acceleration effectively works to speed up the convergence for the larger number of iterations of PRINCIPALS. Furthermore, we evaluate the performance of the \( ve \) acceleration for PRINCIPALS by applying to variable selection in M.PCA of qualitative data. Numerical experiments using simulated and real data showed that \( ve \)-PRINCIPALS improves the speed of convergence of ordinary PRINCIPALS and enables greatly the reduction of computation times in the variable selection for finding a suitable variable set using Backward elimination and Forward selection procedures. The results indicate that the \( ve \) acceleration well works in saving the computational time in variable selection problems.

The computations of variable selection in M.PCA of qualitative data are partially performed by the statistical package VASpca(VAriable Selection in principal component analysis) that was developed by Mori, Iizuka, Tarumi and Tanaka in 1999 and can be obtained from Mori’s website in Appendix C. We will provide VASpca using \( ve \)-PRINCIPALS as the iterative algorithm for PCA and M.PCA of qualitative data.
7. Acknowledgment

The authors would like to thank the editor and two referees whose valuable comments and kind suggestions that led to an improvement of this paper. This research is supported by the Japan Society for the Promotion of Science (JSPS), Grant-in-Aid for Scientific Research (C), No 20500263.

8. Appendix A: PRINCALS

PRINCALS by Gifi [Gifi, 1990] can handle multiple nominal variables in addition to the single nominal, ordinal and numerical variables accepted in PRINCALS. We denote the set of multiple variables by \( \mathcal{J}_M \) and the set of single variables with single nominal and ordinal scales and numerical measurements by \( \mathcal{J}_S \). For \( \mathbf{X} \) consisting of a mixture of multiple and single variables, the algorithm alternates between estimation of \( \mathbf{Z}, \mathbf{A} \) and \( \mathbf{X}^* \) subject to minimizing

\[
\mathbf{A}^* = \text{argmin} \left( \mathbf{Z} - \mathbf{X}^* \mathbf{A} \right)\top (\mathbf{Z} - \mathbf{X}^* \mathbf{A})
\]

under the restriction

\[
\mathbf{Z}\top \mathbf{1}_n = 0, \quad \text{and} \quad \mathbf{Z}\top \mathbf{Z} = n\mathbf{I}_p. \tag{9}
\]

For the initialization of PRINCALS, we determine initial data \( \mathbf{Z}^{(0)}, \mathbf{A}^{(0)} \) and \( \mathbf{X}^{* (0)} \). The values of \( \mathbf{Z}^{(0)} \) are initialized with random numbers under the restriction (9). For \( j \in \mathcal{J}_M \), the initial value of \( \mathbf{X}_j^* \) is obtained by

\[
\mathbf{X}_j^{* (0)} = \mathbf{G}_j (\mathbf{G}_j\top \mathbf{G}_j)^{-1} \mathbf{G}_j\top \mathbf{Z}^{(0)}.
\]

For \( j \in \mathcal{J}_S \), \( \mathbf{X}_j^{* (0)} \) is defined as the first \( K_j \) successive integers under the normalization restriction, and the initial value of \( \mathbf{A}_j \) is calculated as the vector \( \mathbf{A}_j^{(0)} = \mathbf{Z}^{(0)}\top \mathbf{X}_j^{* (0)} \). Given these initial values, PRINCALS as provided in Michailidis and de Leeuw [Michailidis and Leeuw, 1998] iterates the following two steps:

- **Model parameter estimation step**: Calculate \( \mathbf{Z}^{(t+1)} \) by

\[
\mathbf{Z}^{(t+1)} = \mathbf{p}^{-1} \left( \sum_{j \in \mathcal{J}_M} \mathbf{X}_j^{*(t)} + \sum_{j \in \mathcal{J}_S} \mathbf{X}_j^{*(t)} \mathbf{A}_j^{(t)} \right).
\]

