ELSEVIER

Chemometrics and Intelligent Laboratory Systems 36 (1997) 165–172

Chemometrics and intelligent laboratory systems

The kernel PCA algorithms for wide data. Part I: theory and algorithms

W. Wu [a], D.L. Massart [a,*], S. de Jong [b]

[a] *ChemoAC, Pharmaceutical Institute, Vrije Universiteit Brussel, Laarbeeklaan 103, B-1090 Brussels, Belgium*
[b] *Unilever Research Laboratorium Vlaardingen, Olivier van Noortlaan 120, 3133 AT Vlaardingen, The Netherlands*

Received 28 October 1996; revised 12 December 1996; accepted 16 December 1996

Abstract

Four classic PCA algorithms: NIPALS, the power method (POWER), singular value decomposition (SVD) and eigenvalue decomposition (EVD) are modified into their kernel version to analyse wide data sets. For such data sets with many variables and fewer objects, the classic algorithms become very inefficient, because the size of the associated matrix $\mathbf{X}' \cdot \mathbf{X}$ ($p \times p$) is very large. Based on the kernel matrix $\mathbf{X} \cdot \mathbf{X}'$ ($n \times n$), the kernel algorithms are developed. They yield the same principal components but, when the number of variables is higher than the number of objects, they are faster than the corresponding classic algorithms. Simulation results confirm this property and also show that the kernel EVD is the most efficient algorithm for wide data sets, and that the difference between the kernel SVD and EVD is not very large. Therefore, it is recommended to use the kernel EVD for wide data sets whether all PCs or only the first few are required. A Matlab PCA program which efficiently deals with all kinds of data sets was developed.

Keywords: PCA algorithm; Kernel; Wide data

1. Introduction

Principal component analysis (PCA) is one of the most important techniques in multivariate data analysis [1–9]. It is often applied to analyse multivariate data, such as to visualise the data structure, to detect outliers and to reduce the data dimensionality [10–15], and many important methods are based on PCA. For instance, the well-known PCR and SIMCA method are derived from PCA [1,16–20].

The development of new analytical chemical techniques has made it very easy and quick for the analyst to obtain hundreds or even thousands of measurements (variables) from one sample (object). For example, there are more than 500 wavelengths in NIR spectral data [21] and 10,000 variables in the data set from 3D-QSAR [22]. To analyse such wide data, PCA is a commonly used method because PCA deals with the data collinearity naturally. However, when PCA is applied to such large data sets, the speed of calculation can be slow and a large memory space of the computer is needed. The situation is even worse when cross-validation is used to evaluate methods such as PCR, because the PCA is performed several times to obtain the results. In previous work [21], PCA was

* Corresponding author.

PII S0169-7439(97)00010-5

applied to reduce the number of variables for NIR data, and the PC factors were used as the input variables of classifiers such as linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and regularised discriminant analysis (RDA). When leave-one-out full validation is used, several hours of intensive calculation are needed to obtain the results of the classifier because the PCA is repeated hundreds of times. Therefore, there is a need to modify the PCA algorithms to accelerate the computation speed. In this paper, four classical PCA algorithms, namely singular value decomposition (SVD), eigenvalue decomposition (EVD), NIPALS and the power method (POWER), are modified to their so-called kernel versions to efficiently deal with wide data sets. The idea of using kernel versions is not new. For instance, it is incorporated in Wise's PLS toolbox [23] and in the programs published by Karjalainen et al. [24]. However, the utility of kernel versions does not seem evident to many chemometricians, since many standard software products do not utilise them. In this article, explicit attention is given to the problem and the effect of kernel algorithms is demonstrated by a comparison.

2. Theory

2.1. Notation

$\mathbf{X}$	$n \times p$, a data matrix with n row objects and p column variables
$\mathbf{X}_{new}$	$m \times p$, a data matrix with m row new objects and p column variables
$\mathbf{L}$	$p \times r$, the loading matrix
$\mathbf{S}$	$n \times r$, the score matrix
$\mathbf{S}_{new}$	$m \times r$, the predicted score matrix of the new objects
$\mathbf{U}$	$n \times r$, the left singular vectors matrix or the row eigenvector matrix
$\mathbf{V}$	$p \times r$, the right singular vectors matrix or the column eigenvector matrix
$\mathbf{\Lambda}$	$r \times r$, the diagonal matrix of singular values $\lambda_1, \lambda_2, \ldots, \lambda_r$
$\mathbf{\Gamma}$	$r \times r$, the diagonal matrix of eigenvalues $\gamma_1, \gamma_2, \ldots, \gamma_r$
$\mathbf{C}_p$	$p \times p$, the matrix of cross products of the columns of $\mathbf{X}$, $\mathbf{X}' \cdot \mathbf{X}$
$\mathbf{C}_n$	$n \times n$, the matrix of cross products of the rows of $\mathbf{X}$, $\mathbf{X} \cdot \mathbf{X}'$
$\mathbf{t}$	$n \times 1$, the temporary vector for $\mathbf{u}$
$\mathbf{w}$	$p \times 1$, the temporary vector for $\mathbf{v}$
$\mathbf{P}_{ercent}$,	$r \times 1$, the column vector containing the explained variance percentage of each factor

All Matlab built-in functions are expressed in lower case letters with the `typewriter font`.

2.2. Review of the PCA algorithms

There are several PCA algorithms, and four of them often appear in the literature [1–15], namely NIPALS, and SVD which use the data matrix $\mathbf{X}$, and POWER and EVD which work on the cross product matrix $\mathbf{X}' \cdot \mathbf{X}$. SVD and EVD extract the latent variables (PC factors) simultaneously, while NIPALS and POWER calculate PC factors sequentially.

NIPALS decomposes the data to obtain the latent variables factor by factor, in decreasing order of importance, i.e. according to their eigenvalues or singular values. The algorithm can be interrupted at any number of factors. Therefore, the NIPALS algorithm is very efficient when only the first few factors are required. In PCR and SIMCA, only the first few PC factors are interesting, and when PCA is used to display and compress data, normally also the first few factors are used. Therefore, NIPALS is the most often used algorithm, and often considered as the standard PCA algorithm in the field of chemometrics [3,5,6,9–11,13,14].

NIPALS was first described by Fisher and MacKenzie [9]. Then it was rediscovered and developed by H. Wold. A similar algorithm for obtaining the eigenvectors of symmetric matrices was developed by Hotelling and in the literature [7,31,32], Hotelling's algorithm is called the POWER method. The NIPALS algorithm is more often used than the POWER method in chemometrics. However, the POWER method is faster than NIPALS. Both algorithms are very similar. The difference is that they start from a different matrix to form the residue matrix during the iteration. NIPALS begins with the original data matrix $\mathbf{X}$, while the POWER method starts from a symmetric matrix $\mathbf{X}' \cdot \mathbf{X}$. In fact, the POWER method is a sequential method of EVD, while NIPALS is a sequential method of SVD, be-

cause SVD and EVD also decompose the matrices $\mathbf{X}$ and $\mathbf{X}' \cdot \mathbf{X}$ respectively.

SVD is a non-sequential method, and is more efficient than NIPALS when all factors are required. The algorithm is complicated and it includes the Householder transformation and the QR transformation. In Matlab (Matrix Laboratory) [25], for instance, singular vector decomposition (svd) is a built-in function. Using Matlab code, the SVD-PCA algorithm is even simpler and easier to program than the NIPALS-PCA algorithm. Therefore, there is a tendency that the SVD is becoming more and more popular. Moreover, the SVD algorithm has better numerical properties than the NIPALS algorithm, which becomes important when not only the first few dominant factors are required.

EVD is another non-sequential method. It also includes the Householder transformation and the QR transformation steps. However, when using Matlab, it is also very easy because there is a built-in function to estimate eigenvectors and eigenvalues of a matrix. The relation between SVD and EVD is similar to that between NIPALS and POWER. After the data are column centred, the eigenvalues obtained from EVD correspond to the explained variances of PC factors. Therefore, EVD is often used in textbooks to explain the principle of PCA and to interpret the PC factors [1,2]. It is less used in practice probably because its efficiency is between SVD and NIPALS.

2.3. Kernel PCA algorithm

The meaning of the term 'high dimension' is changing as the time passes. Twenty years ago, high dimension meant a number of variables above 50. Now it can mean that 200 to 50,000 variables are measured and used in a data set. However, the PCA algorithms used today were developed 30 to 60 years ago. These algorithms were originally designed and optimised to analyse data sets which do not have very many variables. In PCA, a crucial step is to obtain the loading matrix $\mathbf{L}$. $\mathbf{L}$ can be further used to obtain the score matrix $\mathbf{S}$ and to calculate the score matrix $\mathbf{S}_{new}$ of new objects

$$\mathbf{S} = \mathbf{X} \cdot \mathbf{L} \quad (1)$$

$$\mathbf{S}_{new} = \mathbf{X}_{new} \cdot \mathbf{L} \quad (2)$$

In the classical algorithms such as POWER method and EVD, $\mathbf{L}$ is directly obtained from the eigenvector matrix $\mathbf{V}$ of the matrix $\mathbf{X}' \cdot \mathbf{X}$ which is proportional to the covariance matrix when $\mathbf{X}$ $(n \times p)$ is column centred. The size of the matrix $\mathbf{X}' \cdot \mathbf{X}$ $(p \times p)$ depends on the number of variables; when the number of variables (p) is large, the size of $\mathbf{X}' \cdot \mathbf{X}$ becomes very large. Therefore, the procedure is modified as follows. The eigenvector matrix $\mathbf{U}$ of $\mathbf{X} \cdot \mathbf{X}'$ is first estimated instead of $\mathbf{X}' \cdot \mathbf{X}$. $\mathbf{L}$ is further calculated from $\mathbf{U}$ as $\mathbf{X}' \cdot \mathbf{U} \cdot \mathbf{\Gamma}^{-1/2}$. This means that the $\mathbf{L}$ matrix is indirectly obtained by eigenvector decomposition instead of directly. The size of $\mathbf{X} \cdot \mathbf{X}'$ is $n \times n$, and it is much lower than that of $\mathbf{X}' \cdot \mathbf{X}$ when $p \gg n$. It is much faster to obtain $\mathbf{U}$ from $\mathbf{X} \cdot \mathbf{X}'$ than $\mathbf{V}$ from $\mathbf{X}' \cdot \mathbf{X}$ and less computer space is needed. The $\mathbf{X} \cdot \mathbf{X}'$ matrix contains exactly the same information as the matrix $\mathbf{X}' \cdot \mathbf{X}$, since they have the same eigenvalues and give exactly the same PCA results [23,24,31]. However, the matrix size is greatly reduced from $p \times p$ to $n \times n$, so that, in the terminology of Wold et al. [22,26–29], the matrix $\mathbf{X} \cdot \mathbf{X}'$ can be considered as a kernel matrix of the $\mathbf{X}$ matrix. Therefore, the modified method is designated as the kernel algorithm. This kernel algorithm can be used to modify the four classic PCA algorithms for wide data. The principle behind this kernel algorithm is very similar to that of the kernel PLS algorithm, which is to work with a small matrix as much as possible [22,26–29].

The principle of kernel PCA is very similar to Q-mode PCA which is designed to analyse the row variables (Q-mode) instead of the column variables (R-mode) [7]. However, as the modern approach is to consider both analyses as dual and to unify the two views (of rows and columns) into a single display (biplot), Q-mode analysis is gradually disappearing. It is only shortly mentioned in books by Malinowski [4], Lewi [7] and Martens and Næs [5]. When these books were written, data sets normally had a limited number of variables. For instance, in the book of Martens and Næs [5], the NIR spectral data were measured at 19 fixed wavelength channels. For such data sets, when $p > n$, kernel PCA has no significant advantages compared the classic PCA algorithms. However, even in the well-known chemometric software products such as Unscrambler [6] and Matlab's Chemometrics Toolbox [30], the classic algorithms are still used to analyse spectral data sets with hundreds and thousands of variables.

2.3.1. NIPALS algorithm and its kernel version

NIPALS algorithm [9] extracts the latent variables sequentially in decreasing order of their corresponding singular values. It starts with the **X** matrix as the residue matrix and the basic iteration in this algorithm consists of the following steps:

(1) Estimate the vector **w** by 'projecting' matrix **X** onto **t**

$$\mathbf{w} = \mathbf{X}' \cdot \mathbf{t} \tag{3}$$

(2) Normalise **w** to unit sum of squares to prevent values from becoming too small or too large for the purpose of numerical computation

$$\mathbf{w} = \mathbf{w}/(\mathbf{w}' \cdot \mathbf{w})^{1/2} \tag{4}$$

(3) Update the vector **t** by projecting the **X** matrix on **w**

$$\mathbf{t} = \mathbf{X} \cdot \mathbf{w} \tag{5}$$

These steps constitute a cycle [7,16], in which there are more operations on the vector **w** ($p \times 1$) than on the vector **t** ($n \times 1$). This is efficient for $p < n$, but when $p \gg n$, it is not because **w** is much longer than **t**. In the modified algorithm, the short vector **t** ($n \times 1$) is normalised instead of the long vector **w** ($p \times 1$).

2.3.2. Power method and its kernel version

The POWER method [7,16,31,32] also extracts factors one after the other, in decreasing order of their corresponding eigenvalues. It starts with the $\mathbf{X}' \cdot \mathbf{X}$ matrix as the residue matrix ($\mathbf{C}_p$) and the algorithm includes the following basic iteration steps:

(1) Update the vector **w** by projecting the matrix $\mathbf{C}_p$ onto **w**

$$\mathbf{w} = \mathbf{C}_p \cdot \mathbf{w} \tag{6}$$

(2) Normalise the vector **w**

$$\mathbf{w} = \mathbf{w}/(\mathbf{w}' \cdot \mathbf{w})^{1/2} \tag{7}$$

In this cycle [16], the size of $\mathbf{C}_p$ ($p \times p$) and the length of the vector **w** depends on the number of variables (p). When p is small, it is quite efficient. However, when p is very large, the classic algorithm is very slow. In the kernel algorithm, the cycle is modified as follows. The matrix $\mathbf{C}_n$ ($n \times n$) is used instead of the matrix $\mathbf{C}_p$ ($p \times p$) as the residue matrix, and the vector **t** ($n \times 1$) instead of **w** ($p \times 1$). The kernel algorithm of POWER method starts with the $\mathbf{X} \cdot \mathbf{X}'$ matrix as the residue matrix ($\mathbf{C}_n$).

2.3.3. SVD algorithm and its kernel version

In SVD, all factors are obtained at the same time. The Golub–Reinsch algorithm [7,31] includes two steps. In the first step, the matrix **X** is transformed by a so-called Householder transformation to produce a bi-diagonal matrix, i.e. a matrix in which only the principal diagonal and the diagonal immediately below have non-zero elements. In the second step, the bi-diagonal matrix is decomposed into a diagonal matrix $\mathbf{\Lambda}$ by the so-called QR transformation. At the same time, the matrices **U** and **V** are obtained. In this SVD algorithm, one needs three matrices **U** ($n \times p$), $\mathbf{\Lambda}$ ($p \times p$) and **V** ($p \times p$). When $p \gg n$, the number of non-zero singular values (r) is much lower than p. In this case, only the first r columns of the matrices $\mathbf{\Lambda}$ and **V** have meaning, and all the other columns are useless. This algorithm therefore is wasteful in computer memory space and computer time. In the kernel algorithm, the matrices **U**, $\mathbf{\Lambda}$ and **V** are obtained by analysing $\mathbf{X}'$ instead of **X**,

$$[\mathbf{V}, \mathbf{\Lambda}, \mathbf{U}] = \mathtt{svd}(\mathbf{X}') \tag{8}$$

All the other steps are exactly the same as the classic algorithm. After modification, the matrix sizes are greatly reduced from ($n \times p$), ($p \times p$) and ($p \times p$) to ($p \times n$), ($n \times n$) and ($n \times n$), respectively.

2.3.4. EVD algorithm and its kernel version

EVD is also non-sequential. Similarly to SVD, it also includes the Householder transformation and the QR transformation steps [7,31]. In the Householder transformation step, the $\mathbf{C}_p$ matrix is transformed to produce a tri-diagonal matrix i.e. a matrix in which all elements are zero except the principal diagonal and the diagonals above and below. Then in the QR transformation step, the tri-diagonal matrix is decomposed into a diagonal matrix $\mathbf{\Gamma}$ and the matrix **V** is obtained.

In Matlab, EVD is also easy to program by using the built-in function `eig` [25]

$$[\mathbf{V}, \mathbf{\Gamma}] = \mathtt{eig}(\mathbf{X}' * \mathbf{X}) \tag{9}$$

However, the eigenvalues obtained in $\mathbf{\Gamma}$ and the corresponding column eigenvectors in **V** are not ranked according to the magnitude of the eigenvalues.

Therefore, one first re-arranges the columns of **V** and **Γ** in decreasing order of their eigenvalues. After ranking, the loading matrix **L**, the score matrix **S** and the predicted score matrix $\mathbf{S}_{new}$ are obtained from **V** by Eqs. (1) and (2), and the percentage of explained variance of each factor is calculated from **Γ**

$$\mathbf{P}_{ercent} = \mathrm{diag}(\mathbf{\Gamma})/\mathrm{trace}(\mathbf{\Gamma}) \cdot 100\% \quad (10)$$

The same remarks apply as for the SVD in EVD algorithm. Three $(p \times p)$ matrices are needed for $\mathbf{C}_p$, **V** and **Γ**. When $p \gg n$, then $r \ll p$. In this case, only the first r columns of the matrices **Γ** and **V** have meaning, and all the other columns are without use. In the kernel algorithm, the matrices **U** and **Γ** are obtained by analysing $\mathbf{X} \cdot \mathbf{X}'$ instead of $\mathbf{X}' \cdot \mathbf{X}$,

$$[\mathbf{U}, \mathbf{\Gamma}] = \mathtt{eig}(\mathbf{X} * \mathbf{X}') \quad (11)$$

Then the columns of **U** and **Γ** are ranked in the decreasing order of their eigenvalues. After ranking, the loading matrix **L** is calculated by $(\mathbf{X}' \cdot \mathbf{U}) \cdot \mathbf{\Gamma}^{-1/2}$, and the score matrix **S** and the predicted score matrix $\mathbf{S}_{new}$ are obtained using Eqs. (1) and (2).

2.4. Efficiency of algorithms

To compare algorithms, one normally measures the elapsed time or CPU time of the algorithms. However, the elapsed time depends on the hardware and the programming language. Another way to measure the efficiency of algorithm is to count the number of floating point operations (flop) needed in the operation of the algorithm. In the following studies, the number of flops is used as the measurement to compare different algorithms, because it is independent of the speed of computer and the programming language, and it is proportional to the elapsed time when a given hardware and software are used.

3. Experimental

3.1. Data sets

In this paper, all data sets are simulated by the uniformly distributed random number generator in Matlab. To systematically study the effect of dimension, 8 data sets with 30 objects and different number of variables i.e. 5, 10, 50, 100, 200, 300, 400 and 500 are created. The data set with 30 objects and 500 variables is also used as an example of a wide data set with many more variables than objects.

4. Results and discussion

4.1. Comparison of the classic algorithms and their kernel algorithms

The four classic algorithms and their modified kernel algorithms are first compared when all factors are calculated.

Table 1 displays the results of the comparison for the four paired methods i.e. NIPALS, POWER, SVD and EVD respectively. All results show that the kernel algorithm is faster than the corresponding classic algorithm. As the dimension increases, the improvement of the efficiency increases.

Table 1
Comparison of the classic and kernel algorithms by the number of flops (Mflops) when all PCs are required

No. of variables	Classic NIPALS	Kernel–NIPALS	Classic POWER	Kernel–POWER	Classic SVD	Kernel–SVD	Classic EVD	Kernel–EVD
1	0	0	0	1	0	0	0	0
5	2	2	1	5	0	0	0	0
10	27	24	39	15	1	1	1	1
100	78	71	220	23	3	2	8	2
200	279	221	1403	36	8	4	67	3
300	608	471	2273	55	15	6	228	4
400	873	574	3099	50	25	8	550	5
500	1435	879	5389	69	37	10	1006	7

Table 2
Comparison of the classic and kernel algorithms by the number of flops (Mflops) when only the first 2 PCs are estimated

No. of variables	Classic NIPALS	Kernel–NIPALS	Classic POWER	Kernel–POWER
1	0	0	0	1
5	0	0	0	1
10	2	2	4	1
100	7	8	23	2
200	16	8	46	2
300	31	18	146	2
400	54	27	317	3
500	254	137	3252	14

NIPALS and POWER are normally used to obtain only the first few factors. For these two methods, the time needed to obtain the first 2 factors is estimated. The results are displayed in Table 2. The same conclusion is obtained, namely that the kernel algorithms are faster than the corresponding classic algorithms, especially for the wide data sets.

4.2. Comparison of the algorithms

The literature [3,5,7,9] states that NIPALS is faster than SVD when only the first few factors are required, and SVD is more efficient when all PCs are desired. In order to check if this conclusion is still true for wide data sets, the simulated data sets with 30 objects and 500 variables are used to compare the four classic algorithms when the number of PCs varies from 1 to 10. The four kernel algorithms are also compared using this wide data set.

Fig. 1a–b display the results obtained with the four methods when different number of PCs are required. The number of flops is independent of the number of PCs for SVD and EVD because they are non-sequential. It increases with the number of PCs for NIPALS and POWER method. For the four classic methods (Fig. 1a), in contrast with what is stated in the literature, SVD is the most efficient not only when more PCs but also only a few PCs are calculated. POWER is only faster than EVD when less than 8 PCs are desired. For the four kernel algorithms (Fig. 1b), EVD is the best, next SVD, next POWER and the last NIPALS when more than 2 PCs are required. POWER is the best only when 1 PC is calculated. When the number of variables is much higher than the number of objects, kernel EVD is the most efficient algorithm. This conclusion can be explained, because the kernel EVD needs the smallest space in this case. The reason why both NIPALS and POWER do not perform efficiently when only few PCs are required is that all programs are made in Matlab. Matlab is a matrix oriented language, and matrix operations are quicker than iteration operations [25].

The four algorithms are further compared when the number of variables changes. Fig. 2a–b display the

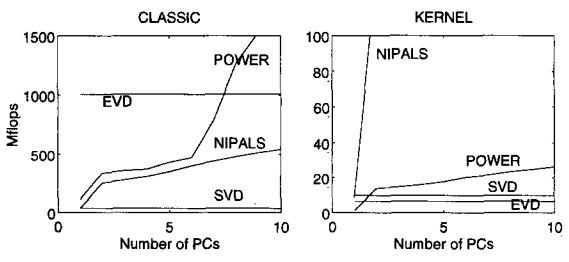

Fig. 1. The number of flops needed with the four algorithms of NIPALS, POWER, SVD and EVD versus the number of PCs, (a) when the classic algorithms are used, and (b) when the kernel algorithms are applied.

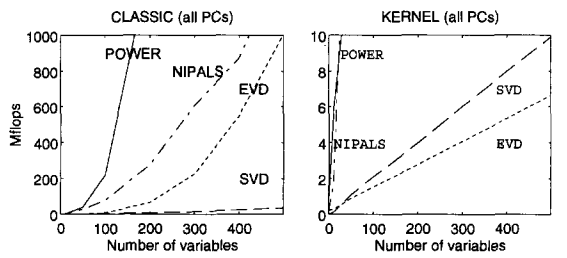

Fig. 2. The number of flops needed with the four algorithms of NIPALS, POWER, SVD and EVD versus the number of variables when all PCs are required, (a) with the classic algorithms, and (b) with the kernel algorithms. Notice the 100-fold difference in vertical scale between (a) and (b).

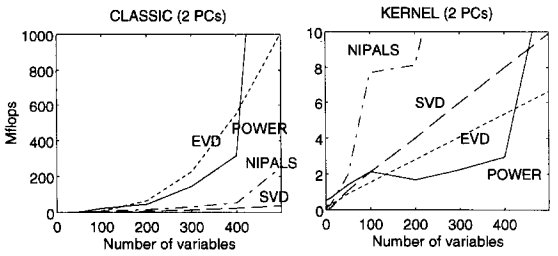

Fig. 3. The number of flops needed with the four algorithms of NIPALS, POWER, SVD and EVD versus the number of variables when only the first 2 PCs are required, (a) with the classic algorithms, and (b) with the kernel algorithms. Notice the 100-fold difference in vertical scale between (a) and (b).

results from the 8 simulated data sets with different dimensions when all PCs are estimated. The figures show that the larger the number of columns, the higher the differences between the four algorithms. When all PCs are required, non-sequential methods are better than sequential methods for both the classic algorithms and the kernel algorithms (Fig. 2). When the classic methods are used SVD is the best followed by EVD, NIPALS and POWER (Fig. 2a). When the kernel algorithms are applied, the order is EVD, SVD, POWER and NIPALS (Fig. 2b). When the number of variables is lower than the number of objects, SVD is slightly better than EVD. This can also be understood from theory.

Fig. 3a–b demonstrate the comparison when only 2 PCs are used. These figures also show that the differences between the four algorithms are relatively small when there are few variables. The differences become large when very wide data sets are used. Of the four classic algorithms, SVD is still the best (Fig. 3a). Of the four kernel algorithms, NIPALS is slowest and the differences between the other three algorithms are very small.

5. Conclusion

With the development of analytical instruments, wide data sets with many variables and relatively few objects occur more and more often. To analyse such data sets, the classic PCA algorithms such as NIPALS, POWER, SVD and EVD become very inefficient, because the size of the associated matrix $\mathbf{X}' \cdot \mathbf{X}$ $(p \times p)$ is very large. The kernel algorithms are based on the matrix $\mathbf{X} \cdot \mathbf{X}'$ $(n \times n)$, and give exactly the same scores and loadings. When the number of variables exceeds the number of objects, the kernel algorithms are therefore faster than the corresponding classic algorithms. The simulation results confirm this. They also show that the kernel–EVD is the most efficient algorithm for wide data sets, and that the difference between the kernel–SVD and kernel–EVD is not very large. SVD is the fastest of the four classic algorithms. Therefore, it is recommended to use the kernel–EVD for wide data sets both when all PCs or just the first few PCs are required. As stated in Section 1 the methods proposed are not entirely novel, but their application has remained limited. We hope that the present comparative study will encourage scientists and developers of chemometric software products to implement more efficient algorithms.

An efficient Matlab PCA program is proposed. In this program, when the number of variables is lower than the number of objects, the classic EVD algorithm is used. When the number of variables is higher than the number of objects, the kernel EVD algorithm is applied.

Acknowledgements

The first author thanks Dr. P. J. Lewi for his authorisation of citing his chapter on PCA in a forthcoming textbook. Dr. S. Rännar is also acknowledged for kindly sending the copy of his Ph.D. thesis. D.L.M. thanks the Nationaal Fonds van Wetenschappelijk Onderzoek, the DWTC and the Standards, Measurement and Testing program of the EU for financial help.

References

[1] D.L Massart, B.G.M. Vandeginste, S.N. Deming, Y. Michotte and L. Kaufman, Chemometrics: a textbook (Elsevier Science Publishers B.V., Amsterdam 1988).
[2] D.L. Massart and L. Kaufman, The interpretation of analytical chemical data by the use of cluster analysis (John Wiley and Sons, New York, 1983).
[3] B.R. Kowalski, Chemometrics: mathematics and statistics in chemistry (Reidel Publishing Company, Dordrecht, 1984).
[4] E.R. Malinowski, Factor analysis in chemistry, 2nd Ed. (John Wiley and Sons, New York, 1991).

[5] H. Martens and T. Næs, Multivariate calibration (John Wiley and Sons, Chichester, 1989).
[6] K. Esbensen, S. Schönkopf and T. Midtgarrd, Multivariate analysis in practice (Camo AS, Trondheim, 1994).
[7] P.J. Lewi, Analysis of measurement tables, in: D.L. Massart et al. (Eds.), Chemometrics and Qualimetrics, Vol. II (Elsevier, Amsterdam), in preparation.
[8] J.E. Jackson, A user's guide to principal components (John Wiley and Sons, New York, 1991).
[9] S. Wold, K. Esbensen and P. Geladi, Chemom. Intell. Lab. Syst. 2 (1987) 37–52.
[10] F.C. Sánchez, P.J. Lewi and D.L. Massart, Chemom. Intell. Lab. Syst. 25 (1994) 157–177.
[11] F.C. Sánchez, Mixture analysis by multivariate techniques: peak purity assessment, Ph.D. thesis, Pharmaceutical Institute, Free University, Brussels (1996).
[12] P.J. Lewi, Chemom. Intell. Lab. Syst. 5 (1989) 105–116.
[13] S. Wold, Technometrics 20 (1978) 397–405.
[14] P. Geladi and B.R. Kowalski, Anal. Chim. Acta 185 (1986) 1–17.
[15] H.T. Eastment and W.J. Krzanowski, Technometrics 24 (1982) 73–77.
[16] P.J. Lewi, Chemom. Intell. Lab. Syst. 28 (1995) 23–33.
[17] D.G. Evans, A. Legrand, K. Jewell and C.N.G. Scotter, J. Near Infrared Spectrosc. 1 (1993) 209–219.
[18] J.M. Sutter, J.H. Kalivas and P.M. Lang, J. Chemom. 6 (1992) 217–225.
[19] J. Sun, J. Chemom. 9 (1995) 21–29.
[20] W.J. Krzanowski, J. Chemom. 6 (1992) 97–102.
[21] W. Wu, B. Walczak, W. Penninckx and D.L. Massart, Anal. Chim. Acta, in press.
[22] S. Rännar, Many variables in multivariate projection methods, Ph.D. thesis, Department of Organic Chemistry, Umeå University, Sweden (1996).
[23] B.M. Wise, PLS_toolbox for use with MATLAB (version 1.4) (Eigenvector Technologies, West Richland, 1994).
[24] E.J. Karjalainen and U.P. Karjalainen, Data analysis for hyphenated techniques (Elsevier Science Publishers B.V., Amsterdam, 1996).
[25] MATLAB reference guide (The MathWorks, Inc., Natick, 1992).
[26] S. Rännar, F. Lindgren, P. Geladi and S. Wold, J. Chemom. 8 (1994) 111–125.
[27] S. Rännar, P. Geladi, F. Lindgren and S. Wold, J. Chemom. 9 (1995) 459–470.
[28] F. Lindgren, P. Geladi and S. Wold, J. Chemom. 7 (1993) 45–59.
[29] F. Lindgren, P. Geladi and S. Wold, J. Chemom. 8 (1994) 377–389.
[30] R. Kramer, Chemometrics toolbox for use with MATLAB user's guide (The MathWorks, Inc., Natick, 1989).
[31] G.H. Golub and C.F. Van Loan, Matrix computions (North Oxford Academic Publishing Co. Ltd., Oxford, 1983).
[32] R. Manne, Chemom. Intell. Lab. Syst. 2 (1987) 187–197.