Received: 7 August 2019 | Revised: 14 November 2019 | Accepted: 17 November 2019

DOI: 10.1002/cem.3201

RESEARCH ARTICLE

WILEY Journal of CHEMOMETRICS

Much faster cross-validation in PLSR-modelling by avoiding redundant calculations

Kristian Hovde Liland | Petter Stefansson | Ulf Geir Indahl

Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

Correspondence
Kristian Hovde Liland, Faculty of Science and Technology, Norwegian University of Life Sciences, NO-1432 Ås, Norway,
kristian.liland@nmbu.no

Abstract

A novel formulation of the wide kernel algorithm for partial least squares regression (PLSR) is proposed. We show how the elimination of redundant calculations in the traditional applications of PLSR helps in speeding up any choice of cross-validation strategy by utilizing precalculated lookup matrices.

The proposed lookup approach is combined with some additional computational shortcuts resulting in highly effective and numerically accurate cross-validation results. The computational advantages of the proposed method are demonstrated by comparisons to the classical NIPALS and the bidiag2 algorithms for calculating cross-validated PLSR models. Problems including both one and several responses, double/nested cross-validated, and one-vs-all classification are among the considered applications.

KEYWORDS

classification, kernel matrix, regression, subspace modelling, validation

1 | INTRODUCTION

The partial least squares regression (PLSR) method was introduced to the field of chemometrics in the early 1980s.[1-4] However, the computational ideas of PLSR were discovered much earlier and are in fact equivalent to a much older implementation of the conjugate gradient method of Hestenes and Stiefel[5] for numerically solving the normal equations of an ordinary least squares (OLS) problem; see Phatak and de Hoog.[6]

Later, numerous algorithms for implementing PLSR have been proposed. Underlying motivations in these developments have been interpretational aspects, computational speed, and numerical precision.[7-9] In the case of modelling with data sets containing more predictors than samples, several kernel-based algorithms have been proposed.[10-13] Recently, it has been demonstrated how the kernel approach can be utilized for efficiently exploring a large number of alternative variable subset selections.[14] The present work introduces a novel kernel-based algorithm designed for obtaining efficient PLSR model selections based on arbitrary cross-validation strategies that are much faster than the traditional cross-validation approaches to PLSR model selection.

Below, we assume that $(\mathbf{X}, \mathbf{y})$ represents the dataset of predictors $\mathbf{X}$ (a matrix of n rows and p columns) and corresponding responses $\mathbf{y}$ (a vector of n rows) subject to PLSR modelling. Our main demonstration is that the essential requirement for obtaining the cross-validation results associated with a PLSR-model is one single calculation of the $(n \times n)$ matrix product $\mathbf{XX}'$. All later calculations of the required (PLS) score vectors in the cross-validated model building are based on indexing into $\mathbf{XX}'$, $\mathbf{y}$ and two additional precalculated matrices.

The optional calculation of loading weights and PLSR regression coefficients can be obtained in a final post-processing step after the final model selection. In a comparison study, we demonstrate that the proposed algorithm retains numerical precision similar to the NIPALS algorithm for PLSR modelling while obtaining a huge advantage in computational speed, in particular for situations with very wide predictor matrices ($p \gg n$) and models including a large number of PLS components. The bidiag2 algorithm,[7,9] known as a computationally faster alternative to the NIPALS algorithm, is also included for comparison.

In Section 2, we will take the NIPALS algorithm as a basis for formulating a more efficient PLSR algorithm for situations with wide **X**-matrices. We will explain how to avoid redundant computations for both the ordinary and the double/nested cross-validation including regression problems with multiple responses. In Section 3, we report the efficiency and accuracy of the proposed method based on a simulation study focusing on single-response regression problems and a one-vs-all classification problem using real data.

2 | METHODOLOGY

2.1 | Single-response PLSR

In the following, we assume that the columns of both **X** and **y** are centred. On the basis of the NIPALS algorithm for PLSR modelling, our goal is to establish a faster alternative for situations where the predictor matrices are wide ($p \gg n$). Equation (1) shows the steps of a single pass in the NIPALS algorithm for extracting the loading weights (**w**), **X**-scores (**t**), **X**-loadings (**p**), **y**-loadings (q), **X**-deflation, and **y**-deflation, respectively. To simplify the notation and enable direct comparison with the proposed kernel algorithm, we will refer to the following version of the NIPALS PLS algorithm where $\mathbf{y}_0 = \mathbf{y}$ and the score vectors are normalized:

$$\begin{aligned}
\mathbf{w}_i &= \mathbf{X}'\mathbf{y}_{i-1}/||\mathbf{X}'\mathbf{y}_{i-1}|| && \text{(Loading weights)}\\
\mathbf{t}_i &= \mathbf{X}\mathbf{w}_i/||\mathbf{X}\mathbf{w}_i|| && \text{(Scores)}\\
\mathbf{p}_i &= \mathbf{X}'\mathbf{t}_i && (\mathbf{X}-\textit{loadings})\\
q_i &= \mathbf{t}_i'\mathbf{y}_{i-1} && (\mathbf{y}-\text{loadings})\\
\mathbf{X}_i &= \mathbf{X}_{i-1} - \mathbf{t}_i\mathbf{p}_i' && (\mathbf{X}-\text{deflation})\\
\mathbf{y}_i &= \mathbf{y}_{i-1} - \mathbf{t}_i q_i && (\mathbf{y}-\text{deflation})
\end{aligned} \tag{1}$$

The above computational steps in (1) are repeated until the appropriate number (k) of PLS-components are extracted.

From the computational details of the NIPALS algorithm, it can be shown that explicit calculations of the loading weights (**w**) and **X**-loadings (**p**) are not necessary to derive the orthogonal PLS-scores (**t**) and associated **y**-loadings (q). By deflating the (residual) response $\mathbf{y}_{i-1}$ into $\mathbf{y}_i = \mathbf{y}_{i-1} - q_i\mathbf{t}_i$ and preserving **X** undeflated, the subsequent score vector can be obtained by $\mathbf{t}_{i+1} = (\mathbf{X}\mathbf{X}')\mathbf{y}_i$ followed by orthogonalization with respect to the previously extracted scores ($\mathbf{t}_1, \ldots, \mathbf{t}_i$) and normalization. Formally, the proposed parsimonious kernel PLS (PKPLS) algorithm has the following steps:

$$\begin{aligned}
\mathbf{y}_0 &= \mathbf{y} \\
\mathbf{C} &= \mathbf{X}\mathbf{X}' \\
\mathbf{T} &= [\ ] && \text{(empty array for appending the orthogonal Scores)}\\
\mathbf{R}_\mathbf{y} &= [\ ] && \text{(empty array for appending the y-residuals)}\\
&- -\\
\mathbf{t}^* &= \mathbf{C}\mathbf{Y}_{i-1} && \text{(non-orthogonal Score)}\\
\mathbf{t}_i &= (\mathbf{t}^* - \mathbf{T}(\mathbf{T}'\mathbf{t}^*))/||\mathbf{t}^* - \mathbf{T}(\mathbf{T}'\mathbf{t}^*)|| && \text{(deflate and normalize } \mathbf{t}^*)\\
\mathbf{T} &= [\mathbf{T}\ \ \mathbf{t}_i] && \text{(appendix } \mathbf{t}_i)\\
\mathbf{R}_y &= [\mathbf{R}_y\ \ \mathbf{y}_{i-1}] && \text{(appendix } \mathbf{y}_{i-1})\\
q_i &= \mathbf{t}_i'\mathbf{y}_{i-1} && \text{(y-loadings)}\\
\mathbf{y}_i &= \mathbf{y}_{i-1} - \mathbf{t}_i q_i && \text{(y-deflation)}
\end{aligned} \tag{2}$$

The computational steps below the dashed line are repeated until the appropriate number (k) of components are extracted. The optional calculation of associated **X**-loadings $\mathbf{P} = [\mathbf{p}_1 \ldots \mathbf{p}_k]$ and loading weights $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w}_k]$ after completing the extraction of k orthonormal scores $\mathbf{T} = [\mathbf{t}_1 \ldots \mathbf{t}_k]$ are given by $\mathbf{P} = \mathbf{X}'\mathbf{T}$ and a column-wise normalization of

$\mathbf{W} = \mathbf{X}'\mathbf{R}_\mathbf{y}$. The matrix $\mathbf{R}_\mathbf{y} = [\mathbf{y}_0\, \mathbf{y}_1\, \mathbf{y}_2 \ldots \mathbf{y}_{k-1}]$ has the original responses $\mathbf{y}_0 = \mathbf{y}$ as its first column and the subsequent $\mathbf{y}$-residuals in the corresponding subsequent columns.

An explicit calculation of the $\mathbf{X}$-regression coefficients $\hat{\beta}$ is obtained by

$$\hat{\beta} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{q},$$

where $\mathbf{q} = [q_1 \ldots q_k]'$ is the vector of $\mathbf{y}$-loadings. A MATLAB implementation of the exact algorithmic steps following the above description is given in Supporting Information S3.

2.2 | Much faster cross-validation

In the following, we will not assume that the data matrix $\mathbf{X}$ is centred. Also, for the uncentred $\mathbf{X}$, the matrix product $\mathbf{C} = \mathbf{X}\mathbf{X}'$ has a simple and advantageous structure. Note that if $\mathbf{X}_{-i}$ denotes the matrix obtained by deleting the n_i rows of the ith cross-validation segment from the original matrix $\mathbf{X}$, the associated matrix product $\mathbf{C}_{-i} = \mathbf{X}_{-i}\mathbf{X}'_{-i}$ can be obtained without performing additional calculations. $\mathbf{C}_{-i}$ is directly available by indexing into the full $\mathbf{C}$, ie, by ignoring the rows and columns of $\mathbf{C}$ corresponding to the ith cross-validation segment. (This is the same type of lookup approach that we used to obtain the variable subset calculations in Stefansson et al[14] based on the ”opposite” kernel matrix $\mathbf{X}'\mathbf{X}$.)

Because the mean centred version of $\mathbf{X}_{-i}$ is given by $\mathbf{X}^*_{-i} = \mathbf{X}_{-i} - \mathbf{1} \cdot \bar{\mathbf{x}}_{-i}$, where $\bar{\mathbf{x}}_{-i}$ is the corresponding row vector of the $\mathbf{X}_{-i}$ column means, the associated (centred) matrix product $\mathbf{C}^*_{-i} = \mathbf{X}^*_{-i}\mathbf{X}^{*\prime}_{-i}$ is given by the expression

$$\begin{aligned}\mathbf{C}^*_{-i} &= (\mathbf{X}_{-i} - \mathbf{1}\cdot\bar{\mathbf{x}}_{-i})(\mathbf{X}_{-i} - \mathbf{1}\cdot\bar{\mathbf{x}}_{-i})' \\ &= \underbrace{\mathbf{C}_{-i}}_{\text{term 1}} - \underbrace{(\mathbf{X}_{-i}(\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})'}_{\text{term 2}} + \underbrace{(\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})\mathbf{X}'_{-i})}_{\text{term 3}} + \underbrace{(\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})(\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})'}_{\text{term 4}}.\end{aligned} \tag{3}$$

Each of the terms 1 to 4 in Equation (3) can be obtained by indexing into precalculated matrices based on the full (uncentred) data matrix $\mathbf{X}$. In particular, by indexing into the precalculated $\mathbf{C} = \mathbf{X}\mathbf{X}'$, we can quickly obtain the matrix updates associated with any partition of the data set into the calibration and validation subsets associated with the various segments of a cross-validation procedure.

The corresponding updates for the validation samples are obtained by complementary indexing in the precalculated matrices to assure the correct centring with respect to the calibration samples included in the model building. Note that the third term in Equation (3) is identical to the transpose of the second term, ie, they share the same precalculations. If we ignore the constant vector $\mathbf{1}$, which is only required for expanding to the correct number of (identical) rows, we only need the calculations of $\mathbf{X}_{-i}\bar{\mathbf{x}}'_{-i}$, with dimensions $(n - n_i \times p)\cdot(p \times 1) = (n - n_i \times 1)$ for each cross-validation segment. For the fourth term, we only have to store the single scalar $\bar{\mathbf{x}}_{-i}\bar{\mathbf{x}}'_{-i}$ for each cross-validation segment (the associated $\mathbf{1}$-vectors represent no additional information required for storing).

2.3 | Segmented cross-validation

In the general setting, with K cross-validation segments of arbitrary sizes, computation of the column means

$$\bar{\mathbf{X}} = \begin{bmatrix} \bar{\mathbf{x}}_{-1} \\ \bar{\mathbf{x}}_{-2} \\ \vdots \\ \bar{\mathbf{x}}_{-K} \end{bmatrix}$$

associated with the matrices $\mathbf{X}_{-1}, \mathbf{X}_{-2}, \ldots, \mathbf{X}_{-K}$ requires successively omitting each of the K segments from the mean calculations. This can be achieved by first doing a summation of all rows in $\mathbf{X}$ into a row vector $S_\mathbf{X}$. With $\mathbf{X}_i$ denoting the n_i samples of the ith segment and $S_{\mathbf{X}_i}$ the corresponding vector obtained by summation of the rows in $\mathbf{X}_i$, we have $\bar{\mathbf{x}}_{-i} = (S_\mathbf{X} - S_{\mathbf{X}_i})/(n - n_i)$. Alternatively, one can create a dummy matrix $\mathbf{D}$ representing the vector of cross-validation segments and a vector $\mathbf{l}$ representing the segment lengths (n_i), to calculate $\bar{\mathbf{X}} = (\mathbf{1}S_\mathbf{X} - \mathbf{D}'\mathbf{X})/(\mathbf{l}\,\mathbf{1})$ in a single pass (using element-wise division). On the basis of the column means in $\bar{\mathbf{X}}$ we calculate

$$\begin{aligned}\mathbf{M} &= \mathbf{X}\bar{\mathbf{X}}' \\ \mathbf{m} &= (\bar{\mathbf{X}} \odot \bar{\mathbf{X}}) \cdot \mathbf{1},\end{aligned} \tag{4}$$

where the operator $\odot$ denotes the Hadamard product (ie, element-wise multiplication) of two matrices, and we obtain the required lookups for Equation (3) and calculation of the fast CV results.

All the quantities needed to create the K versions (one for each cross-validation segment) of Equation (3) are available by indexing into the matrices $\mathbf{C}$ (term 1), $\mathbf{M}$ (term 2/3), and $\mathbf{m}$ (term 4). By applying the indexing corresponding to a segmented CV procedure (positive index (i) means: use values calculated from the ith segment only, negative index ($-i$) means: use values calculated by omitting the ith segment), and Equation (3) corresponds to the following lookups:

$$\mathbf{C}^*_{-i} = \underbrace{\mathbf{C}_{-i}}_{\text{term 1}} - (\underbrace{\mathbf{M}_{-i,i}\mathbf{1}'}_{\text{term 2}} + \underbrace{\mathbf{1}\mathbf{M}'_{-i,i}}_{\text{term 3}}) + \underbrace{\mathbf{1}\mathbf{m}_i\mathbf{1}'}_{\text{term 4}}. \tag{5}$$

For validation purposes, the matrix $\mathbf{C}^*_i = \mathbf{X}^*_i\mathbf{X}^{*\prime}_{-i}$ is also required and can be obtained by the following fast lookups/calculations:

$$\begin{aligned}\mathbf{C}^*_i &= (\mathbf{X}_i - \mathbf{1}\cdot\bar{\mathbf{x}}_{-i})(\mathbf{X}_{-i} - \mathbf{1}\cdot\bar{\mathbf{x}}_{-i})' \\ &= \mathbf{X}_i\mathbf{X}'_{-i} - (\mathbf{X}_i(\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})' + (\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})\mathbf{X}'_{-i}) + (\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})(\mathbf{1}\cdot\bar{\mathbf{x}}_{-i})' \\ &= \mathbf{C}_{i,-i} - (\mathbf{1}\mathbf{M}_{i,i}\mathbf{1}' + \mathbf{1}\mathbf{M}'_{-i,i}) + \mathbf{1}\mathbf{m}_i\mathbf{1}'.\end{aligned} \tag{6}$$

The rationale for using this particular indexing is that we can retain rows corresponding to segment i of $\mathbf{C}$ (instead of removing them) while still using the mean of the training data for centring. As can be seen, the precalculated elements are the same as for the training, only differing in indexing.

From $\mathbf{C}^*_i$, the predictions for the held-out samples can be obtained directly without calculating the PLSR regression coefficients, loading weights, or loadings. This follows by analysing the PLSR regression coefficient expression $\hat{\boldsymbol{\beta}} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{q}$ and the associated prediction(s)

$$\hat{\mathbf{y}}_i = \mathbf{X}^*_i\hat{\beta} = \underbrace{\mathbf{X}^*_i\mathbf{W}}_{\text{term A}}\Big(\underbrace{\mathbf{P}'\mathbf{W}}_{\text{term B}}\Big)^{-1}\mathbf{q}. \tag{7}$$

The factors of term A in Equation (7) are $\mathbf{X}^*_i$ and $\mathbf{W} \propto \mathbf{X}^{*\prime}_{-i}\mathbf{R}_\mathbf{y}$, which can be (proportionally) expressed as $\mathbf{X}^*_i\mathbf{W} \propto \mathbf{X}^*_i\mathbf{X}^{*\prime}_{-i}\mathbf{R}_\mathbf{y} = \mathbf{C}^*_i\mathbf{R}_\mathbf{y}$. The term B can similarly be (proportionally) expressed as $\mathbf{P}'\mathbf{W} \propto \mathbf{T}'\mathbf{X}^*_{-i}\mathbf{X}^{*\prime}_{-i}\mathbf{R}_\mathbf{y} = \mathbf{T}'\mathbf{C}^*_{-i}\mathbf{R}_\mathbf{y}$. It can be shown that the required scaling factors in the indicated proportional substitutions of the terms A and B are identical and thus cancel out because of the matrix inversion in (7). Therefore, the prediction(s) of the held-out sample(s) are alternatively given by

$$\hat{\mathbf{y}}_i = \mathbf{C}^*_i\mathbf{R}_\mathbf{y}(\mathbf{T}'\mathbf{C}^*_{-i}\mathbf{R}_\mathbf{y})^{-1}\mathbf{q}. \tag{8}$$

For a k-component PLSR model, considerable computational savings can be obtained by exchanging the explicit calculations of $\mathbf{W}$ and $\mathbf{P}$ (both of dimension $(p \times k)$) with indexing into $\mathbf{C}^*$ and subsequently multiplying matrices of smaller dimensions (for $n \ll p$). Additional redundancies can be avoided by representing $\mathbf{q}$ as a (sparse) diagonal matrix and preforming cumulative summation over the extracted components to calculate predictions for all the PLSR submodels simultaneously. The predictions obtained by Equation (8) are experimentally verified to be as numerically stable as those obtained from the NIPALS algorithm; see Section 3 below.

2.3.1 | The leave-one-out cross-validation

Leave-one-out cross-validation (LooCV), where only a single sample successively is held aside for validation (ie, $n_i = 1$ in each cross-validation segment), is among the most popular validation strategies. In this particular case, the computation of $\bar{\mathbf{X}}$ can be expressed as $\bar{\mathbf{X}} = (\mathbf{1}(\mathbf{1}'\cdot\mathbf{X}) - \mathbf{X})/(n-1)$. The calculations of $\mathbf{M}$, $\mathbf{m}$, $\mathbf{C}^*_{-i}$, and $\mathbf{C}^*_i$ follow the general description, but since the predictions are restricted to single samples, three of the $\mathbf{1}$-vectors vanish in $\mathbf{C}^*_i = \mathbf{C}_{i,-i} - (\mathbf{M}_{i,i}\mathbf{1}' + \mathbf{M}'_{-i,i}) + \mathbf{m}_i\mathbf{1}'$.

2.4 | The fast double cross-validation

In the same fashion as executing ordinary cross-validation by indexing into $\mathbf{C} = \mathbf{XX}'$, some additional bookkeeping allows for the same approach to obtain fast calculations for the double/nested cross-validation strategy. In a double cross-validation, an outer loop is used for the validation of candidate models that are identified in the inner loop. Therefore, the model selection part of double cross-validation is kept separated from the model validation, to provide a more realistic estimation of the predictive properties of the alternative PLS models. An adaptation that simplifies the required bookkeeping considerably is to keep the inner cross-validation segments as subsets of the outer cross-validation segments. More specifically, we partition the dataset into a number of subsets. Repeatedly, one subset is held aside for validation, while the remaining subsets are used as traditional cross-validation segments when executing the inner cross-validation loop.

2.5 | Multiresponse PLSR

Extension of the above framework to PLSR modelling with $r > 1$ columns in the response matrix $\mathbf{Y}$ is not completely straightforward and requires a choice of strategy guiding the calculation of the PLSR loading weights and corresponding score vectors.

Each step of the PLS2 method extracts the dominant component of the singular value decomposition (SVD) of $\mathbf{X}'\mathbf{Y}$ (where $\mathbf{Y}$ is repeatedly deflated). This corresponds to $(\mathbf{X}, \mathbf{Y})$-covariance maximization and works well also when including a large number of response variables. The PLS2 is therefore a useful alternative for problems with high dimensional responses such as spectroscopic data, eg, when the goal is to predict the measurements of one spectroscopic method on the basis of another method.

The multiresponse version of the PKPLS is obtained by calculating the dominant right singular vector ($\mathbf{v}$) of $\mathbf{Y}'\mathbf{C}\mathbf{Y}$ to form $\mathbf{t}^\star = \mathbf{C}\mathbf{Y}\mathbf{v}$ and thereafter the score-vector $\mathbf{t}$ obtained by orthogonalization and normalization of $\mathbf{t}^\star$ with respect to the previously extracted scores.

$$\begin{aligned}
\mathbf{Y}_0 &= \mathbf{Y} \\
\mathbf{C} &= \mathbf{X}\mathbf{X}' \\
\mathbf{T} &= [\ ] && \text{(empty array for appending the orthogonal Scores)} \\
\mathbf{R}_\mathbf{Y} &= [\ ] && \text{(empty array for appending the y-residuals)} \\
&\text{---} \\
\mathbf{v} &\Leftarrow \text{SVD}(\mathbf{Y}'_{i-1}\mathbf{C}\mathbf{Y}_{i-1}) && \text{(dominant right singular vector)} \\
\mathbf{t}^* &= \mathbf{C}\mathbf{Y}_{i-1}\mathbf{v} && \text{(non-orthogonal Score)} \\
\mathbf{t}_i &= (\mathbf{t}^* - \mathbf{T}(\mathbf{T}'\mathbf{t}^*))/\|\mathbf{t}^* - \mathbf{T}(\mathbf{T}'\mathbf{t}^*)\| && \text{(deflate and normalize } \mathbf{t}^*\text{)} \\
\mathbf{T} &= [\mathbf{T}\ \ \mathbf{t}_i] && \text{(append } \mathbf{t}_i\text{)} \\
\mathbf{R}_\mathbf{Y} &= [\mathbf{R}_\mathbf{Y}\ \ \mathbf{Y}_{i-1}\mathbf{v}] && \text{(append } \mathbf{Y}_{i-1}\mathbf{v}\text{)} \\
\mathbf{q}_i &= \mathbf{t}_i'\mathbf{Y}_{i-1} && \text{(}\mathbf{Y}\text{-loadings)} \\
\mathbf{Y}_i &= \mathbf{Y}_{i-1} - \mathbf{t}_i q_i && \text{(}\mathbf{Y}\text{-deflation)}
\end{aligned} \tag{9}$$

Here, $\mathbf{C} = \mathbf{X}\mathbf{X}'$ is kept fixed, while the responses $\mathbf{Y}$ are deflated after each extracted component ($\mathbf{t}$). For the multiresponse case, the required columns of $\mathbf{R}_\mathbf{Y}$ are given by $\mathbf{Y}\mathbf{v}$ and the $\mathbf{Y}$-loadings ($\mathbf{Q}$) and regression coefficients ($\hat{\beta}$) become matrices instead of vectors. The remaining calculations coincide with those of the single-response case.

Cross-validation with multiple responses are based on exactly the same building blocks as the single-response case (except for the matrix representation $\mathbf{Q}$ of the $\mathbf{Y}$-loadings). A minor modification in calculating the final predictions is required to account for all the dimensions (#samples × #components × #responses). An example of how to handle the extra dimension due to multiple responses is given in the implementation of regression coefficients in Supporting Information S3.

It should be noted that a repeated single-response modelling strategy (considering one $\mathbf{Y}$-column at the time) usually works better than a joint multiresponse approach if good prediction is the main goal of the modelling. The extension to multiclass classification problems is straightforward using the one-vs-all groupwise dummy regression approach. Also in this case, the $\mathbf{C}$ matrix can be used for indexing across all responses or all classes, reducing redundancy in calculations considerably. An example is included in Section 3.

2.6 | Accelerating other methods

In principle, the parsimonious kernel approach can be used with any data analysis method that can exploit the kernel expression $\mathbf{C} = \mathbf{X}\mathbf{X}'$. To be able to fully accelerate cross-validation, as with PLS, it is a prerequisite that the variable dimension of **X** does not have to be reconstructed during calculations, eg, explicit calculation and use of the p-dimensional regression coefficients must be avoided. An obvious example is the principal component analysis (PCA) and associated principal component regression (PCR).

The implementation of PCA most used in data analysis is based on the SVD of a centred **X** matrix. Scores and loadings are produced from left (**U**) and right (**V**) singular vectors combined with the singular values (**S**). For efficient computation of regression coefficients, we also use the vector form of the singular values (**s**):

$$\begin{aligned} \mathbf{USV}' &= SVD(\mathbf{X}) \\ \mathbf{T} &= \mathbf{US} && \text{(scores)} \\ \mathbf{P} &= \mathbf{V} && \text{(loadings)} \\ \beta_k &= \sum_{i=1}^{k} \mathbf{p}_i(\mathbf{y}'\mathbf{t}_i)s_i^{-2} && \text{(reg. coefs.),} \end{aligned} \tag{10}$$

where $\mathbf{p}_i$ and $\mathbf{t}_i$ are vectors of loadings and scores from **P** and **T**, respectively, and s_i is the ith singular value. The parameter k controls the number of principal components included in the regression coefficients. Utilizing the kernel expression $\mathbf{C} = \mathbf{XX}'$, this becomes

$$\begin{aligned} \mathbf{XX}' &= \mathbf{USV}'\mathbf{VSU}' = \mathbf{US}^2\mathbf{U} = \mathbf{US}^*\mathbf{U}, \\ \mathbf{US}^*\mathbf{U}' &= SVD(\mathbf{C}), \\ \{\mathbf{U}, \mathbf{S}^*\} &= \text{eigen}(\mathbf{C}), \\ \mathbf{T} &= \mathbf{U}\sqrt{\mathbf{S}^*} && \text{(scores),} \\ \mathbf{P} &= \mathbf{X}'\mathbf{T}^+ = \mathbf{X}'\mathbf{US}^{*-1} && \text{(loadings),} \\ \beta_k &= \sum_{i=1}^{k} \mathbf{p}_i(\mathbf{y}'\mathbf{t}_i)s^{*-1} && \text{(reg. coefs.).} \end{aligned} \tag{11}$$

For a symmetric nonnegative definite matrix such as **C**, the SVD and eigenvalue decomposition are equivalent (up to computational round-offs). The updating of **C** in the cross-validation loop is the same as with PLS, and the predictions can be obtained without calculations that involve the p dimensional loadings or regression coefficients:

$$\hat{\mathbf{y}}_i = \sum(\mathbf{C}_i^*\mathbf{U}) \odot ((\mathbf{y}'\mathbf{U}) \odot \mathbf{s}^{*-1}). \tag{12}$$

Although PCR can be accelerated for wide **X** matrices, the overall improvements in computational efficiency will be smaller than for PLS as the SVD/eigen calculations, and not the sub-setting or predictions, consume most of the required computational efforts.

2.7 | Preprocessing

Because of the use of the kernel expression $\mathbf{C} = \mathbf{XX}'$ to simplify the calculations, segment-specific preprocessing of variables in the cross-validation loop is not possible if the variables' scales are affected. For instance, if variable standardization (autoscaling) is needed, our approach requires that this is performed on the full data set before cross-validation. This may introduce a marginal information bleed and reduction of the intersegment variation. However, for the purpose of model complexity estimation (identifying the optimal number of components), we consider such issues to have negligible impact. In fact, this view is adopted by several commercial software packages to reduce computation time and code complexity. Similarly, methods like extended multiplicative signal correction[15] are also often applied before cross-validation by default.

Any preprocessing that is individually performed for each sample, like the standard normal variate, baseline correction, or Savitzky-Golay filters[16] (smoothed derivatives), can of course be applied to the full data set before the cross-validation without any risk of information bleed.

Parameter	Values
n (rows)	50, 100, 500
p (cols.)	10, 100, 1000, 10 000
#comp.	1, 2, 3, 5, 10, 15, 20

TABLE 1 Simulation parameters. All parameter combinations were simulated 200 times each, resulting in variations in the analysed data

3 | RESULTS

3.1 | Simulation

To ascertain the differences in speed and stability, simulations were performed for single-response modelling, multiple-response modelling (Supporting Information S1) and double/nested cross-validation comparing the proposed methods with established PLS versions. For completeness, the PCR extension is also included in Supporting Information S2. Table 1 shows the sizes of the **X** matrices, which were filled with uniform random numbers. The table also shows the number of PLS components extracted with the various PLS algorithms and strategies. **y** (and **Y**) matrices were obtained by drawing uniform random regression coefficients (maximum 50 coefficients) and multiplying these with a corresponding number of **X** columns, $\mathbf{y} = \mathbf{X}[1, \ldots, 50]\beta$.

In the first simulation, a single response is used. We show PKPLS with the quick cross-validation, from now on abbreviated PKPLS(QCV), PKPLS with ordinary cross-validation, NIPALS PLS, and PLS by the bidiag2-algorithm. We have also explored the kernel PLS with the opposite kernel $\mathbf{X'X}$. The latter was proposed as a fast alternative when the number of samples is larger than the number of variables[13] and thus should perform better in cases where $n > p$. However, because the bidiag2 consistently outperformed this alternative (and performed much better when **X** matrices were sampled to be wide $p > n$), we decided not to include its performance among the reported results.

Figure 1 shows that PKPLS(QCV) is consistently faster than the NIPALS PLS regardless of matrix size and number of components extracted, except for the extreme case of $n = 500$ and $p = 10$. The largest differences are observed for high numbers of predictors and the smallest are seen for high numbers of samples. PKPLS in ordinary cross-validation is also faster than the NIPALS except when only a few components are combined with high number of samples and few predictors. bidiag2 is consistently faster than the NIPALS and is the quickest algorithm for $n = 500$ and $p < 1000$. For practical applications, the most tangible difference is for p=10000 and $n = 500$ where PKPLS(QCV) takes a few tenths of a second up to 2 seconds, while both NIPALS and bidiag takes between 20 seconds and 10 minutes.

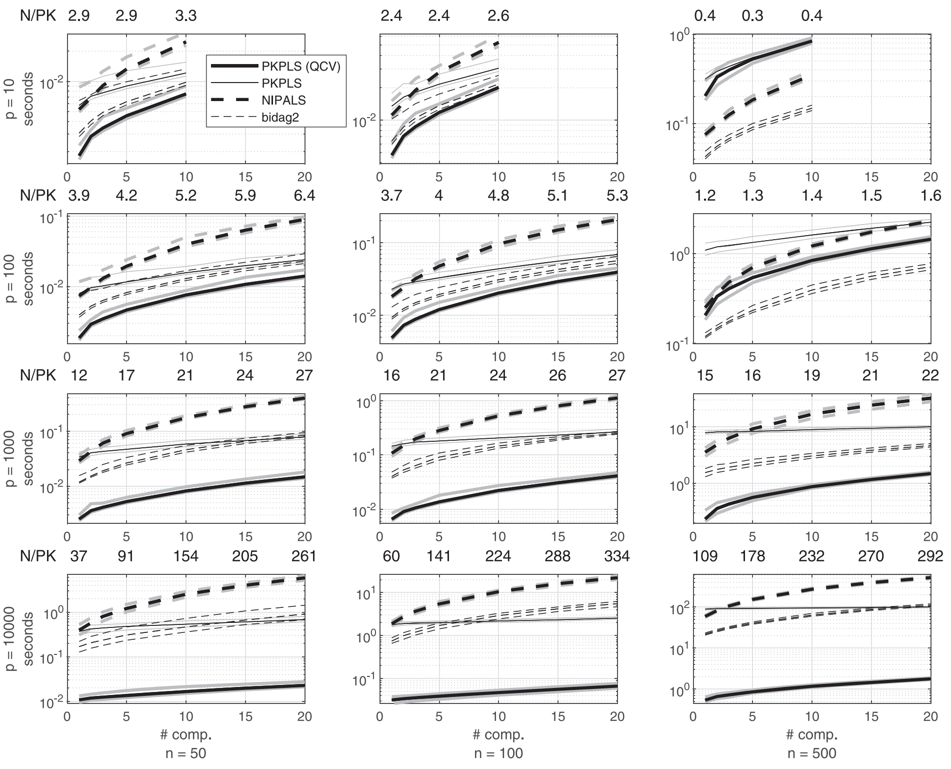

FIGURE 1 Single-response LooCV: Timing of cross-validation using uniform random matrices and component numbers according to Table 1 in single-response predictions. “N/PK” is the average ratio between time usage for NIPALS and PKPLS(QCV) after a given number of components. Black lines represent median time usage, while grey lines indicate the 5th and 95th percentiles of the simulations

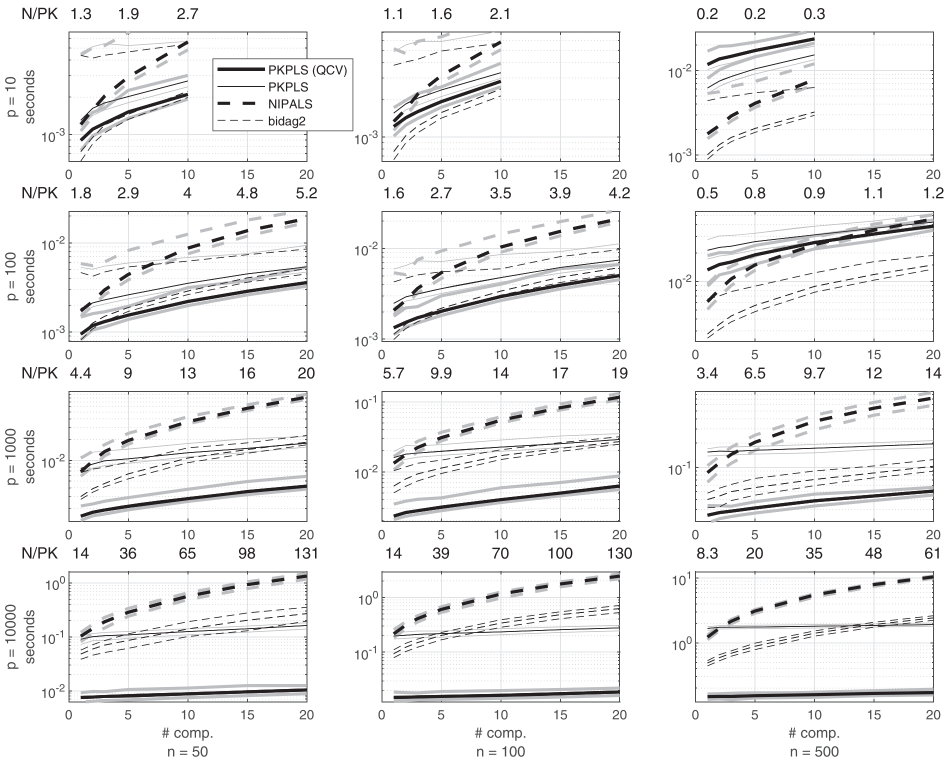

FIGURE 2 Single-response 10-fold CV: Timing of cross-validation using uniform random matrices and component numbers according to Table 1 in single-response predictions. "N/PK" is the average ratio between time usage for NIPALS and PKPLS(QCV) after a given number of components. Black lines represent median time usage, while grey lines indicate the 5th and 95th percentiles of the simulations

Another effect that can be observed is the higher initial cost of the ordinary cross-validation with PKPLS compared with quick cross-validation due to the explicit recalculation of $\mathbf{XX'}$. This manifests in NIPALS and PKPLS performing very similarly for many of the one-component models.

The numbers given in the sub-figure headings in Figure 1 show the average ratios of time consumption from using NIPALS and PKPLS(QCV). The dominant trend is that higher numbers of predictors favour PKPLS(QCV). For data with few predictors, an increase in the number of samples decreases the improvement using PKPLS(QCV), while data with more predictors gain more and more benefit from switching to PKPLS(QCV).

In Figure 2, we observe the same trends for 10-fold cross-validation as we saw in the previous figure using leave-one-out cross-validation. However, the reduction in time usage is of course smaller as the number of cross-validation segments are reduced from 50, 100, and 500, respectively, to 10 segments, thus reducing the reuse of $\mathbf{C} = \mathbf{XX'}$.

3.2 | Numerical stability

To get an impression of the numerical stability to be expected when using the PKPLS(QCV), we reuse a subset of the simulations from the previous subsection. Data with a predictor matrix of size (100 × 1000) and 1 or 10 responses have been simulated 2000 times, each extracting up to 50 components in leave-one-out cross-validation. The relative deviations are calculated as $\log \frac{|RMSECV_{NIPALS} - RMSECV_{PKPLS(2)}|}{RMSECV_{NIPALS}}$ and correspondingly with predictions. In Figure 3, the mean over the 2000 repetitions (and 10 responses) have been plotted.

As can be observed, the numerical differences in the predictions are appreciably small both in the single-response case and in the multiple-response case. Though a bit higher, the observed differences in RMSECV are still small enough (of magnitude 10^{-14} for double precision floating point calculations) to be ignored for any kind of practical purpose.

3.3 | Double cross-validation

In the particular double cross-validation setup we tested, the outer validation loop was a 10-fold cross-validation with consecutive segments. The inner loop was a ninefold cross-validation corresponding to the same sample sets as in the outer loop, ie, reusing the folds. The number of components to use for prediction in the outer loop was optimized to correspond to the minimum cross-validation error in the inner loop.

We generated a dataset of 500 samples and 100 000 variables having a single response as a random linear combination of the first 50 predictors, like in the previous simulations. Models were fitted with 20 components each. For comparison,

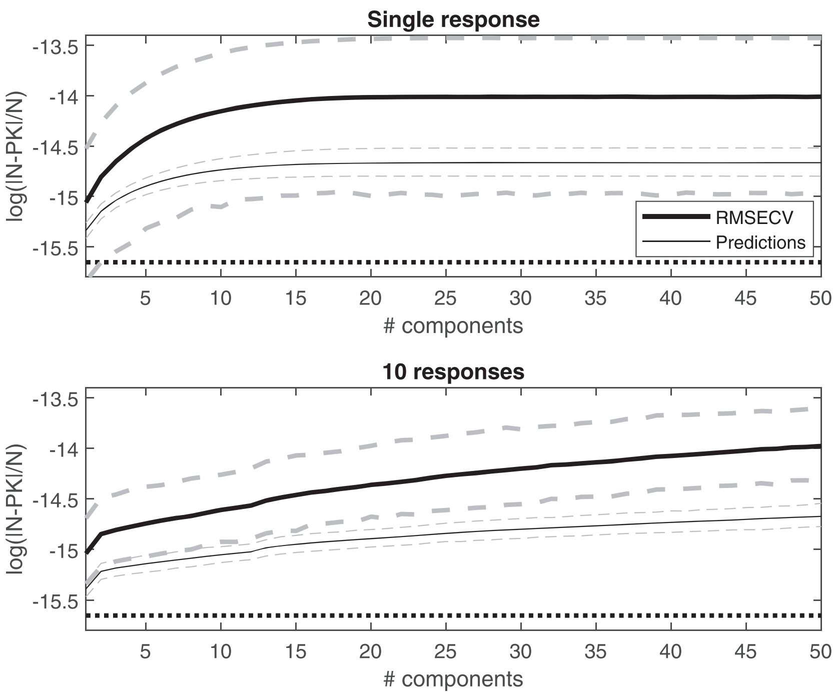

FIGURE 3 Single- and multi-response LooCV: Assessment of RMSECV and prediction accuracy for simulated data. "log(|N-PK|/N)" is the average relative deviation between NIPALS and PKPLS(2) after a given number of components. Dashed lines indicate the 5th and 95th percentiles in the simulations, and the dotted line indicates machine precision

we have also fitted PKPLS models with quick leave-one-out and 10-fold cross-validation. Relative time usage compared with single NIPALS and PKPLS model fits without cross-validation are summarized in Table 2. On our computer, the single NIPALS fitting took around 6.2 seconds, while the PKPLS took around 0.8 seconds. It can be observed that all three choices of cross-validation (LooCV, 10-fold, and double) are quicker than fitting a single NIPALS model with factors ranging from almost 10 times faster for the 10-fold cross-validation to twice as fast for the full LooCV. The latter would have taken 500×6.2 seconds using NIPALS, ie, more than 50 minutes, compared with around 3 seconds using PKPLS(QCV). When the basis for comparison is a single PKPLS model, only 10-fold cross-validation is quicker, while LooCV and double cross-validation use 3.2 and 1.6 times more time.

Two interesting conclusions can be drawn from the above observations. First, the extensive reuse of the estimated $\mathbf{C} = \mathbf{XX}'$ matrix means that double cross-validation is faster than leave-one-out cross-validation for problems of this size. Second, avoiding explicit calculations of $\mathbf{X}$-loadings, loading weights, and regression coefficients in the cross-validation predictions, cross-validation can actually be quicker than single model fitting for certain sizes of data.

Recognizing that 100 000 variables is more than most practical cases encountered, the same setup was rerun with 10 000 variables. This reduced the impact of PKPLS(QCV) a bit, but LooCV still used only twice as much time as a single NIPALS model fit, and both 10-fold and double cross-validations remained substantially faster than a single NIPALS fitting.

3.4 | Multiclass classification

As a demonstration of the one-vs-all group-wise dummy regression approach possibilities, we consider an example from microbial genus identification. A consensus taxonomy data set (called contax.trim) for prokaryotes based on 16S rRNA[15] is used to generate data having 38 781 samples, 65 536 variables, and 1774 classes. Variables are created by counting the frequency of 8-mers, ie, how many times nucleotide sequences of length 8, eg, ACGTTGGC, are encountered in each 16S rRNA sequence. There are 1774 different genera in the data set (one taxonomic class above species), which we will try to classify.

Classification will be done by creating 1774 PLSR models of one genus vs all other genera, concatenating the predictions into an array of size (38 781×1774×c), where *c* is the number of components. The final choice of class for a sample will be

	PKPLS(QCV)		
	LooCV	**10-fold**	**10 × 9-fold**
NIPALS	0.49	0.11	0.25
PKPLS	3.2	0.75	1.6

TABLE 2 Relative time usage of 20 component, cross-validated models (LooCV, 10-fold, and double cross-validation) compared with single model (NIPALS and PKPLS). Data of size (500×100 000)

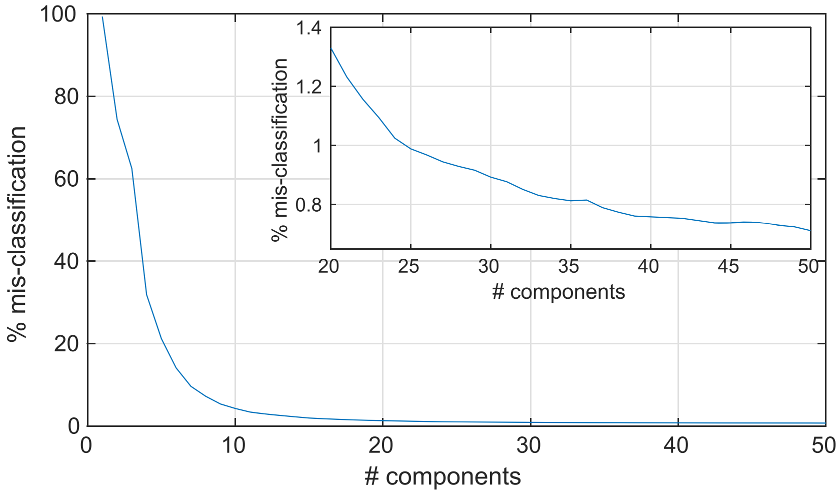

FIGURE 4 Percentage misclassified as a function of number of components in a one-vs-all group-wise dummy regression modelling of k-mers from the 16S rRNA of the contax.trim Prokaryote data set. A total of 156 out of 38 781 samples have been removed because of single sample classes (singletons). The inner sub-figure is a magnified view of the last 30 components

the column having the highest predicted value for each sample (argmax) for each component number, ie, the prediction having highest confidence. The procedure is similar to that in Vinje et al,[16] except for the argmax of predictions replacing nearest neighbour classification on predicted scores.

Figure 4 shows the percentage of misclassification with the contax.trim data set. The interesting point here is not that we are able to achieve the state-of-the-art level of precision, but that the cross-validation was conducted in 11 hours on a single 10 core processor. In comparison, only fitting (and predicting) a single 50 component one-vs-all model of PLSR takes approximately 255 seconds using the same computer. Scaling this with 1774 responses and 10-folds of cross-validation means it would require an estimated 52 days of computations to complete the cross-validation, ie, a speedup of 115 times. Adding overhead for argmax and summaries, this would take even longer. Classification using the multinom method,[17] which is highly specialized for the task, with the same cross-validation strategy and removal of singletons achieves a misclassification of 0.72% in less than 15 minutes, while we achieve a similar result (a misclassification of 0.71%) with our general purpose PLSR using 50 PLS components.

4 | DISCUSSION

We have proposed a new PLSR algorithm that is particularly useful for wide data matrices ($\mathbf{X}$), which applies several shortcuts and lookup techniques to reduce overhead and increase the computational speed of modelling and prediction. The speed advantages and stability have been thoroughly demonstrated by several applications on simulated data and a large real dataset.

On the basis of the simulations, we have observed that the proposed parsimonious kernel PLS with fast cross-validation outperforms the traditional NIPALS PLS regardless of size of the problems being analysed, unless the dataset contains a much higher number of observations than predictors ($n \gg p$). As can be expected, the largest gains of speed are obtained for datasets with a large number of predictors. Our simulations also revealed that the gain in speed is even more profound when also the number of samples is large. The bidiag2 algorithm was demonstrated to be consistently quicker than the NIPALS algorithm, but it only beats PKPLS in two of the tested combinations of data dimensions.

Our calculations in Section 2.5 for the multiresponse case were based on the SVD, leading to a PLS2 solution. An alternative approach based on canonical correlations (CCA) between $\mathbf{XX'Y}$ and $\mathbf{Y}$ yields the canonical PLS (CPLS).[18] The CPLS-alternative, utilizing the responses ($\mathbf{Y}$) twice in the extraction of components, often leads to more efficient subspace identification (fewer PLS components) but should be avoided if the number of responses, r, is "large" compared with the number of samples n. For the CCA/CPLS-alternative, the choice of deflation strategy (ie, deflating $\mathbf{X}$, $\mathbf{y}$, or both) will affect the model building results, which should be explored in more detail in a future publication.

The increased speed of the proposed algorithm can be exploited in several ways. It enables analysis of data sets having a vast amount of variables, eg, originating from higher spectral resolutions in spectroscopy, longer K-mer lengths in bioinformatics, more compounds in chromatographic/mass studies, concatenation of data sources, or inclusion of various transformations or types of preprocessing of the original data to elucidate various aspects of the samples. It can also be used to perform more thorough validation or more extensive exploration, eg, of sample subsets.

FUNDING

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

ORCID

Kristian Hovde Liland https://orcid.org/0000-0001-6468-9423
Petter Stefansson https://orcid.org/0000-0002-6079-6026
Ulf Geir Indahl https://orcid.org/0000-0002-3236-463X

REFERENCES

1. Wold S, Martens H, Wold H. *The multivariate calibration problem in chemistry solved by the PLS method*; 1983:286-293.
2. Wold S, Ruhe A, Wold H, and Dunn III WJ. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. *SIAM J Sci Stat Comput*, 5:735–743, 1984.
3. Martens H, Naes T. *Multivariate Calibration*. John Wiley & Sons; 1984.
4. Wold S, Sjöström M, Eriksson L. PLS-regression: a basic tool of chemometrics. *Chemom Intel Lab Syst*. 2001;58:109-130.
5. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *J Res Natl Bur Stand*. 1952;49:409–436. http://doi.org/10.6028/jres.049.044
6. Phatak A, de Hoog F. Exploiting the connection between PLS, Lanczos methods and conjugate gradients: alternative proofs of some properties of PLS. *J Chemometr*. 2002;16:361-367.
7. Golub G, Kahan W. Calculating the singular values and pseudo-inverse of a matrix. *J Soc Industrial Appl Math Series B Numer Anal*. 1965;2:205-224.
8. de Jong S. SIMPLS: An alternative approach to partial least squares regression. *Chemom Intel Lab Syst*. 1993;18:251-263.
9. Björck A, Indahl UG. Fast and stable partial least squares modelling: a benchmark study with theoretical comments. *J Chemometr*. 2017;31:e2898. https://doi.org/10.1002/cem.2898
10. Höskuldsson A. PLS regression methods. *J Chemometr*. 1988;2:211-228.
11. De Jong S, Ter Braak CJF. Comments on the PLS kernel algorithm. *J Chemometr*. 1994;8:169-174.
12. Rännar S, Lindgren F, Geladi P, Wold S. A PLS kernel algorithm for data sets with many variables and fewer objects. part 1: theory and algorithm. *J Chemometr*. 1994;8:111-125.
13. Dayal BS, MacGregor JF. Improved PLS algorithms. *J Chemometr*. 1997;11:73-85.
14. Stefansson P, Indahl UG, Liland KH, Burud I. Orders of magnitude speed increase in partial least squares feature selection with new simple indexing technique for very tall data sets. *J Chemometr*. 2019;e3141. https://doi.org/10.1002/cem.3141
15. Vinje H, Liland KH, and Snipen L. microcontax: The ConTax Data Package, 2016. R package version 1.0.
16. Vinje H, Liland KH, Almøy T, Snipen L. Comparing K-mer based methods for improved classification of 16S sequences. *BMC Bioinformatics*. 2015;16(1):205. https://doi.org/10.1186/s12859-015-0647-4
17. Liland KH, Vinje H, Snipen L. microclass: an R-package for 16S taxonomy classification. *BMC Bioinformatics*. 2017;18(1):172. https://doi.org/10.1186/s12859-017-1583-2
18. Indahl UG, Liland KH, Næs T. Canonical partial least squares–a unified PLS approach to classification and regression problems. *J Chemometr*. 2009;23:495-504.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of the article.

How to cite this article: Liland KH, Stefansson P, Indahl U G. Much faster cross-validation in PLSR-modelling by avoiding redundant calculations. *Journal of Chemometrics*. 2020;34:e3201. https://doi.org/10.1002/cem.3201