Columnwise center and orthonormalize \( \mathbf{Z}^{(t+1)} \). Estimate \( \mathbf{A}_j^{(t+1)} \) for the single variable \( j \) by

\[
\mathbf{A}_j^{(t+1)} = \mathbf{Z}^{(t+1)}\top \mathbf{X}_j^{*(t)} / \mathbf{X}_j^{*(t)}\top \mathbf{X}_j^{*(t)}.
\]

- **Optimal scaling step**: Estimate the optimally scaled vector for \( j \in \mathcal{J}_M \) by

\[
\mathbf{X}_j^{*(t+1)} = \mathbf{G}_j (\mathbf{G}_j\top \mathbf{G}_j)^{-1} \mathbf{G}_j\top \mathbf{Z}^{(t+1)}.
\]

and for \( j \in \mathcal{J}_S \) by

\[
\mathbf{X}_j^{*(t+1)} = \mathbf{G}_j (\mathbf{G}_j\top \mathbf{G}_j)^{-1} \mathbf{G}_j\top \mathbf{Z}^{(t+1)} \mathbf{A}_j^{(t+1)} / \mathbf{A}_j^{(t+1)}\top \mathbf{A}_j^{(t+1)}
\]

under measurement restrictions on each of the variables.
9. Appendix B: The \( v \varepsilon \) algorithm

Let \( Y(t) \) denote a vector of dimensionality \( d \) that converges to a vector \( Y(\infty) \) as \( t \to \infty \). Let the inverse \( |Y|^{-1} \) of a vector \( Y \) be defined by

\[
|Y|^{-1} = \frac{Y}{\|Y\|^2},
\]

where \( \|Y\| \) is the Euclidean norm of \( Y \).

In general, the \( v \varepsilon \) algorithm for a sequence \( \{Y(t)\}_{t \geq 0} \) starts with

\[
\varepsilon(t, -1) = 0, \quad \varepsilon(t, 0) = Y(t),
\]

and then generates a vector \( \varepsilon(t, k+1) \) by

\[
\varepsilon(t, k+1) = \varepsilon(t+1, k-1) + \left[ \varepsilon(t+1, k) - \varepsilon(t, k) \right]^{-1}, \quad k = 0, 1, 2, \ldots . \tag{10}
\]

For practical implementation, we apply the \( v \varepsilon \) algorithm for \( k = 1 \) to accelerate the convergence of \( \{Y(t)\}_{t \geq 0} \). From Equation (10), we have

\[
\varepsilon(t, 2) = \varepsilon(t+1, 0) + \left[ \varepsilon(t+1, 1) - \varepsilon(t, 1) \right]^{-1} \quad \text{for } k = 1,
\]

\[
\varepsilon(t, 1) = \varepsilon(t+1, -1) + \left[ \varepsilon(t+1, 0) - \varepsilon(t, 0) \right]^{-1} = \left[ \varepsilon(t+1, 0) - \varepsilon(t, 0) \right]^{-1} \quad \text{for } k = 0.
\]

Then the vector \( \varepsilon(t, 2) \) becomes as follows:

\[
\varepsilon(t, 2) = \varepsilon(t+1, 0) + \left[ \varepsilon(t, 0) - \varepsilon(t+1, 0) \right]^{-1} + \left[ \varepsilon(t+2, 0) - \varepsilon(t+1, 0) \right]^{-1}
\]

\[
= Y(t+1) + \left[ Y(t) - Y(t+1) \right]^{-1} + \left[ Y(t+2) - Y(t+1) \right]^{-1}. \tag{10}
\]

10. Appendix C: VASpca

URL of VASpca
http://mo161.soci.ous.ac.jp/vaspca/indexE.html

11. References


This book is aimed at raising awareness of researchers, scientists and engineers on the benefits of Principal Component Analysis (PCA) in data analysis. In this book, the reader will find the applications of PCA in fields such as image processing, biometric, face recognition and speech processing. It also includes the core concepts and the state-of-the-art methods in data analysis and feature extraction.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following: