Stat Papers (2009) 50:711–720
DOI 10.1007/s00362-009-0251-7

Comparison of PLS algorithms when number of objects is much larger than number of variables

Aylin Alin

Received: 12 September 2008 / Accepted: 18 May 2009 / Published online: 5 August 2009

Abstract NIPALS and SIMPLS algorithms are the most commonly used algorithms for partial least squares analysis. When the number of objects, N, is much larger than the number of explanatory, K, and/or response variables, M, the NIPALS algorithm can be time consuming. Even though the SIMPLS is not as time consuming as the NIPALS and can be preferred over the NIPALS, there are kernel algorithms developed especially for the cases where N is much larger than number of variables. In this study, the NIPALS, SIMPLS and some kernel algorithms have been used to built partial least squares regression model. Their performances have been compared in terms of the total CPU time spent for the calculations of latent variables, leave-one-out cross validation and bootstrap methods. According to the numerical results, one of the kernel algorithms suggested by Dayal and MacGregor (J Chemom 11:73–85, 1997) is the fastest algorithm.

Keywords High-dimensional data · Kernel matrix · Multicollinearity · Multiple linear regression · NIPALS · Partial least squares · SIMPLS

List of symbols

$\mathbf{X}$	$N \times K$ matrix of explanatory variables
$\mathbf{Y}$	$N \times M$ matrix of response variables
$\mathbf{F}$	$N \times M$ matrix of residuals
$\mathbf{B}_{\mathrm{PLS}}$	$K \times M$ matrix of PLS regression coefficients
$\mathbf{T}$	$N \times A$ matrix of PLS latent variables for $\mathbf{X}$
$\mathbf{U}$	$N \times A$ matrix of PLS latent variables for $\mathbf{Y}$

A. Alin (✉)
Fen Edebiyat Fakultesi Istatistik Bolumu, Dokuz Eylul Universitesi, Izmir, Turkey
e-mail: aylin.alin@deu.edu.tr

$\mathbf{W}$	$K \times A$ matrix of weights of deflated $\mathbf{X}$ matrix on latent variables $\mathbf{T}$
$\mathbf{R}$	$K \times A$ matrix of weights of original $\mathbf{X}$ matrix on latent variables $\mathbf{T}$
$\mathbf{C}$	$M \times A$ matrix of weights of $\mathbf{Y}$ on latent variables $\mathbf{U}$
$\mathbf{P}$	$K \times A$ matrix of loadings for $\mathbf{X}$
$\mathbf{t}_{\mathrm{a}}$	A column vector of $\mathbf{T}$
$\mathbf{u}_{\mathrm{a}}$	A column vector of $\mathbf{U}$
$\mathbf{w}_{\mathrm{a}}$	A column vector of $\mathbf{W}$
$\mathbf{r}_{\mathrm{a}}$	A column vector of $\mathbf{R}$
$\mathbf{c}_{\mathrm{a}}$	A column vector of $\mathbf{C}$
$\mathbf{p}_{\mathrm{a}}$	A column vector of $\mathbf{P}$

Uppercase bold variables will represent matrices and lower case bold variables will represent column vectors in the paper. The transpose of a matrix will be given with "$'$". N, K, M and A are the number of objects, the number of explanatory variables, the number of response variables and the number of latent variables, respectively. The notations used in the paper are given above. It is assumed that the columns of $\mathbf{X}$ and $\mathbf{Y}$ are mean-centered and scaled prior to PLS model estimation to have mean zero and standard deviation one.

1 Introduction

Partial least squares regression (PLSR), which is a combination of two methods: partial least squares (PLS) analysis and multiple linear regression (MLR), is a latent variable-based multivariate technique which allows modeling and predicting multiple response variables $\mathbf{Y}$ from highly correlated or collinear multiple explanatory variables $\mathbf{X}$. With the combination of PLS and MLR, we form latent variables that capture most of the information in $\mathbf{X}$ for predicting $\mathbf{Y}$ while reducing the dimensions of $\mathbf{X}$ using fewer latent variables than the number of explanatory variables.

Partial least squares creates latent variables for both $\mathbf{X}$ and $\mathbf{Y}$ using different algorithms among which the nonlinear iterative partial least squares (NIPALS) and the straightforward implementation of a statistically inspired modification of the PLS method (SIMPLS) algorithms are the two most commonly used algorithms. For other algorithms see Lindgren and Rännar (1998). When N is much larger than ($\gg$) K and/or M, such as; multivariate image analysis, process control, environmental monitoring studies, etc., using the NIPALS can be time consuming. Even though the SIMPLS is faster than the NIPALS, there are kernel algorithms developed especially for the cases $N \gg K$ and/or M, such as; the kernel algorithms developed by Lindgren et al. (1993), De Jong and Ter Braak (1994) and Dayal and MacGregor (1997) to mention a few. De Jong and Ter Braak (1994) showed that the algorithm they proposed is more economical than the one developed by Lindgren et al. (1993) in terms of the total floating point operations (flops) used during the calculation of latent variables. Later, Dayal and MacGregor (1997) showed that two modified algorithms they developed are faster than the kernel algorithm developed by De Jong and Ter Braak (1994) based on the total number of flops used. However, in both of these studies, they did not include the numerical results for the cross validation (CV) or some other validation procedures which are strongly recommended in PLSR. They only mention that their algorithms may be more efficient than the ones they compared in the case of CV.

In this study, the aim is to compare these kernel algorithms and the mostly used algorithms NIPALS and SIMPLS in terms of the total CPU time used not only for the calculation of the latent variables, but also for the leave-one-out CV and bootstrap methods. It should be noted that detailed explanation of the algorithms and the meaning of the vectors used in these algorithms for PLSR will not be given because of space considerations. For detailed information on the interpretation of PLSR model and its components see Wold et al. (2001). The organization of the paper is as follows. After giving a brief description of the PLSR model in the following section, detailed explanation of the comparison results will be presented in Sect. 4.

2 PLSR model

The objective of all PLSR algorithms is to build the following PLSR model

$$\mathbf{Y} = \mathbf{X}\mathbf{B}_{\mathrm{PLS}} + \mathbf{F}. \tag{1}$$

Even though this model is similar to the MLR model, the calculation of the regression coefficients is different. To estimate $\mathbf{B}_{\mathrm{PLS}}$ as given in Eq. (2), we need to obtain the matrices **W, C** and **P**.

$$\mathbf{B}_{\mathrm{PLS}} = \mathbf{R}\mathbf{C}' \quad \text{where } \mathbf{R} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}. \tag{2}$$

PLS matrices **T** and **U** contain latent variables which are calculated as the linear combination of **X** and **Y**.

$$\mathbf{t}_{\mathrm{a}} = \mathbf{X}\mathbf{r}_{\mathrm{a}}, \quad \mathbf{u}_{\mathrm{a}} = \mathbf{Y}\mathbf{c}_{\mathrm{a}} \quad \text{for } a = 1, 2, \ldots, \mathrm{A} \tag{3}$$

$\mathbf{t}_{\mathrm{a}}$s are orthogonal to each other and also each $\mathbf{t}_{\mathrm{a}}$ is orthogonal to the subsequent $\mathbf{u}_{\mathrm{a}}$, i.e., $\mathbf{t}_i \perp \mathbf{t}_j$ for $i \neq j$ and $\mathbf{u}_j \perp \mathbf{t}_i$ for $j > i$. The column vectors of $\mathbf{w}_{\mathrm{a}}, \mathbf{c}_{\mathrm{a}}, \mathbf{p}_{\mathrm{a}}, \mathbf{t}_{\mathrm{a}}$ and $\mathbf{u}_{\mathrm{a}}$, for $a = 1, 2, \ldots, A$, are calculated sequentially in all algorithms considered in this study. These vectors are obtained by deflating **X** and **Y** matrices in the NIPALS algorithm while deflation is performed directly on covariance matrix $\mathbf{X}'\mathbf{Y}$ in the SIMPLS without multiplying $\mathbf{X}'$ and **Y** at each iteration. The kernel matrix approach used by Lindgren et al. (1993), De Jong and Ter Braak (1994) and Dayal and MacGregor (1997) is based on obtaining $\mathbf{w}_{\mathrm{a}}$ as the eigenvector corresponding to the ath largest eigenvalue of the kernel matrix $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$ when $M > K$ or $\mathbf{c}_{\mathrm{a}}$ as the eigenvector corresponding to the ath largest eigenvalue of the kernel matrix $\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}$ when $M < K$. Then, $\mathbf{p}_{\mathrm{a}}$ and $\mathbf{c}_{\mathrm{a}}$ or $\mathbf{w}_{\mathrm{a}}$ (if $M < K$, $\mathbf{c}_{\mathrm{a}}$ is calculated first) can easily be calculated using small covariance matrices $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{Y}$ having dimensions $K \times K$ and $K \times M$, respectively. Lindgren et al. (1993), De Jong and Ter Braak (1994) both preferred deflating only **X** in $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{Y}$ without multiplying matrices. Difference between their studies is at their deflation process. Dayal and MacGregor (1997) proposed two modified kernel algorithms where **Y** is deflated instead of **X**. Hence, deflation of $\mathbf{X}'\mathbf{X}$ has been removed and only $\mathbf{X}'\mathbf{Y}$ has been deflated. In their first modified algorithm, $\mathbf{X}$ is directly used for the calculation of weights and loadings whereas in the second one, $\mathbf{X}'\mathbf{X}$ is calculated once at the beginning of iterations for weights and loadings and it is directly used for the calculations. All of $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$, $\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}$, $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{Y}$ have dimensions which are independent of the number of objects. Therefore, it does not matter how large N is. For more information on the algorithms see Lindgren et al. (1993), De Jong and Ter Braak (1994) and Dayal and MacGregor (1997).

3 Comparison of algorithms

The NIPALS, SIMPLS, kernel algorithms developed by Lindgren et al. (1993), De Jong and Ter Braak (1994) and Dayal and MacGregor (1997) have been compared. The modified version of NIPALS given by Dayal and MacGregor (1997) is also included in the comparisons to show that even discarding the deflation of $\mathbf{X}$ and deflating only $\mathbf{Y}$ contributes to the speed of the algorithm. Totally, seven algorithms have been compared. The measurement used for the comparison is the total central processing unit (CPU) time elapsed during the calculations including the calculation for latent variables, calculations for the leave-one-out CV method and bootstrap method where 100 bootstrap samples are created. For the calculations, MATLAB 7.0.1© has been used. The code can be obtained from the author upon request. All of the comparisons have been done using the computer with configurations: CPU T5500 @ 1.66 GHz and 1.00 GB RAM. The algorithms were tested for speed via CPU time to variations in N, K, M and A. A total of 49 combinations of the levels of these variables used for the comparison of the algorithms are given in Table 1. Each of the seven algorithms was tested on these combinations. Data distributed uniformly between 0 and 1 have been used.

As mentioned by Rännar et al. (1994), the choice of the number of latent variables, A, is a matrix rank problem which causes overfitting when A has been chosen too large and underfitting when A has been chosen too small. Leave-one-out CV method can be used to determine the optimal A for the regression model with better predictive ability. In this procedure, objects are deleted one at a time and A latent variables are extracted based on the remaining $N - 1$ observations. After determining the optimal A which has the minimum prediction error sum of squares (PRESS), we determine the significance of nonzero weights r_{ka} and c_{ma} for $a = 1, 2, \ldots, A$, $k = 1, 2, \ldots, K$, $m = 1, 2, \ldots, M$. This helps to verify that the weight of $\mathbf{X}_{\mathrm{k}}$ on the corresponding $\mathbf{t}_{\mathrm{a}}$ and the importance of $\mathbf{t}_{\mathrm{a}}$ to model corresponding $\mathbf{Y}_{\mathrm{m}}$ are not only by chance. This will also help to decide the significance of the PLS regression coefficients given in Eq. (2). Bootstrap method can be used for calculating the standard errors of the PLS regression coefficients $B_{(\mathrm{PLS})\mathrm{km}}$, and weights r_{ka} and c_{ma}. Detailed information about the bootstrap and CV methods can be found in Picard and Cook (1984), Efron and Tibshirani (1986), Shao (1993) and Boos (2003). However, it should be noted that these methods are very time-intensive.

Total CPU times in seconds elapsed during the calculations are presented in Table 1 along with the corresponding designs. The algorithm with the smallest CPU time

Table 1 Designs and total CPU times (in seconds) for calculation of latent variables, leave-one-out CV and bootstrap methods

N	*K*	*M*	*A*	NIPALS	Modified NIPALS	SIMPLS	Lindgren et al. alg.	De Jong and Ten Braak alg.	Modified kernel #1	Modified kernel #2
500	10	1	3	10.625	10.344	10.297	10.406	10.359	10.500	**10.266**
500	10	1	5	25.438	24.578	24.647	24.859	24.703	24.703	**24.547**
500	10	5	3	13.188	12.859	12.750	12.891	12.781	12.688	**12.656**
500	10	5	5	31.516	30.922	30.469	30.625	30.422	30.297	**30.188**
500	10	15	3	17.516	17.125	16.703	16.750	16.688	16.625	**16.563**
500	10	15	5	41.609	40.906	39.766	39.828	39.656	39.500	**39.391**
2,500	10	1	3	57.063	51.672	51.578	52.859	52.563	50.656	**50.000**
2,500	10	1	5	136.266	124.453	123.969	124.328	124.219	121.672	**118.797**
2,500	10	5	3	73.797	68.219	65.156	65.469	65.266	62.969	**62.813**
2,500	10	5	5	177.219	164.344	155.844	153.750	152.719	150.453	**147.469**
2,500	10	15	3	106.203	96.031	86.516	86.438	86.078	84.359	**83.500**
2,500	10	15	5	255.813	231.875	205.281	201.500	200.766	198.109	**195.609**
10,000	10	1	3	411.531	265.281	270.438	283.766	283.563	252.859	**244.594**
10,000	10	1	5	954.766	609.547	622.172	604.781	603.844	570.391	**522.719**
10,000	10	5	3	571.281	428.484	358.125	356.469	355.547	322.578	**311.656**
10,000	10	5	5	1,268.500	940.250	805.203	750.531	748.672	721.266	**668.734**
10,000	10	15	3	911.469	774.547	531.203	496.656	498.734	469.719	**457.219**
10,000	10	15	5	2,114.000	1,801.200	1,170.500	1,024.800	1,014.200	991.625	**936.125**
1,500	20	10	4	120.172	106.063	101.250	105.188	104.328	99.375	**99.344**
1,500	20	10	8	410.391	384.953	363.953	367.750	364.563	359.578	**353.313**
500	30	1	3	17.000	16.094	16.281	17.719	17.188	**16.125**	16.328
500	30	1	5	40.594	38.781	38.344	41.328	40.094	**38.281**	38.359

Table 1 continued

N	*K*	*M*	*A*	NIPALS	Modified NIPALS	SIMPLS	Lindgren et al. alg.	De Jong and Ten Braak alg.	Modified kernel #1	Modified kernel #2
500	30	1	10	143.359	137.344	137.563	142.047	139.281	135.359	**134.594**
500	30	5	3	17.875	17.000	16.813	18.344	17.797	**16.672**	16.922
500	30	5	5	42.609	40.750	39.797	42.484	41.484	**39.625**	39.734
500	30	5	10	151.359	143.688	140.234	146.563	142.938	139.719	**138.828**
500	30	15	3	27.016	26.359	25.594	27.016	26.656	**25.547**	25.859
500	30	15	5	70.234	68.578	66.250	69.375	67.578	**65.844**	65.938
500	30	15	10	257.297	251.109	246.531	248.781	247.969	243.406	**241.703**
2,500	30	1	3	111.344	84.891	84.625	110.844	109.172	**83.422**	89.016
2,500	30	1	5	262.938	201.719	200.875	241.313	237.172	197.484	**197.031**
2,500	30	1	10	919.688	716.000	707.234	778.125	764.438	704.188	**672.891**
2,500	30	5	3	124.266	99.688	93.281	118.984	116.594	**90.547**	95.230
2,500	30	5	5	293.859	235.531	213.578	253.500	247.875	209.203	**208.156**
2,500	30	5	10	1,033.500	831.469	748.906	812.125	792.109	732.797	**701.688**
2,500	30	15	3	185.219	158.766	140.125	167.219	162.906	**136.844**	143.922
2,500	30	15	5	479.516	419.531	359.297	394.438	391.063	348.766	**346.875**
2,500	30	15	10	1,746.300	1,546.200	1,306.900	1,360.400	1,326.100	1,267.300	**1,244.300**
10,000	30	1	3	1,117.300	595.141	567.391	904.625	895.313	**550.688**	577.688
10,000	30	1	5	2,577.200	1,309.000	1,232.300	1,681.400	1,668.100	1,382.100	**1,030.000**
10,000	30	1	10	9,120.500	4,524.200	4,118.900	4,482.000	4,397.100	3,998.100	**2,973.900**
10,000	30	5	3	1,265.000	742.344	654.484	980.813	972.250	**620.375**	646.984
10,000	30	5	5	2,971.800	1,757.700	1,411.100	1,803.600	1,796.800	1,323.800	**1,151.300**
10,000	30	5	10	10,334.000	6,050.400	4,590.600	4,693.400	4,623.000	4,140.300	**3,194.000**
10,000	30	15	3	1,862.100	1,328.300	959.984	1,265.800	1,261.400	**916.500**	982.313

Table 1 continued

N	*K*	*M*	*A*	NIPALS	Modified NIPALS	SIMPLS	Lindgren et al. alg.	De Jong and Ten Braak alg.	Modified kernel #1	Modified kernel #2
10,000	30	15	5	4,446.800	3,161.200	2,127.800	2,490.400	2,470.100	1,984.400	**1,834.500**
10,000	30	15	10	15,631.000	11,241.000	7,351.000	8,601.700	6,957.700	6,549.900	**5,541.200**
100,000	30	1	3	88,126.000	29,529.000	28,166.000	62,473.000	61,913.000	**26,565.000**	30,373.000
100,000	30	1	5	184,910.000	63,797.000	54,050.000	99,954.000	99,638.000	48,775.000	**36,302.000**

is given in bold for each design. Without giving any numerical results, Dayal and MacGregor (1997) recommended using the modified kernel #2 algorithm for the leave-one-out CV. They thought that one time construction of $\mathbf{X}'\mathbf{X}$ at the beginning and using it directly for the calculations of weights and loadings for each object would be faster than using $\mathbf{X}$ with bigger dimensions. Our results support their suggestions for the designs with $K = 10$ and $K = 20$ regardless of the levels of N, M and A. However, for the largest K considered, the modified kernel #1 algorithm is better for the combinations with $N = 500$, $M = 1, 5, 15$ and $A = 3, 5$. When K gets larger, using $\mathbf{X}'\mathbf{X}$ for the calculations of weights and loadings by calculating it at the beginning of iterations for each object needs more effort than using bigger matrix $\mathbf{X}$. On the other hand, this effort is compensated if the number of latent variables extracted is increased to 10 where the modified kernel #2 algorithm gets better. Moreover, as we increase N the modified kernel #2 algorithm is faster not only for $A = 10$ but also for $A = 5$. The larger the sample size, the better the modified kernel #2 algorithm gets compared to the modified kernel #1 algorithm for $A = 5, 10$.

Table 2 includes the pair-wise comparisons of the algorithms in terms of average decrease in the total CPU time when all of the 49 designs are considered. The simulation results reveal that even omitting to deflate $\mathbf{X}$ in the NIPALS speeds up the algorithm up to 67% for the combinations with $N = 100{,}000$ and $K = 30$. This is the case where the modified NIPALS algorithm compared to the classical NIPALS in which both $\mathbf{X}$ and $\mathbf{Y}$ matrices are deflated. Including all designs, average decrease in the total CPU time for the modified NIPALS algorithm is 19% compared to the classical NIPALS algorithm. De Jong (1993) developed the SIMPLS algorithm over the NIPALS to get a faster algorithm. According to the results, the SIMPLS is almost 25% faster than the NIPALS on average when all designs are considered. Compared to the NIPALS, average decrease in the total CPU time for the modified kernel #1 and #2 algorithms are 27 and 29%, respectively. The CPU time advantages of the SIMPLS and the modified kernel #1 and #2 algorithms over the NIPALS are more significant than other results. It should be noted that effectiveness of these algorithms over the NIPALS algorithm is much more apparent for the largest sample size considered such that for $N = 100{,}000$, $K = 30$, $M = 1$, $A = 5$, efficiency of the SIMPLS and the modified kernel #1 and #2 algorithms over the NIPALS in terms of speed increase to 71, 74 and 80%, respectively as illustrated in Fig. 1. The SIMPLS algorithm is 13 and 11% faster than the algorithms developed by Lindgren et al. (1993) and De Jong and Ter Braak (1994), respectively even though these algorithms have been especially developed for the designs with $N \gg K$ and/or M. It seems that directly deflating covariance matrix $\mathbf{X}'\mathbf{Y}$ at each iteration needs less effort than deflating $\mathbf{X}$ in $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{Y}$. According to the results, the speed improvements for the modified kernel #1 and #2 algorithms are 11 and 13% compared to the De Jong and Ter Braak algorithms. The results also reveal that the modified kernel #1 and #2 algorithms are 3 and 6% faster than the SIMPLS. The advantage of the modified kernel #2 over the modified kernel #1 is 3%. Performance of the modified kernel #2 algorithm over #1 gets much better for the combinations with large sample size, large number of explanatory variables, and moderate and large numbers of latent variables. Even though the modified kernel #1 algorithm is better than #2 for $A = 3$, its superiority is 14% for the design where $N = 100{,}000$.

Table 2 Pairwise comparisons in terms of average decrease (as %) in CPU times

Algorithms	Average decrease (as %)
The NIPALS versus The modified NIPALS	19
The NIPALS versus The SIMPLS	25
Lindgren et al. algorithm versus De Jong and Ter Braak algorithm	2
The SIMPLS versus Lindgren et al. algorithm	−13
The SIMPLS versus De Jong and Ter Braak algorithm	−11
The NIPALS versus The modified kernel #1 algorithm	27
The NIPALS versus The modified kernel #2 algorithm	29
De Jong and Ter Braak algorithm versus The modified kernel #1 algorithm	11
De Jong and Ter Braak algorithm versus The modified kernel #2 algorithm	13
The SIMPLS versus The modified kernel #1 algorithm	3
The SIMPLS versus The modified kernel #2 algorithm	6
The modified kernel #1 algorithm versus The modified kernel #2 algorithm	3

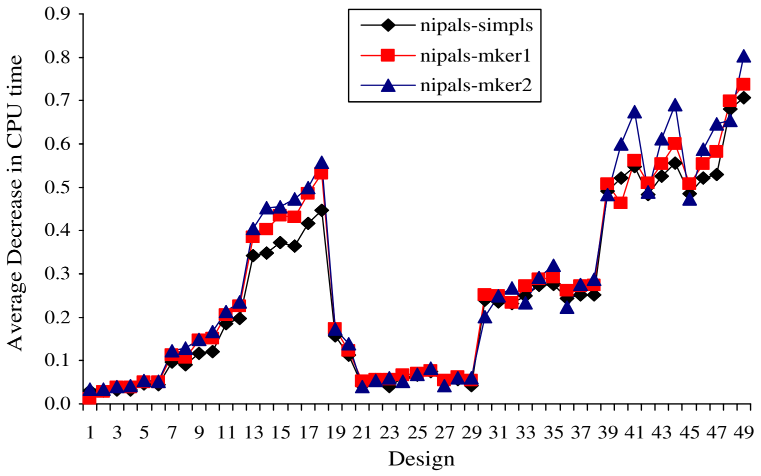

Fig. 1 Average decrease (as %) in total CPU time versus design

However, the superiority of the modified kernel #2 over #1 is 25% when we combine $N = 100{,}000$ with $A = 5$. Figure 2 illustrates performances of these two algorithms for the designs where $K = 30$ have been matched with sample sizes 10,000 and 100,000.

4 Conclusion

Considering the popularity of the SIMPLS algorithm and its competitiveness to the other algorithms especially developed for the cases where sample size is much larger than the number of variables, some researchers would prefer to use the SIMPLS.

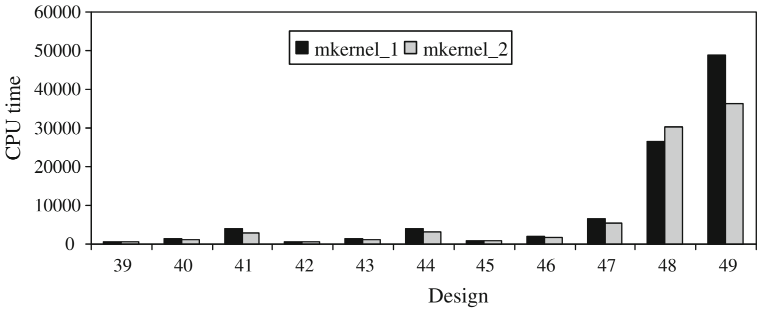

Fig. 2 CPU times for the modified kernel #1 and #2 algorithms for the last 11 designs

However, the results of this study reveal that the modified kernel #2 algorithm is generally faster than the other algorithms including the SIMPLS in terms of the total CPU time elapsed during PLSR model building including the leave-one-out-CV and bootstrap methods. With the rapid improvement in the science and technology to gather the data, it gets normal to study with the large number of samples such as 10,000, 100,000 or even more. Considering the performance of the modified kernel #2 algorithm with large sample sizes as well as its overall performance for all designs, it would be wise decision to use it to built PLSR model.

Acknowledgments The author would like to thank Prof. Hamparsum Bozdogan for his comments and corrections for the paper. The author also thanks two anonymous reviewers for their constructive comments which improved the paper a lot.

References

Boos DD (2003) Introduction to the bootstrap world. Stat Sci 18:168–174

Dayal BS, MacGregor JF (1997) Improved PLS algorithms. J Chemom 11:73–85

De Jong S (1993) SIMPLS: an alternative approach to partial least squares regression. Chemom Intell Lab Syst 18:251–263

De Jong S, Ter Braak CJF (1994) Short communication: comments on the PLS kernel algorithm. J Chemom 8:169–174

Efron B, Tibshirani R (1986) Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Stat Sci 1:54–77

Lindgren F, Rännar S (1998) Alternative partial least squares (PLS) algorithms. Perspectives Drug Discov Des 12/13/14:105–113

Lindgren F, Geladi P, Wold S (1993) The kernel algorithm for PLS. J Chemom 7:45–59

Picard RR, Cook RD (1984) Cross-validation of regression models. J Am Stat Assoc 79:575–583

Rännar S, Lindgren F, Geladi P, Wold S (1994) A PLS kernel algorithm for data sets with many variables and fewer objects. Part 1: theory and algorithm. J Chemom 8:111–125

Shao J (1993) Linear model selection by cross-validation. J Am Stat Assoc 88:486–494

Wold S, Sjöström M, Eriksson L (2001) PLS-regression: a basic tool of chemometrics. Chemom Intell Lab Syst 58:109–130