Statistics Surveys
Vol. 13 (2019) 119–149
ISSN: 1935-7516
https://doi.org/10.1214/19-SS125

PLS for Big Data: A unified parallel algorithm for regularised group PLS*

Pierre Lafaye de Micheaux

School of Mathematics and Statistics, UNSW Sydney, Australia
e-mail: lafaye@unsw.edu.au

Benoît Liquet

Laboratoire de Mathematiques et de leurs Applications
Université de Pau et des Pays de l'Adour, UMR CNRS 5142, E2S-UPPA, Pau, France
ARC Centre of Excellence for Mathematical and Statistical Frontiers and
School of Mathematical Science, Queensland University of Technology, Brisbane, Australia
e-mail: benoit.liquet@univ-pau.fr

and

Matthew Sutton

ARC Centre of Excellence for Mathematical and Statistical Frontiers and
School of Mathematical Science, Queensland University of Technology, Brisbane, Australia
e-mail: matt.sutton@qut.edu.au

Abstract: Partial Least Squares (PLS) methods have been heavily exploited to analyse the association between two blocks of data. These powerful approaches can be applied to data sets where the number of variables is greater than the number of observations and in the presence of high collinearity between variables. Different sparse versions of PLS have been developed to integrate multiple data sets while simultaneously selecting the contributing variables. Sparse modeling is a key factor in obtaining better estimators and identifying associations between multiple data sets. The cornerstone of the sparse PLS methods is the link between the singular value decomposition (SVD) of a matrix (constructed from deflated versions of the original data) and least squares minimization in linear regression. We review four popular PLS methods for two blocks of data. A unified algorithm is proposed to perform all four types of PLS including their regularised versions. We present various approaches to decrease the computation time and show how the whole procedure can be scalable to big data sets. The `bigsgPLS` R package implements our unified algorithm and is available at https://github.com/matt-sutton/bigsgPLS.

MSC 2010 subject classifications: Primary 6202, 62J99.

Keywords and phrases: High dimensional data, Lasso penalties, Partial Least Squares, Singular Value Decomposition.

Received August 2018.

*This is an original survey paper.

Contents

1. Introduction

In this article, we review the Partial Least Squares (PLS) approach to big data. When one faces the task of statistically analysing data resulting from the observation of cases on a large number of variables, interpretation of the results can be difficult. Analysing each variable separately is time consuming, and describing the results using graphs and numerical indicators is space consuming. One way to circumvent this problem is to select a few of the more important variables while discarding the others. However, selecting the best variables is not easy, not to mention that interesting interactions between these variables will remain unexplored. A better approach is to create a few new variables that combine in a clever way the original ones. Using *linear* combinations is a good strategy. Indeed, such new variables are easy to compute. (Being composed of all the original variables, they are called *components* and, being not directly observed, often also called "*latent variables*".) Furthermore, the weights of each component fully characterise the relationships between the new variables and the original ones, which makes interpretation of the former in terms of the latter easy. This approach is for example at the core of *Principal Components Analysis* (PCA), a statistical technique that computes weights (of each linear

combination) in such a way that the variance of each component is maximal, under the constraint that components are orthogonal. Geometrically, constructing the first component consists in finding the one-dimensional subspace (i.e., a line) such that projected data onto this subspace will have, after projection, maximal variance. (Broadly speaking, this line gives the direction in which the cloud of data points is most elongated.) The second component is obtained by finding another one-dimensional space that is orthogonal to the first one while maximising the variance of data projected onto the former. And so on for subsequent components, stopping after a small number of components have been extracted. In many cases, these few first components are sufficient to recover a large proportion of the overall multidimensional variability present in the original data set, thereby performing a reduction of dimension while keeping most of the information. In practice, these directions (given by the weights of each linear combination) are easily obtained thanks to a linear algebra tool called the Singular Value Decomposition (SVD); see (Meyer 2000, Figure 5.12.1, page 413) for a very nice geometric explanation of how the SVD of a matrix A describes the ellipsoid that one gets when a unit sphere is distorted by the linear transformation associated to A. Note that if the original data have been centered first, the requested orthogonality between components translates directly into their uncorrelatedness, rendering the information brought by each component independent from each other (at least in the ideal world of Gaussian distributions). This facilitates their interpretation. Also, an interesting graph often used in PCA, the so-called *correlation circle*, enables one to rapidly grasp how PCA components are linked (i.e., correlated) to the original variables, making interpretation in terms of the original variables possible.

The above description focuses on the analysis of a single set of data. Sometimes, however, one is interested in the simultaneous analysis of multiple blocks of data, each comprising a large number of variables. The strategy used in PCA can be extended to this situation, giving rise to a statistical method called *Projection to Latent Structures* (PLS). Originally developed by Wold (1966) as a set of iterative algorithms involving optimisation of quantities as in the classical *Least Squares* method, it was termed *Partial Least Squares* (PLS), a name still often used nowadays. PLS methods perform a wide range of multivariate supervised and unsupervised statistical techniques on multiple blocks of data. Their goal is thus to analyse multiple relationships among several sets of variables measured on the same objects. They construct new variables known as scores (or components), which are linear combinations of the original variables. Instead of maximising variances as in PCA, and since PLS focuses on relationships between sets of data, linear combinations are here obtained by maximising a covariance (or correlation) criterion. Once again, this maximisation is easily achieved in practice using an SVD; see Lafaye de Micheaux et al. (2017) for a rigorous mathematical description of this link along with its proof. The current article focuses on PLS modeling when there are only two blocks of data. One distinguishes between the case when the two blocks are interchangeable (symmetric situation) and when they are not (asymmetric situation). In the latter case, the goal is to predict one set of data from the other. (As a simple analogy,

think about "correlation between two variables" versus "simple linear regression between two variables".) In the two-block case, the PLS acronym (for Partial Least Squares or Projection to Latent Structures) usually refers to one of four related methods, which are detailed in Section 2.2:

(i) Partial Least Squares Correlation (PLSC) also called PLS-SVD (Krishnan et al. 2011, Abdi & Williams 2013, Rohlf & Corti 2000),
(ii) PLS in mode A (PLS-W2A, for Wold's Two-Block, Mode A PLS) (Vinzi et al. 2010, Cak et al. 2016, Wegelin 2000),
(iii) PLS in mode B (PLS-W2B) also called Canonical Correlation Analysis (CCA) (Guo & Mu 2013, Hardoon et al. 2004, Hotelling 1936), and
(iv) Partial Least Squares Regression (PLS-R, or PLS2) (Wold et al. 2001, Rosipal & Krämer 2006, Geladi & Kowalski 1986).

The first three methods model a symmetric relationship between the data, aiming to explain the shared correlation or covariance between the data sets, while the fourth method (PLS-R), models an asymmetric relationship, where one block of predictors is used to explain the other block. These methods are now widely used in many fields of science, such as genetics (Boulesteix & Strimmer 2007, Ji et al. 2011, Liquet et al. 2016), neuroimaging (McIntosh et al. 1996, Roon et al. 2014) and imaging-genetics (Lorenzi et al. 2016, Liu & Calhoun 2014).

Recently, some authors have started to modify these methods using sparse modelling techniques; see e.g., (Lê Cao et al. 2008, Dhanjal et al. 2009, Witten et al. 2009, Chung & Keleş 2010, Chun & Keleş 2010). These techniques refer to methods in which a relatively small number of covariates influence the model. They are powerful methods in statistics that provide improved interpretability and better estimators, especially for the analysis of big data. For example, in imaging-genetics, sparse models demonstrated great advantages for the identification of biomarkers, leading to more accurate classification of diseases than many existing approaches (Lin et al. 2014).

This is not really surprising. Indeed, as nicely summarised by Wegelin (2000): "The coefficients computed in a PLS analysis are well-defined and easy to interpret. PLS is especially useful when the columns of X or of Y are collinear or nearly collinear, or when there are more variables than observations ($p > n$ or $q > n$), since few other methods are available in such a case." Among the few statistical methods that can deal with two blocks of data, one can obviously mention classical multivariate linear regression (a direct competitor of case (iv) when $q > 1$). Nevertheless, this method fails in the presence of collinearity or when there are more variables than observations, since the least squares estimator is not unique in that case ($X^T X$ is not invertible). A way to circumvent this problem is to impose sparsity on the coefficients through a shrinkage lasso approach; see Friedman et al. (2010) and their associated R package `glmnet` (Friedman et al. 2018). By imposing the assumption of sparsity these methods provide stable estimates of the coefficients, and often have improved predictive performance. However, ultra high dimensional data sets ($p >> n$) pose a computational challenge in fitting these methods. This problem has only been addressed recently by Zeng & Breheny (2017*a*) but only for an univariate re-

sponse (see their R package `biglasso` (Zeng & Breheny 2017*b*)).

In this article, we consider big data extensions to PLS methods which incorporate sparsity in fitting the PLS weights. Similarly to standard PLS methods, these sparse approachs permit dimension reduction by constructing components as linear weighted combinations of the original data. However, sparse PLS approaches allow for some of these weights to be set to zero. Consequently, a variable in the original dataset might well be used in constructing only a subset of the PLS components. An advantage of having several components in PLS is to visualise the data projected on various 2D planes spanned by pairs of components (as illustrated on our Figure 3) in order to gain better insight into the underlying structure of the data. Also, these sparse approaches are often only considered for two blocks of data in a regression context, whereas standard PLS can deal with several blocks of data in a more general analysis. Closely related to Lasso regression is (multivariate) ridge regression (Brown & Zidek 1980) that shrinks the coefficients using a L^2 norm (instead of a L^1 norm for the Lasso). This different type of shrinkage does not lead to sparsity of the coefficients though, and consequently does not perform a reduction of dimension. Another potential competitor of PLS worth mentioning is Principal Component Regression (PCR). Once again, this method is only applicable in a regression context. It solves the problem of data collinearity and reduces the number of regressor variables by replacing the original regressor variables by a few orthogonal (hence non-collinear) principal components obtained via a PCA of X. Compared to PLS, these X-components are obtained independently of the response Y which seems to lead to lower prediction performances than PLS (Yeniay & Goktas 2002). A nice discussion that explores the relationships between PLS, PCR, Ridge and the Lasso can be found in (Hastie et al. 2009, Section 3.6). A last competitor method worth mentioning is the Vector Generalised Additive Model (VGAM); see Yee & Wild (1996) for details, and the `VGAM` R package (Yee 2018). This very flexible technique relates the response to the regressors through a sum of non-linear transformations of the original regressors. Results are difficult to interpret though, all the more when the number of variables is large. Finally, note that inference in PLS is done either by using resampling or Bootstrap techniques (see Abdi & Williams (2013, Chapter 23)) or by computation of its degrees of freedom (Kraemer & Sugiyama 2011) thanks to the R package `plsdof` (Nicole Kraemer 2018). Inference for the Lasso can also be done using resampling techniques, or by using a covariance test statistic (Lockhart et al. 2014) thanks to the R package `selectiveInference` (Tibshirani et al. 2017).

In Section 2, we survey the standard PLS methods. The optimisation criteria and algorithmic computation are detailed. Gathering an accurate description of all these methods in a single document, along with their complete mathematical proofs, constitutes a valuable addition to the literature; see also (Lafaye de Micheaux et al. 2017, Wegelin 2000). In Section 3 we present the sparse versions of the four types of PLS, as well as a recent group and sparse group version. A new unified algorithm is then presented in Section 4 to perform all four types of PLS including the regularised versions. Various approaches to decrease the

computation time are proposed. We explain how the whole procedure can be made scalable to big data sets (any number of measurements, or variables). In Section 5, we demonstrate the performance of the method on simulated data sets including the case of a categorical response variable. Our algorithm is implemented in the R programming language (R Core Team 2017) and is available at `https://github.com/matt-sutton/bigsgPLS` as a comprehensive package called `bigsgPLS` that implements parallel computations, either using several cores or a GPU device.

2. Partial Least Squares family

2.1. Notation

Let $\boldsymbol{X} := \boldsymbol{X}_0 : n \times p$ and $\boldsymbol{Y} := \boldsymbol{Y}_0 : n \times q$ be the two matrices (or "blocks") of data. They comprise n observations collected on p and q variables, respectively. Before analysis, the X and Y matrices are transformed by subtracting their column averages. Scaling each column by their standard deviation is also often recommended (Geladi & Kowalski 1986). To make explicit the columns of a $n \times r$ matrix $\boldsymbol{A}$, we write $\boldsymbol{A} := [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r] := (\boldsymbol{a}_j)$. We also note $\boldsymbol{A}_{\bullet h} := [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_h]$ for the submatrix of the first h columns ($1 \leq h \leq r$), and $\boldsymbol{A}_{\bullet \bar{h}} := [\boldsymbol{a}_{h+1}, \ldots, \boldsymbol{a}_r]$ for the remaining ones. For two zero-mean vectors $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ of the same size, we note $Cov(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) = \tilde{\boldsymbol{u}}^\intercal \tilde{\boldsymbol{v}}$ and $Cor(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) = \tilde{\boldsymbol{u}}^\intercal \tilde{\boldsymbol{v}} / \sqrt{(\tilde{\boldsymbol{u}}^\intercal \tilde{\boldsymbol{u}})(\tilde{\boldsymbol{v}}^\intercal \tilde{\boldsymbol{v}})}$. The scaling factor $(n-1)^{-1}$ is omitted. Let $\boldsymbol{X}^+$ be the Moore-Penrose (generalised) inverse of $\boldsymbol{X}$. We denote the space spanned by the columns of $\boldsymbol{X}$ by $\mathcal{I}(\boldsymbol{X})$. The orthogonal projection matrix onto $\mathcal{I}(\boldsymbol{X})$ is denoted $\mathcal{P}_{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{X}^+$, and $\mathcal{P}_{\boldsymbol{X}^\perp} = \boldsymbol{I} - \mathcal{P}_{\boldsymbol{X}}$ denotes the orthogonal projection matrix on the space orthogonal to $\mathcal{I}(\boldsymbol{X})$. When the inverse of $\boldsymbol{X}^\intercal \boldsymbol{X}$ exists, we have $\boldsymbol{X}^+ = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal$. The L_p vector norm ($p = 1, 2$) of an n-length vector $\boldsymbol{x}$, is $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$. The Frobenius norm of a $n \times r$ matrix $\boldsymbol{A}$ is $\|\boldsymbol{A}\|_F = \|\mathtt{vec}(\boldsymbol{A})\|_2$, where the `vec` operator transforms $\boldsymbol{A}$ into an $nr \times 1$ vector by stacking its columns. The soft thresholding function is $g^{\text{soft}}(x, \lambda) = \text{sign}(x)(|x| - \lambda)_+$, where $(a)_+ = \max(a, 0)$. Finally, $\otimes$ denotes the Kronecker product (Lütkepohl 2005, (3), p. 662).

2.2. The four standard PLS methods

In this subsection, we survey the four standard PLS methods (i)–(iv) introduced in Section 1. The PLS methods are used to iteratively construct a small number H of pairs of meaningful linear combinations $\boldsymbol{\xi}_h = \boldsymbol{X}_0 \boldsymbol{w}_h$ and $\boldsymbol{\omega}_h = \boldsymbol{Y}_0 \boldsymbol{z}_h$ ($h = 1, \ldots, H$), of the original X- and Y-variables, in the sense that they should have **maximal covariance** (or correlation). These linear combinations reveal the linear relationships between the two blocks of data. They are called *component scores*, or *latent variables*, while $\boldsymbol{w}_h$ and $\boldsymbol{z}_h$ are the (adjusted) *weights*. The construction of the components leads to decompositions of the original matrices

$\boldsymbol{X}$ and $\boldsymbol{Y}$ of the form:

$$\boldsymbol{X} = \boldsymbol{\Xi}_H \boldsymbol{C}_H^{\mathsf{T}} + \boldsymbol{F}_H^X, \qquad \boldsymbol{Y} = \boldsymbol{\Omega}_H \boldsymbol{D}_H^{\mathsf{T}} + \boldsymbol{F}_H^Y, \tag{1}$$

where $\boldsymbol{\Xi}_H = (\boldsymbol{\xi}_j)$, $\boldsymbol{\Omega}_H = (\boldsymbol{\omega}_j)$ are called the X- and Y-scores, $\boldsymbol{C}_H = (\boldsymbol{c}_j)$, $\boldsymbol{D}_H = (\boldsymbol{d}_j)$ are the X- and Y-loadings, and $\boldsymbol{F}_H^X$, $\boldsymbol{F}_H^Y$ are the residual matrices. The decomposition model states that the blocks $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be expressed as a weighted combination of H latent features plus some noise. For any set of X and Y-scores we have the decomposition:

$$\boldsymbol{X} = \mathcal{P}_{\boldsymbol{\Xi}_H} \boldsymbol{X} + \mathcal{P}^{\perp}_{\boldsymbol{\Xi}_H} \boldsymbol{X}, \quad \text{and,} \quad \boldsymbol{Y} = \mathcal{P}_{\boldsymbol{\Omega}_H} \boldsymbol{Y} + \mathcal{P}^{\perp}_{\boldsymbol{\Omega}_H} \boldsymbol{Y}.$$

Thus the elements of the decomposition model can always be written as

$$\begin{aligned} \boldsymbol{C}_H &= \boldsymbol{X}^{\mathsf{T}} \boldsymbol{\Xi}_H^{+}, \quad \boldsymbol{D}_H = \boldsymbol{Y}^{\mathsf{T}} \boldsymbol{\Omega}_H^{+}, \\ \boldsymbol{F}_H^X &= \mathcal{P}^{\perp}_{\boldsymbol{\Xi}_H} \boldsymbol{X}, \quad \boldsymbol{F}_H^Y = \mathcal{P}^{\perp}_{\boldsymbol{\Omega}_H} \boldsymbol{Y}, \end{aligned}$$

which may be simplified when certain orthogonality properties hold for $\boldsymbol{\Xi}_H$ and $\boldsymbol{\Omega}_H$. We can interpret the X and Y loading matrices $\boldsymbol{C}_H$ and $\boldsymbol{D}_H$ as regression coefficients in the reconstruction of the X and Y blocks. In this way, the size of the loadings describes the importance of the score for reconstructing the original data.

Although the four PLS methods all conform to the unified framework, the methods provide different loadings and error matrices due to the calculation of the scores. We now detail the four classical cases (i)-(iv). For each method, the scores may be calculated as linear combinations $\boldsymbol{\xi}_h = \boldsymbol{X}\boldsymbol{w}_h$ and $\boldsymbol{\omega}_h = \boldsymbol{Y}\boldsymbol{z}_h$. We state the PLS objective functions in terms of the data $\boldsymbol{X}$ and $\boldsymbol{Y}$ for the adjusted weights $\boldsymbol{w}_h$ and $\boldsymbol{z}_h$ (or $\tilde{\boldsymbol{w}}_h$ and $\tilde{\boldsymbol{z}}_h$ for unnormalised versions). We comment on the focus of the analysis in each of these methods and survey important properties. Additional technical details may be found in Wegelin (2000) and Lafaye de Micheaux et al. (2017).

(i) For PLS-SVD, the roles of $\boldsymbol{X}$ and $\boldsymbol{Y}$ are symmetric, and the analysis focuses on modeling shared information (rather than prediction) as measured by the cross-product matrix $\boldsymbol{R} = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{Y}$. Note that $\boldsymbol{R}$ is proportional to the empirical covariances between X- and Y-variables when the columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$ are centered. When the columns are standardised, $\boldsymbol{R}$ is proportional to the empirical correlations between X- and Y-variables. In this setting, the method is sometimes called PLSC, for Partial Least Squares Correlation (Krishnan et al. 2011).
For PLS-SVD, the weights at step h, $(\boldsymbol{w}_h, \boldsymbol{z}_h)$, are defined as the solution to the optimisation problem:

$$\underset{\|\boldsymbol{w}\|_2 = \|\boldsymbol{z}\|_2 = 1}{\text{maximise}} \; Cov(\boldsymbol{X}\boldsymbol{w}, \boldsymbol{Y}\boldsymbol{z}), \quad \text{s.t. } \boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}_j = \boldsymbol{z}^{\mathsf{T}}\boldsymbol{z}_j = 0,$$

for $1 \leq j < h$. PLS-SVD searches for orthonormal directions $\boldsymbol{w}_h$ and orthonormal directions $\boldsymbol{z}_h$ $(h = 1, \ldots, H)$, such that the score vectors have

maximal covariance. In contrast to the other PLS methods, the X-scores $\boldsymbol{\xi}_h$ and Y-scores $\boldsymbol{\omega}_h$ are in general not mutually orthogonal (Rosipal & Krämer 2006) which can make interpretation of the results more delicate.

(ii) For PLS-W2A, the weights at step h, $(\tilde{\boldsymbol{w}}_h, \tilde{\boldsymbol{z}}_h)$, are defined as the solution to the optimisation problem:

$$\underset{\|\mathcal{P}^{\perp}_{\tilde{\boldsymbol{W}}_{\bullet h-1}}\tilde{\boldsymbol{w}}\|=\|\mathcal{P}^{\perp}_{\tilde{\boldsymbol{Z}}_{\bullet h-1}}\tilde{\boldsymbol{z}}\|=1}{\text{maximise}} \quad Cov(\boldsymbol{X}\tilde{\boldsymbol{w}}, \boldsymbol{Y}\tilde{\boldsymbol{z}}),$$
$$\text{s.t.} \quad Cov(\boldsymbol{\xi}_h, \boldsymbol{\xi}_j) = Cov(\boldsymbol{\omega}_h, \boldsymbol{\omega}_j) = 0,$$

for $1 \leq j < h$. PLS-W2A thus searches for successive X-score vectors (resp. Y-score vectors) that are orthogonal to the previous ones. The first pair $(\boldsymbol{\xi}_1, \boldsymbol{\omega}_1)$ of X- and Y- score vectors is the one with maximal covariance. The next pairs are searched for using successively deflated (i.e., after removing the information contained in the previous pairs of scores) versions of $\boldsymbol{X}_0$ and $\boldsymbol{Y}_0$.

(iii) For CCA, the weights at step h, $(\tilde{\boldsymbol{w}}_h, \tilde{\boldsymbol{z}}_h)$, are defined as the solution to the optimisation problem:

$$\begin{aligned} &\underset{\tilde{\boldsymbol{w}}\in\mathbb{R}^p,\ \tilde{\boldsymbol{z}}\in\mathbb{R}^q}{\text{maximise}} \quad Cor(\boldsymbol{X}\tilde{\boldsymbol{w}}, \boldsymbol{Y}\tilde{\boldsymbol{z}}), \\ &\text{subject to} \quad Cov(\boldsymbol{X}\tilde{\boldsymbol{w}}, \boldsymbol{X}\tilde{\boldsymbol{w}}_j) = Cov(\boldsymbol{Y}\tilde{\boldsymbol{z}}, \boldsymbol{Y}\tilde{\boldsymbol{z}}_j) = 0, \end{aligned}$$

for $1 \leq j < h$. Classical CCA relates $\boldsymbol{X}$ and $\boldsymbol{Y}$ by maximising the *correlation* between the scores (also called canonical variates) $\boldsymbol{\xi}_h = \boldsymbol{X}\tilde{\boldsymbol{w}}_h$ and $\boldsymbol{\omega}_h = \boldsymbol{Y}\tilde{\boldsymbol{z}}_h$, but without imposing a unit norm to the adjusted weights (or canonical) vectors $\tilde{\boldsymbol{w}}_h$ and $\tilde{\boldsymbol{z}}_h$. Using the change of variables $\tilde{\boldsymbol{w}} = (\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1/2}\boldsymbol{w}$ and $\tilde{\boldsymbol{z}} = (\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y})^{-1/2}\boldsymbol{z}$ we have the following equivalent objective for $(\boldsymbol{w}_h, \boldsymbol{z}_h)$:

$$\begin{aligned} &\underset{\|\boldsymbol{w}\|=\|\boldsymbol{z}\|=1}{\text{maximise}} \quad Cov(\boldsymbol{X}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1/2}\boldsymbol{w}, \boldsymbol{Y}(\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y})^{-1/2}\boldsymbol{z}), \\ &\text{subject to} \quad \boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}_j = \boldsymbol{z}^{\mathsf{T}}\boldsymbol{z}_j = 0, \end{aligned}$$

for $1 \leq j < h$. However, when $p > n$, this optimisation is not feasible because $(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}$ and $(\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y})^{-1}$ are not uniquely defined. A common approach to this problem is to use the Moore-Penrose inverse (Mardia et al. 1979, p. 287)(Nielsen 2002, p. 75). However, this approach can produce a meaningless solution, with correlations trivially equal to one. Moreover, in this case, a small change in the data can lead to large changes in the weights and scores (Wegelin 2000, pp. 26–27).

An alternative approach is to perform regularisation on the sample covariance matrices. Regularisation was first introduced to the CCA method by Vinod (1976) and later refined by S. E. Leurgans (1993). This method is known as regularised CCA (rCCA) or canonical ridge analysis. This regularisation is imposed by replacing the matrices $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$ and $\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}$ with $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X} + \lambda_x\boldsymbol{I}_p$ and $\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y} + \lambda_y\boldsymbol{I}_q$ respectively in the optimisation criterion.

The regularisation parameters λ_x and λ_y should be nonnegative and if they are nonzero, then the regularised covariance matrices may be nonsingular. Note that ordinary CCA is obtained at $\lambda_x^* = \lambda_y^* = 0$, and PLS-SVD is obtained with $\lambda_x^* = \lambda_y^* = 1$. Other approaches exist; see e.g., (Witten et al. 2009, eq. (13)).

(iv) PLS-R (also called PLS1 if $q = 1$ or PLS2 if $q > 1$) is a regression technique that predicts one set of data from another, hence termed asymmetric, while describing their common structure. It finds latent variables (also called component scores) that model $\boldsymbol{X}$ and simultaneously predict $\boldsymbol{Y}$. While several algorithms have been developed to solve this problem, we focus on the two most well known variants. The first is an extension of the Nonlinear estimation by Iterative PArtial Least Squares (NIPALS), modified by Wold et al. (1984) to obtain a regularised component based regression tool. The second is the Statistically Inspired Modification of PLS (SIMPLS) developed by de Jong (1993). Other PLS regression algorithms can be found in Lindgren & Rännar (1998), and see also Alin (2009) for a numerical comparison.

The h-th set of PLS regression weights $(\tilde{\boldsymbol{w}}_h, \boldsymbol{z}_h)$ given by NIPALS solve the optimisation problem (ter Braak & de Jong 1998, eq. (15))

$$\underset{\|\mathcal{P}^{\perp}_{\tilde{\boldsymbol{W}}_{\bullet h-1}}\tilde{\boldsymbol{w}}\|=\|\boldsymbol{z}\|=1}{\text{maximize}} \; Cov(\boldsymbol{X}\tilde{\boldsymbol{w}}, \boldsymbol{Y}\boldsymbol{z}), \qquad \text{subject to} \;\; Cov(\boldsymbol{X}\tilde{\boldsymbol{w}}, \boldsymbol{X}\tilde{\boldsymbol{w}}_j) = 0,$$

for $1 \leq j < h$. The first pair $(\boldsymbol{\xi}_1, \boldsymbol{\omega}_1)$ of X- and Y- score vectors is the one with maximal covariance. The next pairs are searched for using successively deflated versions of $\boldsymbol{X}_0$ and $\boldsymbol{Y}_0$.

Two equivalent versions of the NIPALS algorithm are found in the literature; whether $\boldsymbol{z}_h$ is scaled (Höskuldsson 1988), or not (Wold et al. 1984, Tenenhaus 1998). We note that both algorithms provide equivalent regression parameters, and only differ in the calculation of the Y-scores and loadings. At the end of both algorithms, the fitted values $\widehat{\boldsymbol{Y}}_H$ are computed (Phatak & de Jong 1997, Equ. (20))

$$\widehat{\boldsymbol{Y}}_H = \mathcal{P}_{\boldsymbol{\Xi}_H}\boldsymbol{Y} = \boldsymbol{\Xi}_H(\boldsymbol{\Xi}_H^{\mathsf{T}}\boldsymbol{\Xi}_H)^{-1}\boldsymbol{\Xi}_H^{\mathsf{T}}\boldsymbol{Y}.$$

In addition to the usual decomposition equations (1) that will be explicated below, the PLS regression algorithm includes an additional "inner relationship" which relates the Y-scores $\boldsymbol{\Omega}_{\bullet h}$ to the X-scores $\boldsymbol{\Xi}_{\bullet h}$ explicitly:

$$\boldsymbol{\Omega}_{\bullet h} \;\; = \;\; \boldsymbol{\Xi}_{\bullet h}\boldsymbol{P}_h + \boldsymbol{R}_{\bullet h}, \tag{2}$$

where $\boldsymbol{\Omega}_{\bullet h} = (\boldsymbol{\omega}_j)_{1\leq j\leq h}$, $\boldsymbol{P}_h = \mathrm{diag}(p_j)_{1\leq j\leq h}$ is a diagonal matrix, and where $\boldsymbol{R}_{\bullet h}$ is a matrix of residuals. We have

$$\boldsymbol{Y} \;\; = \;\; \boldsymbol{\Omega}_H\boldsymbol{D}_H^{\mathsf{T}} + \boldsymbol{F}_H^Y = \boldsymbol{X}\widehat{\boldsymbol{B}}_{PLS} + \boldsymbol{E}_H^Y,$$

where $\boldsymbol{D}_H = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_H]$, $\widehat{\boldsymbol{B}}_{PLS} := \tilde{\boldsymbol{W}}_H\boldsymbol{P}_H\boldsymbol{D}_H^{\mathsf{T}}$, and where the matrices of residuals are $\boldsymbol{F}_H^Y = \boldsymbol{\Xi}_H\boldsymbol{G}_H^{\mathsf{T}} - \boldsymbol{\Omega}_H\boldsymbol{D}_H^{\mathsf{T}} + \boldsymbol{Y}_H$ and $\boldsymbol{E}_H^Y = \boldsymbol{R}_H\boldsymbol{D}_H^{\mathsf{T}} + \boldsymbol{F}_H^Y$.

The second commonly used PLSR algorithm, called SIMPLS (de Jong 1993), calculates the PLS latent components directly as linear combinations of the original variables. The objective function to optimise is (Chun & Keleş 2010, eq. (3))

$$(\boldsymbol{w}_h, \boldsymbol{z}_h) = \underset{\|\boldsymbol{w}\|=\|\boldsymbol{z}\|=1}{\operatorname{argmax}} \; Cov(\boldsymbol{X}\boldsymbol{w}, \boldsymbol{Y}\boldsymbol{z}), \quad \text{subject to} \quad Cov(\boldsymbol{X}\boldsymbol{w}, \boldsymbol{X}\boldsymbol{w}_j) = 0,$$

for $1 \leq j < h$. It is important to note that both algorithms maximise the same covariance function but have different constraints and thus yield different sets of direction vectors.

The decomposition model for SIMPLS is identical to the decomposition of PLS2, the only difference being in how the weights are calculated. In both models we have $\boldsymbol{\Xi}_{\bullet h} = \boldsymbol{X}\boldsymbol{W}_{\bullet h}$ (or $\boldsymbol{\Xi}_{\bullet h} = \boldsymbol{X}\tilde{\boldsymbol{W}}_{\bullet h}$), but the different constraints on the adjusted weights $\boldsymbol{w}_h$ (or $\tilde{\boldsymbol{w}}_h$) give different score vectors; namely $\|\boldsymbol{w}_h\|_2 = 1$ or $\|\mathcal{P}_{\tilde{\boldsymbol{W}}_{\bullet h-1}^{\perp}} \tilde{\boldsymbol{w}}_h\|_2 = 1$.

2.3. Computational details

As described in Section 2.2, the PLS algorithms iteratively construct linear combinations of the original data. These quantities are calculated recursively using *deflated* matrices $\boldsymbol{X}_{h-1}$ and $\boldsymbol{Y}_{h-1}$. At the h-th iteration ($h = 1, \ldots, H$), we compute $\boldsymbol{\xi}_h = \boldsymbol{X}_{h-1}\boldsymbol{u}_h = \boldsymbol{X}\boldsymbol{w}_h$ and $\boldsymbol{\omega}_h = \boldsymbol{Y}_{h-1}\boldsymbol{v}_h = \boldsymbol{Y}\boldsymbol{z}_h$. The (normed) weights $\boldsymbol{u}_h$ and $\boldsymbol{v}_h$ are called the weight vectors (also direction vectors, saliences, or effective loading weight vectors), while $\boldsymbol{w}_h$ and $\boldsymbol{z}_h$ are called the adjusted weights. Since the adjusted weights define the score vectors in terms of the original data matrices (as opposed to the deflated matrices), the size of the elements of the weight vector can be interpreted as the effect of the corresponding variables in the component score. On the other hand, the weight vectors $\boldsymbol{u}_h$ and $\boldsymbol{v}_h$ are defined in terms of the deflated matrices and cannot be interpreted this way. However, these deflated weights offer easier computational implementation.

Each of the four PLS methods can be described as an optimisation problem for the weights $(\boldsymbol{u}_h, \boldsymbol{v}_h)$ coupled with a deflation method to ensure the required orthogonality. Generally, this optimisation problem can be stated as:

$$\underset{\|\boldsymbol{u}\|_2=\|\boldsymbol{v}\|_2=1}{\text{maximise}} \quad \boldsymbol{u}^{\mathsf{T}}\boldsymbol{M}_{h-1}\boldsymbol{v} \tag{3}$$

where $\boldsymbol{M}_{h-1} = \boldsymbol{X}_{h-1}^{\mathsf{T}}\boldsymbol{Y}_{h-1}$ and $\boldsymbol{u}^{\mathsf{T}}\boldsymbol{M}_{h-1}\boldsymbol{v} = Cov(\boldsymbol{X}_{h-1}\boldsymbol{u}, \boldsymbol{Y}_{h-1}\boldsymbol{v})$. One can show (Lafaye de Micheaux et al. 2017, Appendix A) that computation of the h-th pair of PLS weight vectors, from equation (3), is equivalent to solving

$$(\boldsymbol{u}_h, \boldsymbol{v}_h) \quad = \underset{\|\boldsymbol{u}\|_2=\|\boldsymbol{v}\|_2=1,\ \delta>0}{\operatorname{argmin}} \|\boldsymbol{M}_{h-1} - \delta\boldsymbol{u}\boldsymbol{v}^{\mathsf{T}}\|_F^2.$$

The solution to this problem may be calculated by taking the first left and right singular vectors from the SVD of $\boldsymbol{M}_{h-1}$. The remaining deflation methods and initialisations for the PLS methods are given in Table 1.

TABLE 1
Computational details for the four PLS methods. The optimisation (3) is solved for $\boldsymbol{M}_{h-1} = \boldsymbol{X}_{h-1}^{\mathsf{T}}\boldsymbol{Y}_{h-1}$ *with the corresponding initialisation and deflation steps.*

Method	initialise	Deflation	Deflation M
PLS-SVD	$\boldsymbol{X}_0 = \boldsymbol{X}$ $\boldsymbol{Y}_0 = \boldsymbol{Y}$	$\boldsymbol{X}_h = \boldsymbol{X}_{h-1}(\boldsymbol{I} - \boldsymbol{u}_h\boldsymbol{u}_h^{\mathsf{T}})$ $\boldsymbol{Y}_h = \boldsymbol{Y}_{h-1}(\boldsymbol{I} - \boldsymbol{v}_h\boldsymbol{v}_h^{\mathsf{T}})$	$\boldsymbol{M}_h = \boldsymbol{M}_{h-1} - \delta_h\boldsymbol{u}_h\boldsymbol{v}_h^{\mathsf{T}}$ $\delta_h = \boldsymbol{u}_h\boldsymbol{M}_{h-1}\boldsymbol{v}_h^{\mathsf{T}}$
PLS-W2A	$\boldsymbol{X}_0 = \boldsymbol{X}$ $\boldsymbol{Y}_0 = \boldsymbol{Y}$	$\boldsymbol{X}_h = [\boldsymbol{I} - \boldsymbol{\xi}_h(\boldsymbol{\xi}_h^{\mathsf{T}}\boldsymbol{\xi}_h)^{-1}\boldsymbol{\xi}_h^{\mathsf{T}}]\boldsymbol{X}_{h-1}$ $\boldsymbol{Y}_h = [\boldsymbol{I} - \boldsymbol{\omega}_h(\boldsymbol{\omega}_h^{\mathsf{T}}\boldsymbol{\omega}_h)^{-1}\boldsymbol{\omega}_h^{\mathsf{T}}]\boldsymbol{Y}_{h-1}$	$\boldsymbol{M}_h = (\boldsymbol{c}_h\boldsymbol{u}_h^{\mathsf{T}} - \boldsymbol{I}_p)$ $\times \boldsymbol{M}_{h-1}(\boldsymbol{v}_h d_h^{\mathsf{T}} - \boldsymbol{I}_q)$
CCA	$\boldsymbol{X}_0 = \boldsymbol{X}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{+1/2}$ $\boldsymbol{Y}_0 = \boldsymbol{Y}(\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y})^{+1/2}$	$\boldsymbol{X}_h = \boldsymbol{X}_{h-1}(\boldsymbol{I} - \boldsymbol{u}_h\boldsymbol{u}_h^{\mathsf{T}})$ $\boldsymbol{Y}_h = \boldsymbol{Y}_{h-1}(\boldsymbol{I} - \boldsymbol{v}_h\boldsymbol{v}_h^{\mathsf{T}})$	$\boldsymbol{M}_h = \boldsymbol{M}_{h-1} - \delta_h\boldsymbol{u}_h\boldsymbol{v}_h^{\mathsf{T}}$ $\delta_h = \boldsymbol{u}_h\boldsymbol{M}_{h-1}\boldsymbol{v}_h^{\mathsf{T}}$
PLSR	$\boldsymbol{X}_0 = \boldsymbol{X}$ $\boldsymbol{Y}_0 = \boldsymbol{Y}$ $\boldsymbol{N}_0 = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$	$\boldsymbol{X}_h = [\boldsymbol{I} - \boldsymbol{\xi}_h(\boldsymbol{\xi}_h^{\mathsf{T}}\boldsymbol{\xi}_h)^{-1}\boldsymbol{\xi}_h^{\mathsf{T}}]\boldsymbol{X}_{h-1}$ $\boldsymbol{Y}_h = [\boldsymbol{I} - \boldsymbol{\xi}_h(\boldsymbol{\xi}_h^{\mathsf{T}}\boldsymbol{\xi}_h)^{-1}\boldsymbol{\xi}_h^{\mathsf{T}}]\boldsymbol{Y}_{h-1}$	$\boldsymbol{M}_h = (\boldsymbol{I} - \boldsymbol{c}_h\boldsymbol{u}_h^{\mathsf{T}})$ $\times (\boldsymbol{M}_{h-1} - \boldsymbol{N}_{h-1}\boldsymbol{u}_h\boldsymbol{d}_h^{\mathsf{T}})$ $\boldsymbol{N}_h = (\boldsymbol{c}_h\boldsymbol{u}_h^{\mathsf{T}} - \boldsymbol{I})$ $\times \boldsymbol{N}_{h-1}(\boldsymbol{u}_h\boldsymbol{c}_h^{\mathsf{T}} - \boldsymbol{I})$

3. Penalised PLS

All of the previous PLS methods can be written in terms of a single optimisation problem coupled with an appropriate deflation to ensure the appropriate orthogonal constraints. In this section, we introduce the framework for penalised partial least squares in the unified PLS methodology. Several penalisations are then considered and presented in a unified algorithm that can perform all four PLS methods, and their regularised versions.

3.1. Finding the PLS weights

We focus on a particular class of algorithm designed to induce sparsity for the PLS weights. For this class of algorithm the weights, $(\boldsymbol{u}_h, \boldsymbol{v}_h)$ at step h, are defined as the solution to the optimisation problem:

$$\text{maximise } \boldsymbol{u}^{\mathsf{T}}\boldsymbol{M}_{h-1}\boldsymbol{v} - P_{\lambda_1}(\boldsymbol{u}) - P_{\lambda_2}(\boldsymbol{v}) \quad \text{subject to } \|\boldsymbol{u}\|_2 \leq 1,\ \|\boldsymbol{v}\|_2 \leq 1, \quad (4)$$

where P_{λ_1} and P_{λ_2} are convex penalty functions with tuning parameters λ_1 and λ_2, and the matrix $\boldsymbol{M}_{h-1}$ is defined using the appropriate deflation (see Table 1). The resulting objective function can be recognised as the Lagrangian of the penalised matrix decomposition introduced by Witten et al. (2009). If the penalty functions are homogenous of order one: $P(cx) = cP(x)$ for all $c > 0$, then the weights $(\boldsymbol{u}_h, \boldsymbol{v}_h)$ can be found by iteratively calculating

$$\tilde{\boldsymbol{u}}_h = \underset{\tilde{\boldsymbol{u}} \in \mathbb{R}^p}{\text{argmin}} \left\{ \|\boldsymbol{M}_{h-1} - \tilde{\boldsymbol{u}}\boldsymbol{v}^{\mathsf{T}}\|_F^2 + P_{\lambda_1}(\tilde{\boldsymbol{u}}) \right\}, \quad (5)$$

$$\tilde{\boldsymbol{v}}_h = \underset{\tilde{\boldsymbol{v}} \in \mathbb{R}^q}{\text{argmin}} \left\{ \|\boldsymbol{M}_{h-1}^{\mathsf{T}} - \tilde{\boldsymbol{v}}\boldsymbol{u}^{\mathsf{T}}\|_F^2 + P_{\lambda_2}(\tilde{\boldsymbol{v}}) \right\}, \quad (6)$$

and scaling; $\boldsymbol{u}_h = \tilde{\boldsymbol{u}}_h / \|\tilde{\boldsymbol{u}}_h\|_2$ if $\|\tilde{\boldsymbol{u}}_h\|_2 > 0$ and $\boldsymbol{u}_h = \boldsymbol{0}$ otherwise, $\boldsymbol{v}_h = \tilde{\boldsymbol{v}}_h / \|\tilde{\boldsymbol{v}}_h\|_2$ if $\|\tilde{\boldsymbol{v}}_h\|_2 > 0$ and $\boldsymbol{v}_h = \boldsymbol{0}$ otherwise. This algorithm for finding penalised weights was studied in Allen et al. (2014) for regularised principal component analysis. We note that similar variants of this process have been proposed elsewhere,

but these methods are presented as stand-alone methods such as sparse PCA (Shen & Huang 2008), sparse CCA (Witten et al. 2009), sparse PLS regression or sparse canonical PLS (Lê Cao et al. 2008). Our work provides a clear, unified framework for multiple PLS methods that allows for calculation of the weights with sparsity-inducing penalisations.

Note that the optimisation problem (4) is a biconvex problem and thus multiple optimal solutions may exist (Tseng 1988). By iteratively solving the optimisation problems in (5) and (6) we are guaranteed to improve or retain the same objective value in problem (4). However, if the initial values for $\boldsymbol{u}$ and $\boldsymbol{v}$ are poorly chosen, it is possible for the method to get caught in a local optimum or to end up oscillating between solutions (Netrapalli et al. 2015). We use the SVD solution to initialise our algorithm, an approach that is common for implementing these types of algorithms (Allen & Tibshirani 2010, Witten et al. 2009).

3.2. Deflation and the PLS weights

Computing the penalised versions of the four different PLS methods is achieved by alternating between two subtasks: solving (5) and (6) for the weights, and matrix deflation. Without the penalties, P_{λ_1} and P_{λ_2}, the matrix deflation enforces certain orthogonality constraints for each of the four standard PLS methods. However, with either penalty P_{λ_1} or P_{λ_2}, these deflations do not ensure any orthogonal constraints. Although these constraints are lost, Witten et al. (2009) state that it is not clear that orthogonality is desirable as it may be at odds with sparsity. That is, enforcing the additional orthogonality constraints may result in less sparse solutions. Similar to Witten et al. (2009) and Lê Cao et al. (2008) we use the standard deflation methods in our implementation of the penalised PLS methods. Alternative matrix deflations have been proposed for sparse PCA (Mackey 2009). However, these methods have not been extended in the general penalised PLS framework.

Another key observation is that for the NIPALS PLS regression, PLS-W2A and CCA the scores were defined in terms of the deflated matrices $\boldsymbol{\xi}_h = \boldsymbol{X}_{h-1}\boldsymbol{u}_h$ and $\boldsymbol{\omega}_h = \boldsymbol{Y}_{h-1}\boldsymbol{v}_h$. Consequently, the sparse estimators given by solving (5) and (6) perform variable selection of the deflated matrices. Thus the latent components formed using these methods have the interpretation given by Remark 1 below. In our implementation, we also calculate the adjusted weights $\boldsymbol{w}_h$ and $\boldsymbol{z}_h$ (or $\tilde{\boldsymbol{w}}_h$ and $\tilde{\boldsymbol{z}}_h$), where $\boldsymbol{\xi}_h = \boldsymbol{X}\boldsymbol{w}_h$ and $\boldsymbol{\omega}_h = \boldsymbol{Y}\boldsymbol{z}_h$. These weights allow for direct interpretation of the selected variables in the PLS model. Note that although $\boldsymbol{w}_h$ and $\boldsymbol{z}_h$ allow for direct interpretation of the selected variables, the sparsity is enforced on $\boldsymbol{u}_h$ and $\boldsymbol{v}_h$. So if $\boldsymbol{u}_h$ and $\boldsymbol{v}_h$ are sparse, this does not necessarily mean that the adjusted weights $\boldsymbol{w}_h$ and $\boldsymbol{z}_h$ will be sparse.

Remark 1 *The first latent variable $\boldsymbol{\xi}_1 = \boldsymbol{X}\boldsymbol{u}_1$ is built as a sparse linear combination (with weights in $\boldsymbol{u}_1$) of the original variables. The next latent variable $\boldsymbol{\xi}_2 = \mathcal{P}_{\boldsymbol{\xi}_1^{\perp}}\boldsymbol{X}\boldsymbol{u}_2$ is the part of the sparse linear combination (with weights in $\boldsymbol{u}_2$) of the original variables that has not been already explained by the first latent*

variable. And more generally, the h-th latent variable is built as a sparse linear combination of the original variables, from which we extract (by projection) the information not already brought by the previous latent variables.

We note that an alternative SIMPLS formulation for the penalised PLS methods was proposed in a regression setting by Allen et al. (2013). In the SIMPLS method the weights are directly interpreted in terms of the original variables, so $\boldsymbol{w}_h = \boldsymbol{u}_h$ and $\boldsymbol{z}_h = \boldsymbol{v}_h$. Although this method allows for direct penalisation of the weights, the orthogonality conditions still do not hold. We have incorporated this method and a similar variant for PLS-W2A into our R package `bigsgPLS` to allow for direct penalisation of the weights.

3.3. The penalised PLS methods

Computationally, the PLS method is an efficient approach to sparse latent variable modeling. The main computational cost is in solving for the PLS weights as described in equations (5) and (6). The cost of solving for these weights is penalty-specific but can be minimal in a number of useful applications. We detail a few examples where these equations have been solved analytically and provide an algorithm that treats the penalised versions of the four PLS cases (i)–(iv).

3.3.1. Sparse PLS

The (original) sparse PLS version sPLS (Lê Cao et al. 2008, Chun & Keleş 2010) considers the following penalty functions

$$P_{\lambda_1}(\tilde{\boldsymbol{u}}) = \sum_{i=1}^{p} 2\lambda_1 |\tilde{u}_i| \quad \text{and} \quad P_{\lambda_2}(\tilde{\boldsymbol{v}}) = \sum_{j=1}^{q} 2\lambda_2 |\tilde{v}_j|. \tag{7}$$

These penalties induce the desired sparsity of the weight vectors $\boldsymbol{u}_h = \tilde{\boldsymbol{u}}_h / \|\tilde{\boldsymbol{u}}_h\|_2$ and $\boldsymbol{v}_h = \tilde{\boldsymbol{v}}_h / \|\tilde{\boldsymbol{v}}_h\|_2$, thanks to the well known properties of the ℓ_1-norm or lasso penalty (Tibshirani 1994). The closed form solution for this problem is:

$$\tilde{\boldsymbol{u}} = g^{\text{soft}}(\boldsymbol{M}\boldsymbol{v}, \lambda_1), \qquad \tilde{\boldsymbol{v}} = g^{\text{soft}}(\boldsymbol{M}^{\mathsf{T}}\boldsymbol{u}, \lambda_2). \tag{8}$$

where $g^{\text{soft}}(\cdot, \lambda_1)$ is the soft thresholding function, with the understanding that the function is applied componentwise. To unify these results with the ones to come, we introduce the sparsifier functions S_u and S_v to denote analytical functions that provide the solution for the weights. The sparsifiers are functions of the data $\boldsymbol{M}$, the fixed weight $\boldsymbol{u}$ (or $\boldsymbol{v}$) and additional penalty specific parameters θ_u (or θ_v). For sparse PLS we have,

$$\tilde{\boldsymbol{u}}_h = S_u(\boldsymbol{v}; \boldsymbol{M}, \theta_u) = g^{\text{soft}}(\boldsymbol{M}\boldsymbol{v}, \lambda_1), \quad \tilde{\boldsymbol{v}}_h = S_v(\boldsymbol{u}; \boldsymbol{M}, \theta_v) = g^{\text{soft}}(\boldsymbol{M}^{\mathsf{T}}\boldsymbol{u}, \lambda_2), \tag{9}$$

where $\theta_u = \lambda_1$ and $\theta_v = \lambda_2$.

3.3.2. Group PLS

There are many statistical problems in which the data has a natural grouping structure. In these problems, it is preferable to estimate all coefficients within a group to be zero or nonzero simultaneously. A leading example is in gene expression data, where genes within the same gene pathway have a similar biological function. Selecting a group amounts to selecting a pathway (Palermo et al. 2011). Variables can be grouped for other reasons, for example, when we have categorical covariates in our data. The categorical data is coded by its factor levels using dummy variables, and selection or exclusion of this group of dummy variables is equivelant to selection of the categorical covariate (Nguyen & Rocke 2002).

Let us consider a situation where both matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be divided respectively into K and L sub-matrices (i.e., groups) $\boldsymbol{X}^{(k)} : n \times p_k$ and $\boldsymbol{Y}^{(l)} : n \times q_l$, where p_k (resp. q_l) is the number of covariates in group k (resp. l). The aim is to select only a few groups of $\boldsymbol{X}$ which are related to a few groups of $\boldsymbol{Y}$. We define $\boldsymbol{M}^{(k,\cdot)} = \boldsymbol{X}^{(k)^\intercal}\boldsymbol{Y}$ and $\boldsymbol{M}^{(\cdot,l)} = \boldsymbol{Y}^{(l)^\intercal}\boldsymbol{X}$.

Group selection is accomplished using the group lasso penalties (Yuan & Lin 2006) in the optimisation problems (5) and (6):

$$P_{\lambda_1}(\tilde{\boldsymbol{u}}) = \lambda_1 \sum_{k=1}^{K} \sqrt{p_k} \|\tilde{\boldsymbol{u}}^{(k)}\|_2; \qquad P_{\lambda_2}(\tilde{\boldsymbol{v}}) = \lambda_2 \sum_{l=1}^{L} \sqrt{q_l} \|\tilde{\boldsymbol{v}}^{(l)}\|_2, \tag{10}$$

where $\tilde{\boldsymbol{u}}^{(k)}$ and $\tilde{\boldsymbol{v}}^{(l)}$ are the sub vectors of the (unscaled) weights $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ corresponding to the variables in group k of $\boldsymbol{X}$ and group l of $\boldsymbol{Y}$ respectively. This penalty is a group generalisation of the lasso penalty. Depending on the tuning parameter $\lambda_1 \geq 0$ (or $\lambda_2 \geq 0$), the entire weight subvector $\tilde{\boldsymbol{u}}^{(k)}$ (or $\tilde{\boldsymbol{v}}^{(l)}$) will be zero, or nonzero together. The closed form solution for the group PLS method for the k-th subvector of $\tilde{\boldsymbol{u}}$, and the l-th subvector of $\tilde{\boldsymbol{v}}$ is given by $\tilde{\boldsymbol{u}}^{(k)} = S_u^{(k)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u)$ and $\tilde{\boldsymbol{v}}^{(l)} = S_v^{(l)}(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v)$ respectively, where

$$S_u^{(k)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u) = \left(1 - \frac{\lambda_1}{2} \frac{\sqrt{p_k}}{\|\boldsymbol{M}^{(k,\cdot)}\tilde{\boldsymbol{v}}\|_2}\right)_+ \boldsymbol{M}^{(k,\cdot)}\tilde{\boldsymbol{v}}, \tag{11}$$

$$S_v^{(l)}(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v) = \left(1 - \frac{\lambda_2}{2} \frac{\sqrt{q_l}}{\|\boldsymbol{M}^{(\cdot,l)}\tilde{\boldsymbol{u}}\|_2}\right)_+ \boldsymbol{M}^{(\cdot,l)}\tilde{\boldsymbol{u}}. \tag{12}$$

The sparsifier functions are applied groupwise

$$\tilde{\boldsymbol{u}} = S_u(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u) = \left(S_u^{(1)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u), \ldots, S_u^{(K)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u)\right)$$
$$\tilde{\boldsymbol{v}} = S_v(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v) = \left(S_v^{(1)}(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v), \ldots, S_v^{(L)}(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v)\right),$$

with $\boldsymbol{\theta}_u = (p_1, \ldots, p_K, \lambda_1)$ and $\boldsymbol{\theta}_v = (q_1, \ldots, q_L, \lambda_2)$. A proof of these equations is given in (Liquet et al. 2016).

3.3.3. Sparse group PLS

One potential drawback of gPLS is that it includes a group in the model only when all individual weights in that group are non-zero. However, sometimes we would like both sparsity of groups and sparsity within each group. For example, if the predictor matrix contains genes, we might be interested in identifying particularly important genes in pathways of interest. The sparse group lasso (Simon et al. 2013) achieves this within group sparsity. The sparse group selection in the PLS methodology is accomplished using the sparse group lasso penalty in the optimisation problem (5) and (6):

$$\begin{aligned} P_{\lambda_1}(\tilde{\boldsymbol{u}}) &= (1-\alpha_1)\lambda_1 \sum_{k=1}^{K} \sqrt{p_k} \|\tilde{\boldsymbol{u}}^{(k)}\|_2 + \alpha_1 \lambda_1 \|\tilde{\boldsymbol{u}}\|_1, \\ P_{\lambda_2}(\tilde{\boldsymbol{v}}) &= (1-\alpha_2)\lambda_2 \sum_{l=1}^{L} \sqrt{q_l} \|\tilde{\boldsymbol{v}}^{(l)}\|_2 + \alpha_2 \lambda_2 \|\tilde{\boldsymbol{v}}\|_1. \end{aligned}$$

The sparse group penalty introduces tuning parameters α_1 and α_2 which provide a link between the group lasso penalty ($\alpha_1 = 0$, $\alpha_2 = 0$) and the lasso ($\alpha_1 = 1$, $\alpha_2 = 1$). Depending on the combination of α_1 and λ_1 (or α_2 and λ_2) the (unscaled) weight subvector $\tilde{\boldsymbol{u}}^{(k)}$ (or $\tilde{\boldsymbol{v}}^{(k)}$) will be eliminated entirely, or sparsely estimated. The adaptation of the sparse group penalty for the PLS method was first considered in (Liquet et al. 2016). The closed form solution of the sparse group PLS method for the k-th subvector of $\tilde{\boldsymbol{u}}^{(k)}$ is given by

$$S_u^{(k)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u) = \begin{cases} \mathbf{0} & \text{if } \frac{\|g_1\|_2}{(1-\alpha_1)\sqrt{p_k}} \leq \lambda_1 \\ \frac{g_1}{2} - \frac{\lambda_1(1-\alpha_1)\sqrt{p_k} g_1}{2\|g_1\|} & \text{otherwise} \end{cases}$$

where $g_1 = g^{\text{soft}}\left(\boldsymbol{M}^{(k,\cdot)}\tilde{\boldsymbol{v}}, \lambda_1\alpha_1/2\right)$. Similarly, the l-th subvector of $\tilde{\boldsymbol{v}}$ is given by

$$S_v^{(l)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_v) = \begin{cases} \mathbf{0} & \text{if } \frac{\|g_2\|_2}{(1-\alpha_2)\sqrt{q_l}} \leq \lambda_2 \\ \frac{g_2}{2} - \frac{\lambda_2(1-\alpha_2)\sqrt{q_l} g_2}{2\|g_2\|} & \text{otherwise} \end{cases}$$

where $g_2 = g^{\text{soft}}\left(\boldsymbol{M}^{(\cdot,l)}\tilde{\boldsymbol{u}}, \lambda_2\alpha_2/2\right)$. The sparsifier functions for these penalties are:

$$\begin{aligned} \tilde{\boldsymbol{u}} = S_u(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u) &= \left(S_u^{(1)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u), \ldots, S_u^{(K)}(\tilde{\boldsymbol{v}}; \boldsymbol{M}, \boldsymbol{\theta}_u)\right) \\ \tilde{\boldsymbol{v}} = S_v(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v) &= \left(S_v^{(1)}(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v), \ldots, S_v^{(L)}(\tilde{\boldsymbol{u}}; \boldsymbol{M}, \boldsymbol{\theta}_v)\right) \end{aligned}$$

with $\boldsymbol{\theta}_u = (p_1, \ldots, p_K, \lambda_1, \alpha_1)$ and $\boldsymbol{\theta}_v = (q_1, \ldots, q_L, \lambda_2, \alpha_2)$.

3.3.4. Other penalties

The penalties discussed so far have enforced general sparsity or sparsity for a known grouping structure in the data. Extensions to the group structured

sparsity in partial least squares setting have also been considered in terms of overlapping groups (Chen & Liu 2012), or additional grouping restrictions (Sutton et al. 2018). The penalisations considered so far have all resulted in closed form solutions for the updates of $\boldsymbol{u}$ and $\boldsymbol{v}$. We note here that this is not always the case. The fused lasso penalty (Tibshirani et al. 2005) is defined by:

$$\begin{aligned} P_{\lambda_1}(\tilde{\boldsymbol{u}}) &= \lambda_1\alpha_1 \sum_{i=2}^{p} |\tilde{u}_i - \tilde{u}_{i-1}| + (1-\alpha_1)\lambda_1 \|\tilde{\boldsymbol{u}}\|_1, \\ P_{\lambda_2}(\tilde{\boldsymbol{v}}) &= \lambda_1\alpha_1 \sum_{i=2}^{q} |\tilde{v}_i - \tilde{v}_{i-1}| + (1-\alpha_1)\lambda_1 \|\tilde{\boldsymbol{v}}\|_1. \end{aligned}$$

The first term in this penalty causes neighboring coefficients to shrink together and will cause some to be identical, and the second causes regular lasso shrinkage of the parameters for variable selection. Unlike the previous methods, a closed form solution for the fused lasso cannot be directly achieved. This is because the penalty is not a separable function of the coordinates. Because there is no closed form solution for the fused lasso, we cannot write a sparsifier function, so we have not considered this method. We note that methods exist that can solve the fused lasso problem, either by reparameterisation, dynamic programming or path based algorithms. In particular, Witten et al. (2009) have considered solving problems of the form (5) and (6) with the fused lasso penalty. In their paper, they propose a sparse and fused penalised CCA, however in their derivation they assume $\boldsymbol{X}^\mathsf{T}\boldsymbol{X} = \boldsymbol{I}$ and $\boldsymbol{Y}^\mathsf{T}\boldsymbol{Y} = \boldsymbol{I}$. In our framework, this method would be sparse and fused penalised PLS-SVD.

4. The unified algorithm

Algorithm 1 allows for a unified computation of all four PLS versions (i)–(iv), with a possibility to add sparsity. Adjusted weights can also be computed and, if the number of requested components H is greater than 1, a deflation step is executed. Note that, if $\boldsymbol{Y}$ is taken equal to $\boldsymbol{X}$, this algorithm performs PCA, as well as sparse PCA versions. If this is the case, the optimised criteria are simply restated in terms of variance instead of covariance. We now have all the ingredients to propose a unifying algorithm.

Remark 2 *On line 10, we impose that* $\|\boldsymbol{u}_1\|_2 = \|\boldsymbol{v}_1\|_2 = 1$ *and* $u_{1,i} > 0$ *where* $i = argmax_{1\le j\le p}|u_{1,j}|$ *to ensure uniqueness of the results.*

Note that $\boldsymbol{w}_h$ *and* $\boldsymbol{z}_h$ *of lines 22, 24 and 28 correspond to* $\tilde{\boldsymbol{w}}_h$ *and* $\tilde{\boldsymbol{z}}_h$ *in the text.*

At this point, it is worthwhile noting that when p and q are small compared to n, one can slightly modify Algorithm 1 by using the recursive equations that express $\boldsymbol{M}_h$ in terms of $\boldsymbol{M}_{h-1}$, instead of using the recursions on $\boldsymbol{X}_h$ and $\boldsymbol{Y}_h$. These recursions are provided in Table 1 of Section 2.3. This should increase speed of execution of the algorithm.

Algorithm 1 Sparse and non-sparse PLS algorithm for the four cases (i)–(iv)

Require: $\boldsymbol{X}_0 = \boldsymbol{X}$ (with n rows), $\boldsymbol{Y}_0 = \boldsymbol{Y}$, H, Case, ϵ, $\boldsymbol{\theta}_x$, $\boldsymbol{\theta}_y$, S_u, S_v

Extract λ_x and λ_y as the first element of $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$ respectively

$\boldsymbol{M}_0 \leftarrow \boldsymbol{X}_0^\intercal \boldsymbol{Y}_0/(n-1)$; $\boldsymbol{P} \leftarrow \boldsymbol{I}_p$ and $\boldsymbol{Q} \leftarrow \boldsymbol{I}_q$ ▷ Initialisation

$\boldsymbol{u}_0 \leftarrow \boldsymbol{0}_p$; $\boldsymbol{c}_0^\intercal \leftarrow \boldsymbol{0}_p^\intercal$; $\boldsymbol{v}_0 \leftarrow \boldsymbol{0}_q$; $\boldsymbol{e}_0^\intercal \leftarrow \boldsymbol{0}_q^\intercal$ and $\boldsymbol{\xi}_0 \leftarrow \boldsymbol{\omega}_0 \leftarrow \boldsymbol{1}_n$

If Case (iii) **then**

  $\boldsymbol{A} \leftarrow \sqrt{n-1}(\boldsymbol{X}_0^\intercal \boldsymbol{X}_0 + \lambda_x \boldsymbol{P})^{-1/2}$; $\boldsymbol{B} \leftarrow \sqrt{n-1}(\boldsymbol{Y}_0^\intercal \boldsymbol{Y}_0 + \lambda_y \boldsymbol{Q})^{-1/2}$; $\boldsymbol{X}_0 \leftarrow \boldsymbol{X}_0 \boldsymbol{A}$; $\boldsymbol{Y}_0 \leftarrow \boldsymbol{Y}_0 \boldsymbol{B}$ and $\boldsymbol{M}_0 \leftarrow \boldsymbol{A}\boldsymbol{M}_0\boldsymbol{B}$

end if

for $h = 1, \ldots, H$ **do**

  Apply the SVD to $\boldsymbol{M}_{h-1}$ and extract the first triplet $(\delta_1, \boldsymbol{u}_1, \boldsymbol{v}_1)$ of singular value and vectors.

  Set $\boldsymbol{u}_\mathrm{h} \leftarrow \boldsymbol{u}_1$ and $\boldsymbol{v}_\mathrm{h} \leftarrow \boldsymbol{v}_1$

  while $\boldsymbol{u}_h$ has not converged$^{(*)}$ **do** ▷ Sparsity step if $\lambda_x > 0$ or $\lambda_y > 0$

    $\tilde{\boldsymbol{u}}_h \leftarrow S_u(\boldsymbol{v}_h; \boldsymbol{M}_{h-1}, \boldsymbol{\theta}_x)$; $\boldsymbol{u}_h \leftarrow \tilde{\boldsymbol{u}}_h / \|\tilde{\boldsymbol{u}}_h\|_2$

    $\tilde{\boldsymbol{v}}_h \leftarrow S_v(\boldsymbol{u}_h; \boldsymbol{M}_{h-1}, \boldsymbol{\theta}_y)$; $\boldsymbol{v}_h \leftarrow \tilde{\boldsymbol{v}}_h / \|\tilde{\boldsymbol{v}}_h\|_2$

  end while ▷ End of sparsity step

  $\boldsymbol{\xi}_h \leftarrow \boldsymbol{X}_{h-1}\boldsymbol{u}_h$ ▷ X-score

  $\boldsymbol{\omega}_h \leftarrow \boldsymbol{Y}_{h-1}\boldsymbol{v}_h$ ▷ Y-score

  If Case (i) **then** ▷ Adjusted weights step

    $\boldsymbol{w}_h \leftarrow \boldsymbol{u}_h$ and $\boldsymbol{z}_h \leftarrow \boldsymbol{v}_h$

  end if

  If Case (ii) **then**

    $\boldsymbol{P} \leftarrow \boldsymbol{P}(\boldsymbol{I}_p - \boldsymbol{u}_{h-1}\boldsymbol{c}_{h-1}^\intercal)$

    $\boldsymbol{Q} \leftarrow \boldsymbol{Q}(\boldsymbol{I}_q - \boldsymbol{v}_{h-1}\boldsymbol{e}_{h-1}^\intercal)$

    $\boldsymbol{w}_h \leftarrow \boldsymbol{P}\boldsymbol{u}_h$ and $\boldsymbol{z}_h \leftarrow \boldsymbol{Q}\boldsymbol{v}_h$

  end if

  If Case (iii) **then** $\boldsymbol{w}_h \leftarrow \boldsymbol{A}\boldsymbol{u}_h$ and $\boldsymbol{z}_h \leftarrow \boldsymbol{B}\boldsymbol{v}_h$

  If Case (iv) **then**

    $\boldsymbol{P} \leftarrow \boldsymbol{P}(\boldsymbol{I}_p - \boldsymbol{u}_{h-1}\boldsymbol{c}_{h-1}^\intercal)$

    $\boldsymbol{w}_h \leftarrow \boldsymbol{P}\boldsymbol{u}_h$

    $\boldsymbol{z}_h \leftarrow \boldsymbol{v}_h$

  end if ▷ End of adjusted weights step

  If Case (i) or (iii) **then** ▷ Deflation step

    $\boldsymbol{c}_h^\intercal \leftarrow \boldsymbol{u}_h^\intercal$ and $\boldsymbol{e}_h^\intercal \leftarrow \boldsymbol{v}_h^\intercal$

  end if

  If Case (ii) or (iv) **then** $\boldsymbol{c}_h^\intercal \leftarrow \boldsymbol{\xi}_h^\intercal \boldsymbol{X}_{h-1} / \|\boldsymbol{\xi}_h\|_2^2$

  If Case (ii) **then** $\boldsymbol{e}_h^\intercal \leftarrow \boldsymbol{\omega}_h^\intercal \boldsymbol{Y}_{h-1} / \|\boldsymbol{\omega}_h\|_2^2$

  If Case (iv) **then** $\boldsymbol{d}_h^\intercal \leftarrow \boldsymbol{\xi}_h^\intercal \boldsymbol{Y}_{h-1} / \|\boldsymbol{\xi}_h\|_2^2$

  $\boldsymbol{X}_h \leftarrow \boldsymbol{X}_{h-1} - \boldsymbol{\xi}_h \boldsymbol{c}_h^\intercal$

  If Case (iv) **then**

    $\boldsymbol{Y}_h \leftarrow \boldsymbol{Y}_{h-1} - \boldsymbol{\xi}_h \boldsymbol{d}_h^\intercal$ ▷ PLSR

  Else

    $\boldsymbol{Y}_h \leftarrow \boldsymbol{Y}_{h-1} - \boldsymbol{\omega}_h \boldsymbol{e}_h^\intercal$

  End If

  $\boldsymbol{M}_h \leftarrow \boldsymbol{X}_h^\intercal \boldsymbol{Y}_h$ ▷ End of deflation step

  Store $\boldsymbol{\xi}_h$, $\boldsymbol{\omega}_h$, $\boldsymbol{u}_h$, $\boldsymbol{v}_h$, $\boldsymbol{w}_h$, $\boldsymbol{z}_h$, $\boldsymbol{c}_h^\intercal$ and ($\boldsymbol{d}_h^\intercal$ or $\boldsymbol{e}_h^\intercal$)

end for

$(*)$ Convergence of a vector $\boldsymbol{t}$ is tested on the change in $\boldsymbol{t}$, i.e., $\|\boldsymbol{t}_\mathrm{old} - \boldsymbol{t}_\mathrm{new}\|_2 / \|\boldsymbol{t}_\mathrm{new}\|_2 < \epsilon$, where ϵ is "small", e.g., 10^{-6}.

Moreover, one can use various approaches to deal with the cases when n, p or q are too large in our algorithm, making some objects too large for the computer's memory. These can be divided into *chunk approaches* and *streaming*

(or incremental) approaches, which are presented in the next subsections. Of course, any combinations of these approaches can be used if necessary. Some of these approaches might also increase the computation speed, even in a context where all objects would fit into memory.

4.1. Matrix multiplication using chunks

To scale Algorithm 1 to big data (i.e., very large $n \gg p$ and q), we can use a simple idea to multiply two very large matrices that are too big to fit into the computer's memory.

Let us divide the total number n of rows of $\boldsymbol{X}$ (resp. of $\boldsymbol{Y}$) into blocks $\boldsymbol{X}_{(g)}$ (resp. $\boldsymbol{Y}_{(g)}$), $g = 1, \ldots, G$, of (approximatively) the same size. We have

$$\boldsymbol{X}^\mathsf{T}\boldsymbol{Y} = \sum_{g=1}^{G} \boldsymbol{X}_{(g)}^\mathsf{T}\boldsymbol{Y}_{(g)}.$$

The number of blocks G has to be chosen so that each product $\boldsymbol{X}_{(g)}^\mathsf{T}\boldsymbol{Y}_{(g)} : p \times q$ can be done within the available RAM. Note that all these products can be performed in parallel if the required computing equipment is available.

4.2. SVD when p or q is very large

The main step of our algorithm is the computation of the first triplet $(\delta_1, \boldsymbol{u}_1, \boldsymbol{v}_1)$ in the SVD of the $(p \times q)$ matrices $\boldsymbol{M}_{h-1}$. The `irlba` (Baglama & Reichel 2015) R package can be used to compute quite easily this triplet for values of p and q as big as $50,000$. This package is based on an augmented implicitly restarted Lanczos bidiagonalisation method.

When p (or q) is much larger, another approach is necessary to compute the SVD of $\boldsymbol{M}_{h-1}$; see e.g., (Liang et al. 2016). Suppose that p is large but not q, which is common in several applications. We thus suppose that $p \gg q$. The Algorithm 1 in Liang et al. (2016) is now presented to highlight the elements needed in our algorithm. We can partition a large matrix $\boldsymbol{M} : p \times q$ by rows into a small number s of submatrices (or chunks):

$$\boldsymbol{M}^\mathsf{T} = \begin{pmatrix} \boldsymbol{M}_1^\mathsf{T} & \boldsymbol{M}_2^\mathsf{T} & \cdots & \boldsymbol{M}_s^\mathsf{T} \end{pmatrix}.$$

Let $\boldsymbol{M}_i = \boldsymbol{U}_i\boldsymbol{D}_i\boldsymbol{V}_i^\mathsf{T}$ denote the SVD of $\boldsymbol{M}_i : g \times q$ such that $gs = p$ (w.l.o.g.). We can take g much larger than q as long as it is still possible to compute the SVD of these submatrices. Define $\tilde{\boldsymbol{U}} : p \times p$ and $\boldsymbol{H} : p \times q$ by

$$\tilde{\boldsymbol{U}} = \begin{pmatrix} \boldsymbol{U}_1 & & & \\ & \boldsymbol{U}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{U}_s \end{pmatrix} \quad \text{and} \quad \boldsymbol{H} = \begin{pmatrix} \boldsymbol{D}_1\boldsymbol{V}_1^\mathsf{T} \\ \boldsymbol{D}_2\boldsymbol{V}_2^\mathsf{T} \\ \vdots \\ \boldsymbol{D}_s\boldsymbol{V}_s^\mathsf{T} \end{pmatrix},$$

where $\boldsymbol{U}_i : g \times g$ and $\boldsymbol{D}_i : g \times q$ and $\boldsymbol{V}_i : q \times q$, so that $\boldsymbol{M} = \tilde{\boldsymbol{U}}\boldsymbol{H}$. Let $\boldsymbol{H} = \boldsymbol{U}_H \boldsymbol{D}_H \boldsymbol{V}_H^\intercal$ be the SVD of $\boldsymbol{H}$. Note that this matrix is as large as $\boldsymbol{M}$ so one may wonder what has been gained with this approach. But $\boldsymbol{D}_i$ being a $g \times q$ diagonal rectangular matrix, $\boldsymbol{D}_i \boldsymbol{V}_i^\intercal$ has $g - q$ zero row-vectors in its bottom. Consequently, the matrix $\boldsymbol{H}$ contains only sq non-zero row vectors. Now let $\tilde{\boldsymbol{H}} = \boldsymbol{R}\boldsymbol{H} : p \times q$ be a rearrangement in rows for $\boldsymbol{H}$ such that its first sq row vectors are non-zero and $p - sq$ row vectors are in its bottom. We now have to compute $\boldsymbol{U}^* \boldsymbol{D}^* \boldsymbol{V}^{*\intercal}$, the SVD of a (much smaller) $sq \times q$ matrix[1]:

$$\boldsymbol{H} \quad = \quad \boldsymbol{R}^\intercal \tilde{\boldsymbol{H}} = \underbrace{\boldsymbol{R}^\intercal \begin{bmatrix} \boldsymbol{U}^* & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{p-sq} \end{bmatrix}}_{\boldsymbol{U}_H} \underbrace{\begin{bmatrix} \boldsymbol{D}^* \\ \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{D}_H} \underbrace{\boldsymbol{V}^{*\intercal}}_{\boldsymbol{V}_H^\intercal}.$$

We obtain $\boldsymbol{M} = (\tilde{\boldsymbol{U}}\boldsymbol{U}_H)\boldsymbol{D}_H \boldsymbol{V}_H^\intercal$ which forms a SVD of $\boldsymbol{M}$.

Now, let $\boldsymbol{1}$ be a vector containing only 0s but a 1 in the first position. For our PLS algorithm, we only need to compute the first triplet in the SVD of $\boldsymbol{M}$, namely $\delta_1 = \boldsymbol{1}_p^\intercal \boldsymbol{D}_H \boldsymbol{1}_q = \boldsymbol{D}^*_{1,1}$, $\boldsymbol{v}_1 = \boldsymbol{V}_{\bullet 1} = \boldsymbol{V}_H \boldsymbol{1}_q = \boldsymbol{V}^* \boldsymbol{1}_q$ and the first column of $(\tilde{\boldsymbol{U}}\boldsymbol{U}_H)$:

$$\boldsymbol{u}_1 = (\tilde{\boldsymbol{U}}\boldsymbol{U}_H)\boldsymbol{1}_p \quad = \quad \tilde{\boldsymbol{U}}\boldsymbol{R}^\intercal \begin{bmatrix} \boldsymbol{U}^* & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{p-sq} \end{bmatrix} \boldsymbol{1}_p = \begin{bmatrix} \boldsymbol{U}_{1,\bullet q}(\boldsymbol{U}^*_{\bullet 1})_{1,\ldots,q} \\ \boldsymbol{U}_{2,\bullet q}(\boldsymbol{U}^*_{\bullet 1})_{q+1,\ldots,2q} \\ \vdots \\ \boldsymbol{U}_{s,\bullet q}(\boldsymbol{U}^*_{\bullet 1})_{(s-1)q+1,\ldots,sq} \end{bmatrix}.$$

It is seen above that only the q first triplets of the SVDs of the $\boldsymbol{M}_i$s are required. So, overall we "only" have to compute s truncated $(q \times q)$ SVDs (of the $\boldsymbol{M}_i$s) and one truncated (1×1) SVD (of the sq first lines of $\tilde{\boldsymbol{H}}$, which are easily obtained from these truncated SVDs).

Moreover, we can compute $\boldsymbol{u}_1$ from $\boldsymbol{v}_1$ using the simple formula $\boldsymbol{u}_1 = \boldsymbol{M}\boldsymbol{v}_1 / \|\boldsymbol{M}\boldsymbol{v}_1\|$ (using a chunk approach).

When q is larger than p, we just partition $\boldsymbol{M}$ in columns instead of rows. When both p and q are large, one can adapt Algorithm 2 in Liang et al. (2016) which generalises the above. They even propose a third algorithm for the case of online (streaming) SVDs.

Note that these algorithms based on the split-and-merge strategy possess an embarrassingly parallel structure and thus can be efficiently implemented on a distributed or multicore machine.

4.3. Incremental SVD when n is large

We want to compute the truncated SVD of the matrix $\boldsymbol{M}_h = \boldsymbol{X}_h^\intercal \boldsymbol{Y}_h$ when n is very large (and the X- and Y-matrices are split in blocks, or chunks, of size n/G for some given G). One can use the divide and conquer approach presented

[1]The transpose sign on $\boldsymbol{R}$ is missing in Liang et al. (2016).

in subsection 4.1 to compute first the matrix $\boldsymbol{M}_h = \boldsymbol{X}_h^\mathsf{T}\boldsymbol{Y}_h$ and then evaluate the SVD of this matrix. We present here an alternative approach to Cardot & Degras (2017) by considering an incremental version of the SVD.

Let $\boldsymbol{X}^\mathsf{T} = [\boldsymbol{x}_1^\mathsf{T}, \ldots, \boldsymbol{x}_n^\mathsf{T}]$ and $\boldsymbol{Y}^\mathsf{T} = [\boldsymbol{y}_1^\mathsf{T}, \ldots, \boldsymbol{y}_n^\mathsf{T}]$ be non-centered data matrices. We note

$$\boldsymbol{M}_n = \dot{\boldsymbol{X}}_n^\mathsf{T}\dot{\boldsymbol{Y}}_n = \sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu}_{X,n})(\boldsymbol{y}_i - \boldsymbol{\mu}_{Y,n})^\mathsf{T}$$

with the centered data matrices

$$\dot{\boldsymbol{X}}_n = \boldsymbol{X}_n - \boldsymbol{1}_n\boldsymbol{\mu}_{X,n}^\mathsf{T}, \quad \dot{\boldsymbol{Y}}_n = \boldsymbol{Y}_n - \boldsymbol{1}_n\boldsymbol{\mu}_{Y,n}^\mathsf{T}$$

where $\boldsymbol{\mu}_{X,n} = n^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{1}_n = n^{-1}\sum_{i=1}^n \boldsymbol{x}_i^\mathsf{T}$ and $\boldsymbol{\mu}_{Y,n} = n^{-1}\boldsymbol{Y}^\mathsf{T}\boldsymbol{1}_n = n^{-1}\sum_{i=1}^n \boldsymbol{y}_i^\mathsf{T}$.

We have the streaming updating formulas

$$\boldsymbol{\mu}_{X,n+1} = \frac{n}{n+1}\boldsymbol{\mu}_{X,n} + \frac{1}{n+1}\boldsymbol{x}_{n+1},$$
$$\boldsymbol{\mu}_{Y,n+1} = \frac{n}{n+1}\boldsymbol{\mu}_{Y,n} + \frac{1}{n+1}\boldsymbol{y}_{n+1},$$

and

$$\boldsymbol{M}_{n+1} \quad = \quad \boldsymbol{M}_n + \frac{n}{(n+1)}(\boldsymbol{x}_{n+1} - \boldsymbol{\mu}_{X,n})(\boldsymbol{y}_{n+1} - \boldsymbol{\mu}_{Y,n})^\mathsf{T}. \tag{13}$$

Now, let the H-rank truncated SVD of $\boldsymbol{M}_n$ be $\boldsymbol{M}_n^{(H)} = \boldsymbol{U}_{n,\bullet H}\boldsymbol{\Delta}_{n,H}\boldsymbol{V}_{n,\bullet H}^\mathsf{T}$. Let $\tilde{\boldsymbol{x}}_{n+1} = \boldsymbol{x}_{n+1} - \boldsymbol{\mu}_{X,n}$ and $\tilde{\boldsymbol{y}}_{n+1} = \boldsymbol{y}_{n+1} - \boldsymbol{\mu}_{Y,n}$. Since $\boldsymbol{U}_{n,\bullet H}^\mathsf{T}\boldsymbol{U}_{n,\bullet H} = \boldsymbol{I}$, we have

$$\begin{aligned}
\tilde{\boldsymbol{x}}_{n+1} &= \mathcal{P}_{\boldsymbol{U}_{n,\bullet H}}\tilde{\boldsymbol{x}}_{n+1} + \mathcal{P}_{\boldsymbol{U}_{n,\bullet H}^\perp}\tilde{\boldsymbol{x}}_{n+1} \\
&= \boldsymbol{U}_{n,\bullet H}\boldsymbol{U}_{n,\bullet H}^\mathsf{T}\tilde{\boldsymbol{x}}_{n+1} + \mathcal{P}_{\boldsymbol{U}_{n,\bullet H}^\perp}\tilde{\boldsymbol{x}}_{n+1} \\
&= \boldsymbol{U}_{n,\bullet H}\boldsymbol{c}_{n+1} + \tilde{\boldsymbol{x}}_{n+1}^\perp
\end{aligned}$$

with $\boldsymbol{c}_{n+1} = \boldsymbol{U}_{n,\bullet H}^\mathsf{T}\tilde{\boldsymbol{x}}_{n+1}$ and $\tilde{\boldsymbol{x}}_{n+1}^\perp = \mathcal{P}_{\boldsymbol{U}_{n,\bullet H}^\perp}\tilde{\boldsymbol{x}}_{n+1}$. Similarly,

$$\tilde{\boldsymbol{y}}_{n+1} = \boldsymbol{V}_{n,\bullet H}\boldsymbol{d}_{n+1} + \tilde{\boldsymbol{y}}_{n+1}^\perp$$

with $\boldsymbol{d}_{n+1} = \boldsymbol{V}_{n,\bullet H}^\mathsf{T}\tilde{\boldsymbol{y}}_{n+1}$ and $\tilde{\boldsymbol{y}}_{n+1}^\perp = \mathcal{P}_{\boldsymbol{V}_{n,\bullet H}^\perp}\tilde{\boldsymbol{y}}_{n+1}$. Now, in view of (13), we have the approximation

$$\boldsymbol{M}_{n+1}^{(H)} \approx \boldsymbol{M}_n^{(H)} + \frac{n}{n+1}\tilde{\boldsymbol{x}}_{n+1}\tilde{\boldsymbol{y}}_{n+1}^\mathsf{T}.$$

Remark 3 *Note that this approximation is in fact exact when $H = rank(\boldsymbol{M}_n)$. So if we want to use this approach in our algorithm, we would have to compute all the singular elements and not only the first triplet. This being said, if for example q is not too large (e.g., $q = 1$ which is common in applications) this is not a*

problem anymore. Moreover, one can easily find that $\boldsymbol{u}_1 = \boldsymbol{X}^\mathsf{T}\boldsymbol{Y}\boldsymbol{v}_1/\|\boldsymbol{X}^\mathsf{T}\boldsymbol{Y}\boldsymbol{v}_1\|$ *and* $\boldsymbol{v}_1 = \boldsymbol{Y}^\mathsf{T}\boldsymbol{X}\boldsymbol{u}_1/\|\boldsymbol{Y}^\mathsf{T}\boldsymbol{X}\boldsymbol{u}_1\|$ *(Lafaye de Micheaux et al. 2017). Note also that* $\boldsymbol{u}_1$ *is the first eigenvector of the* $p \times p$ *matrix* $(\boldsymbol{Y}^\mathsf{T}\boldsymbol{X})^\mathsf{T}\boldsymbol{Y}^\mathsf{T}\boldsymbol{X}$ *whereas* $\boldsymbol{v}_1$ *is the first eigenvector of the* $q \times q$ *matrix* $(\boldsymbol{X}^\mathsf{T}\boldsymbol{Y})^\mathsf{T}\boldsymbol{X}^\mathsf{T}\boldsymbol{Y}$*. We only need to compute either* $\boldsymbol{u}_1$ *(if* $p < q$*) or* $\boldsymbol{v}_1$ *(if* $q \leq p$*), from which we obtain the other one.*

At this point, one can write

$$\boldsymbol{M}_{n+1}^{(H)} = \left[\boldsymbol{U}_{n,\bullet H}, \frac{\tilde{\boldsymbol{x}}_{n+1}^{\perp}}{\|\tilde{\boldsymbol{x}}_{n+1}^{\perp}\|_2}\right] \boldsymbol{Q}_{n+1} \left[\boldsymbol{V}_{n,\bullet H}, \frac{\tilde{\boldsymbol{y}}_{n+1}^{\perp}}{\|\tilde{\boldsymbol{y}}_{n+1}^{\perp}\|_2}\right]^\mathsf{T}$$

with

$$\boldsymbol{Q}_{n+1} = \frac{n}{n+1}\begin{pmatrix} \frac{n+1}{n}\boldsymbol{\Delta}_n + \boldsymbol{c}_{n+1}\boldsymbol{d}_{n+1}^\mathsf{T} & \|\tilde{\boldsymbol{y}}_{n+1}^{\perp}\|_2 \boldsymbol{c}_{n+1} \\ \|\tilde{\boldsymbol{x}}_{n+1}^{\perp}\|_2 \boldsymbol{d}_{n+1}^\mathsf{T} & \|\tilde{\boldsymbol{x}}_{n+1}^{\perp}\|_2 \|\tilde{\boldsymbol{y}}_{n+1}^{\perp}\|_2 \end{pmatrix}.$$

It then suffices to perform the SVD of the matrix $\boldsymbol{Q}_{n+1}$ of dimension $(H+1) \times (H+1)$. Writing $\boldsymbol{Q}_{n+1} = \boldsymbol{A}_{n+1}\boldsymbol{S}_{n+1}\boldsymbol{B}_{n+1}^\mathsf{T}$, we have

$$\boldsymbol{M}_{n+1}^{(H)} = \boldsymbol{U}_{n+1}\boldsymbol{\Delta}_{n+1}\boldsymbol{V}_{n+1}^\mathsf{T}$$

with $\boldsymbol{\Delta}_{n+1} = \boldsymbol{S}_{n+1}$,

$$\boldsymbol{U}_{n+1} = \left[\boldsymbol{U}_n, \frac{\tilde{\boldsymbol{x}}_{n+1}^{\perp}}{\|\tilde{\boldsymbol{x}}_{n+1}^{\perp}\|_2}\right] \boldsymbol{A}_{n+1}$$

and

$$\boldsymbol{V}_{n+1} = \left[\boldsymbol{V}_n, \frac{\tilde{\boldsymbol{y}}_{n+1}^{\perp}}{\|\tilde{\boldsymbol{y}}_{n+1}^{\perp}\|_2}\right] \boldsymbol{B}_{n+1}.$$

To keep the approximation $\boldsymbol{M}_{n+1}^{(H)}$ of $\boldsymbol{M}_{n+1}$ at rank H, the row and column of $\boldsymbol{\Delta}_{n+1}$ containing the smallest singular value are deleted and the associated singular vectors are deleted from $\boldsymbol{U}_{n+1}$ and $\boldsymbol{V}_{n+1}$.

This incremental way to compute the SVD provides a promising alternative for handling very large sample sizes (especially when q is not too large). Moreover, the incremental SVD is well suited to a data stream context.

5. Numerical Experiments

In this section, we use the previously mentioned packages and our new R package to carry out a short simulation study in order to illustrate the numerical behavior of the new proposed approach. The experiments have been conducted using a laptop with a 2.53 GHz processor and 8 GB of memory. The parallel strategy utilises four processor cores.

We present two simulations to illustrate the good performance of the proposed approaches and the scalability to large sample sizes of our algorithm. The first simulation considers the PLS-R model (case (iv)) on group structure data while the second simulation presents an extension of PLS approaches to discriminant analysis.

5.1. Group PLS model

We generate data with a group structure: 20 groups of 20 variables for $\boldsymbol{X}$ ($p = 400$) and 25 groups of 20 variables for $\boldsymbol{Y}$ ($q = 500$). To highlight the scalability of our algorithm, we generate two big matrices from the following models linked by $H = 2$ latent variables:

$$\boldsymbol{X} = \boldsymbol{\Xi}_H \boldsymbol{C}_H^{\mathsf{T}} + \boldsymbol{F}_H^X, \qquad \boldsymbol{Y} = \boldsymbol{\Xi}_H \boldsymbol{D}_H^{\mathsf{T}} + \boldsymbol{F}_H^Y, \tag{14}$$

where the matrix $\boldsymbol{\Xi}_H = (\boldsymbol{\xi}_j)$ contains 2 latent variables $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$. The entries in these vectors have all been independently generated from a standard normal distribution. The rows of the residual matrix $\boldsymbol{F}_H^X$ (respectively, $\boldsymbol{F}_H^Y$) have been generated from a multivariate normal distribution with zero mean $\boldsymbol{\mu}_X$ (resp. $\boldsymbol{\mu}_Y$) and covariance matrix $\boldsymbol{\Sigma}_X = 1.5^2 \boldsymbol{I}_p$ (resp. $\boldsymbol{\Sigma}_Y = 1.5^2 \boldsymbol{I}_q$).

Among the 20 groups of $\boldsymbol{X}$, only four groups each containing 15 true variables and five noise variables are associated with the response variables of $\boldsymbol{Y}$. We set the p-vector $\boldsymbol{c}_1$ (first column of the $\boldsymbol{C}_H$ matrix) to have 15 1's, 30 -1's and 15 1.5's, the other entries being all set to 0. All 15 non-zero coefficients are assigned randomly into one group along with the remaining 5 zero coefficients corresponding to noise variables. The vector $\boldsymbol{c}_2$ is chosen in the same way as $\boldsymbol{c}_1$. The two columns of $\boldsymbol{D}_H$ are q-vectors containing 15 -1's, 15 -1.5's and 30 1's and the rest are 0's such that the matrix $\boldsymbol{Y}$ has a similar group structure for four groups containing the signal. Finally, the sample size is set to $n = 560,000$ observations which corresponds to storage requirements of approximately 5 GB for each matrix, thus with a total exceeding the 8 GB of memory available on our laptop.

The top four plots of Figure 1 show the results of the group PLS estimated with only $n = 100$ observations. For such a sample size, the usual group PLS can be used without any computational time or memory issues. In this case, group PLS manages to select the relevant groups and performs well to estimate the weight vectors $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ related to the first component and the weight vectors $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$ related to the second component.

The bottom four plots of Figure 1 show the results of the group PLS estimated on the **full** data set which can be only analysed by using the extended version of our algorithm for big data. In this run, we use $G = 100$ chunks for enabling matrix multiplication. The execution time was around 15 minutes for two components ($H = 2$) and took less than 2 minutes to get the first component. We can observe that the signal has been perfectly identified and estimated, which is expected for such a huge amount of information.

Note that for validation purposes, the extended version of our algorithm for big data has been executed and exactly matched the usual algorithm on the small data set ($n = 100$).

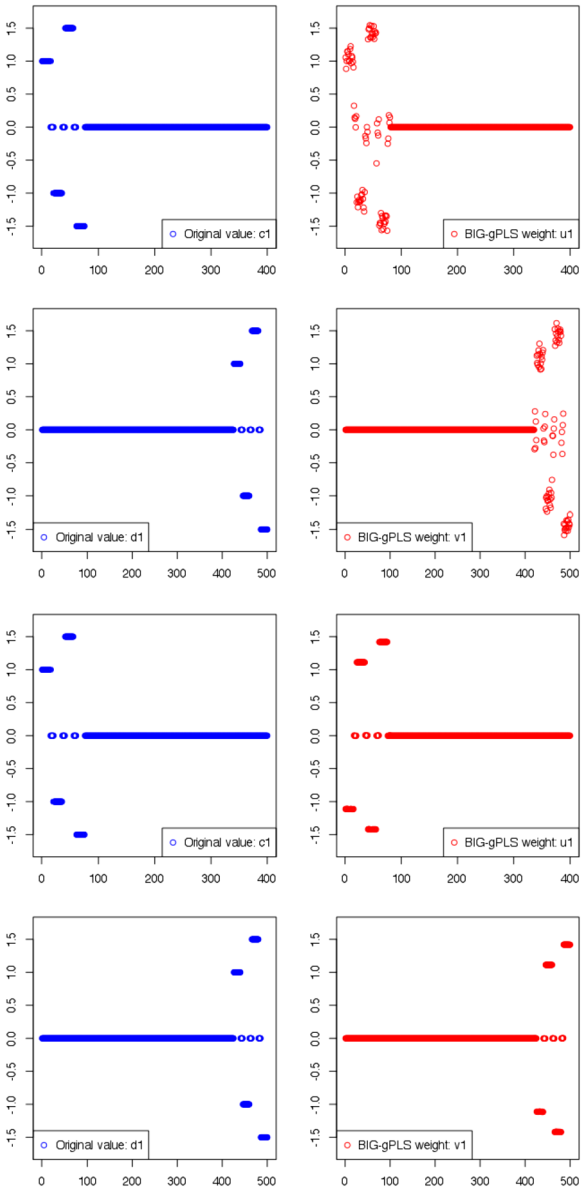

FIG 1. *Comparison of the signal recovered (weights* $\boldsymbol{u}_1$ *and* $\boldsymbol{v}_1$*) by the first component* ($H = 2$) *of the gPLS. For the top four plots,* $n = 100$*, and for the four bottom plots* $n = 560,000$*. Left column: the true values of* $\boldsymbol{c}_1$ *and* $\boldsymbol{d}_1$ *for small and large sample sizes. Right column: the estimated values of* $\boldsymbol{c}_1$ *and* $\boldsymbol{d}_1$ *for small and large sample sizes. Note that the values of* $\boldsymbol{u}_1$ *(resp.* $\boldsymbol{v}_1$*) have been rescaled so that its norm equals that of the original* $\boldsymbol{c}_1$ *(resp.* $\boldsymbol{d}_1$*).*

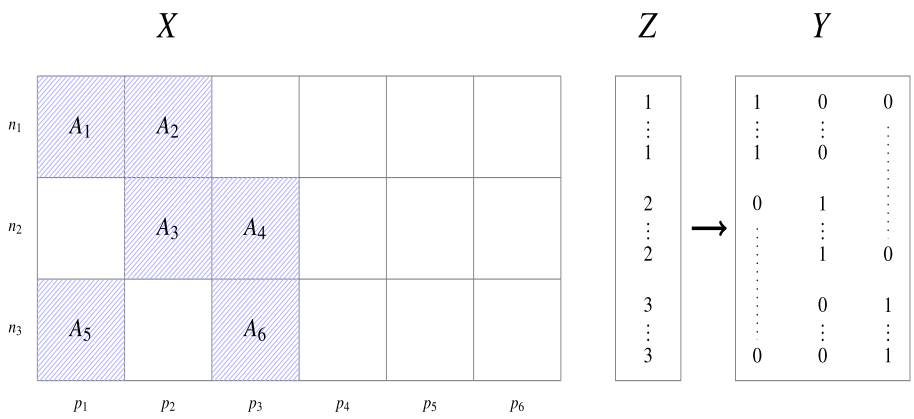

FIG 2. *Discriminant Analysis Design Matrices*

5.2. Case of regularised PLS-DA

We consider here the case of qualitative response variables for discrimination analysis. In this framework, PLS approaches have often been used (Nguyen & Rocke 2002) by recoding the qualitative response as a dummy block matrix $\boldsymbol{Y} : n \times c$ indicating the class of each sample (c being the number of categories). One can also directly apply PLS regression on the data as if $\boldsymbol{Y}$ was a matrix with continuous entries (from now on called PLS-DA). Note that Barker & Rayens (2003) give some theoretical justification for this approach. A group and a sparse group version have been proposed by Liquet et al. (2016) using only penalties on the loading related to the variables in $\boldsymbol{X}$. Our unified algorithm is then naturally extended in the same way to deal with categorical variables. We illustrate it on a big data set defined as follows. Let A_k be the set of indices (i, j) of the i-th observation and j-th variable that are associated with the corresponding gray cell as shown in Figure 2.

We have $\forall k = 1, \ldots, 6,\ \forall i = 1, \ldots, n, \forall j = 1, \ldots, p$

$$X_{i,j} = \mu_k \times 1_{\{(i,j) \in A_k\}} + \epsilon_{i,j}$$

where $\boldsymbol{\mu}^{\mathsf{T}} = (\mu_1, \ldots, \mu_6) = (-1.0, 1.5, 1.0, 2.5, -0.5, 2.0)$, and $\epsilon_{i,j} \overset{iid}{\sim} N(0, 1)$. As illustrated on Figure 2, the matrix $\boldsymbol{X}$ is composed of 6 groups of $p_k = 100$ variables ($p = \sum_{k=1}^{6} p_k = 600$) and each of the 3 categories of the response variable are linked to two groups of variables. We used a sample size of $n = 486,000$ which corresponds to storage requirements of approximately 5 GB for the $\boldsymbol{X}$ matrix. We use $G = 100$ chunks for computing the different matrix products. The run took around 9 minutes for a model using 2 components. The relevant groups have been selected in both components. We randomly sample $9,000$ observations and present in Figure 3 their projection on the two components estimated on the full data set. A nice discrimination of the 3 categories of the response variable is observed.

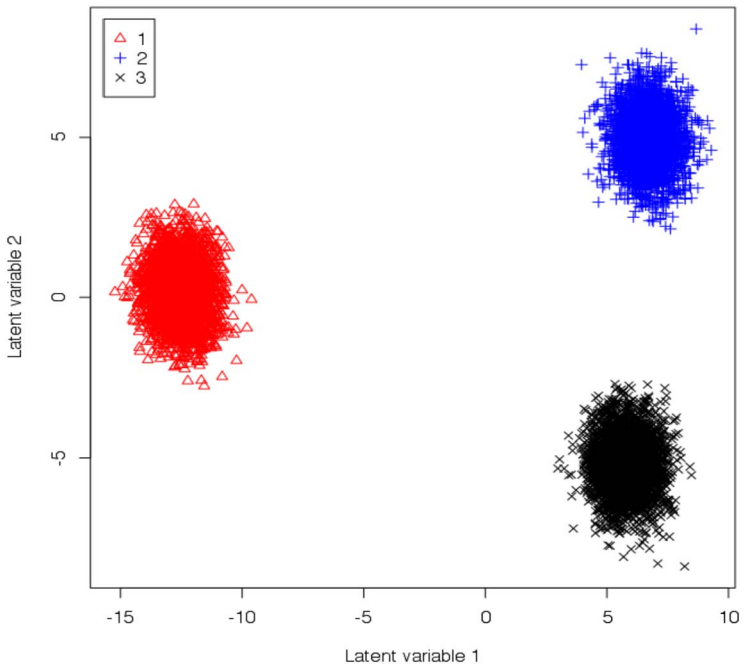

FIG 3. *Group PLS-DA on a big data set*

5.3. The EMNIST data set

The now famous MNIST data set (LeCun & Cortes 2010) has become a standard benchmark for classification systems. The Extended MNIST (EMNIST), another freely available data set, "constitutes a more challenging classification task involving letters and digits, and that shares the same image structure and parameters as the original MNIST task" (Cohen et al. 2017). This data set consists of $n = 280,000$ handwritten digit images. It contains an equal number of samples for each digit class (0 to 9). The images are already split into a training set of 240,000 cases and a test set of 40,000 cases. Each case is a 28×28 pixels gray-scale image whose intensity values range in $[0, 255]$. Overall, if imported into R, they would use around 1.6 GB. We applied our PLS-DA algorithm on this data set, where the matrix $\mathbf{X} : n \times p$ ($p = 784$) contains the images, and the matrix $\mathbf{Y} : n \times q$ ($q = 10$) indicates the label of each image. We ran a PLS-DA model with 20 latent components which enabled us to get an accuracy (percentage of correct classification) of 86% in around 3 minutes using 2 cores. Note that in Cohen et al. (2017), linear classifiers produced an accuracy of around 85% for the EMNIST data. We also considered a larger version of the EMNIST data set where each image in the training set is rotated 5 and 10 degrees in both a clockwise and counterclockwise direction. The enlarged data set contains 1,200,000 images and uses 7.5 GB of memory. It is more representative of data sets which

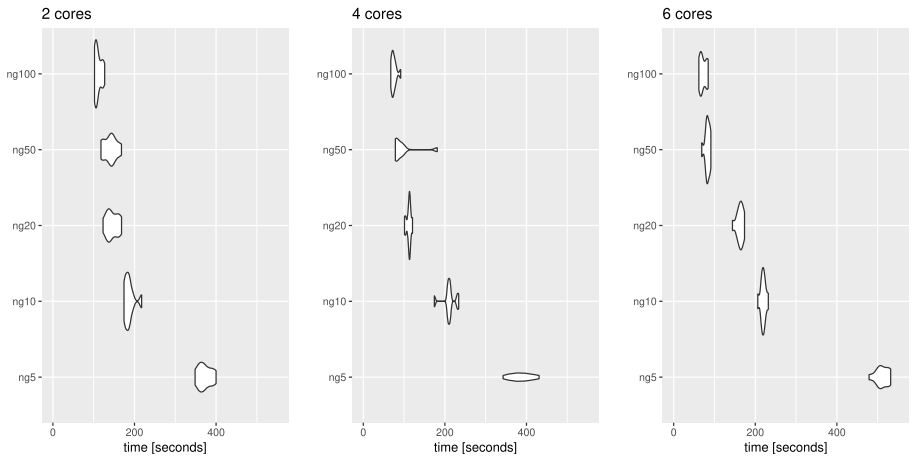

FIG 4. *Wine plot of the elapsed computation time for fitting a single component of the PLS discriminant analysis algorithm using 2, 4 or 6 cores (on a laptop equipped with 8 cores). On the vertical axis,* `ngx` *indicates that* x *chunks were used in our algorithm. The wine plot is obtained from ten independent runs on the same data set.*

are too big to be loaded into a standard R session. The PLS-DA model with 20 latent components obtained an accuracy of 85.5% in around 18 minutes with 2 cores.

We also explored different versions of our algorithm by varying the number of chunks (`ng` option) to compute the different cross product matrices required during the run. The algorithm was run 10 times on the rotated data set and we investigated the effect of using 2 to 6 cores on the computation time to obtain the first component.

The results illustrated in Figure 4 indicate that increasing the number of chunks (for a fixed number of cores) leads to better computational performance. However, it is worth noticing that for `ng` set to a small value (5 say), increasing the number of cores had a negative effect on computation time. This increase may be due to an increased computational overhead involved with splitting and recombining the data sets with more cores. This computational trade-off can be hard to predict, but we have found that using more cores and a larger `ng` for big data improves the computational performance.

6. Conclusion and future work

This paper surveys four popular partial least squares methods and unifies these methods with recent variable selection techniques based on penalised singular value decomposition. We present a general framework for both symmetric and asymmetric penalised PLS methods and showcase some possible convex penalties. A unified algorithm is described and implemented for the penalised PLS methods, and we offer further extensions to deal with massive data sets (n, p and

q very large). A full comparison in terms of time and memory of the different proposed extensions is an open area of future research.

Aside from computational issues, it is unclear if retaining the deflations of the usual PLS methods is appropriate when there is penalisation. In particular, we note that the orthogonality constraints of the original PLS methods are not retained for the penalised methods. Further development of our methods could seek to preserve the orthogonality constraints. We are perusing this open area using ideas from Tibshirani & Taylor (2011), and Mackey (2009) for the simple lasso penalty. However, further investigation is required in the context of more complex penalties such as group or sparse group penalties.

References

Abdi, H. & Williams, L. J. (2013), 'Partial least squares methods: partial least squares correlation and partial least square regression', *Methods Mol. Biol.* **930**, 549–579.

Alin, A. (2009), 'Comparison of pls algorithms when number of objects is much larger than number of variables', *Statistical Papers* **50**, 711–720. MR2551345

Allen, G. I., Grosenick, L. & Taylor, J. (2014), 'A generalized least-square matrix decomposition', *Journal of the American Statistical Association* **109**(505), 145–159. MR3180553

Allen, G. I., Peterson, C., Vannucci, M. & Maletic-Savatic, M. (2013), 'Regularized Partial Least Squares with an Application to NMR Spectroscopy', *Statistical Analysis and Data Mining* **6**(4), 302–314. MR3092060

Allen, G. I. & Tibshirani, R. (2010), 'Transposable regularized covariance models with an application to missing data imputation', *Ann Appl Stat* **4**(2), 764–790. MR2758420

Baglama, J. & Reichel, L. (2015), *irlba: Fast Truncated SVD, PCA and Symmetric Eigendecomposition for Large Dense and Sparse Matrices*. R package version 2.0.0. http://CRAN.R-project.org/package=irlba

Barker, M. & Rayens, W. (2003), 'Partial least squares for discrimination', *Journal of Chemometrics* **17**(3), 166–173.

Boulesteix, A.-L. & Strimmer, K. (2007), 'Partial least squares: a versatile tool for the analysis of high-dimensional genomic data', *Briefings in Bioinformatics* **8**(1), 32–44.

Brown, P. J. & Zidek, J. V. (1980), 'Adaptive multivariate ridge regression', *Ann. Statist.* **8**(1), 64–74. https://doi.org/10.1214/aos/1176344891 MR0557554

Cak, A. D., Moran, E. F., de O. Figueiredo, R., Lu, D., Li, G. & Hetrick, S. (2016), 'Urbanization and small household agricultural land use choices in the Brazilian amazon and the role for the water chemistry of small streams', *Journal of Land Use Science* **11**(2), 203–221.

Cardot, H. & Degras, D. (2017), 'Online principal component analysis in high dimension: Which algorithm to choose?', *International Statistical Review*. http://dx.doi.org/10.1111/insr.12220 MR3796510

Chen, X. & Liu, H. (2012), 'An efficient optimization algorithm for structured sparse cca, with applications to eqtl mapping', *Statistics in Biosciences* **4**(1), 3–26. MR3012522

Chun, H. & Keleş, S. (2010), 'Sparse partial least squares regression for simultaneous dimension reduction and variable selection', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**(1), 3–25. MR2751241

Chung, D. & Keleş, S. (2010), 'Sparse Partial Least Squares Classification for High Dimensional Data', *Statistical Applications in Genetics and Molecular Biology* **9**(1), 17. MR2721697

Cohen, G., Afshar, S., Tapson, J. & van Schaik, A. (2017), 'EMNIST: an extension of MNIST to handwritten letters', *CoRR* **abs/1702.05373**. http://arxiv.org/abs/1702.05373

de Jong, S. (1993), 'Simpls: an alternative approach to partial least squares regression', *Chemometrics and Intelligent Laboratory Systems* **18**, 251–263.

Dhanjal, C., Gunn, S. R. & Shawe-Taylor, J. (2009), 'Efficient sparse kernel feature extraction based on partial least squares', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**(8), 1347–1361.

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22. http://www.jstatsoft.org/v33/i01/

Friedman, J., Hastie, T., Tibshirani, R., Simon, N., Narasimhan, B. & Qian, J. (2018), *glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 2.0-16. https://CRAN.R-project.org/package=glmnet

Geladi, P. & Kowalski, B. R. (1986), 'Partial least-squares regression: a tutorial', *Analytica Chimica Acta* **185**, 1–17.

Guo, G. & Mu, G. (2013), Joint estimation of age, gender and ethnicity: Cca vs. pls, *in* '10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)', pp. 1–6.

Hardoon, D. R., Szedmak, S. & Shawe-Taylor, J. (2004), 'Canonical correlation analysis: an overview with application to learning methods', *Neural Computation* **16**(12), 2639–2664.

Hastie, T., Tibshirani, R. & Friedman, J. H. (2009), *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*, Springer series in statistics, Springer. http://www.worldcat.org/oclc/300478243 MR2722294

Höskuldsson, A. (1988), 'Pls regression methods', *Journal of Chemometrics* **2**, 211–228.

Hotelling, H. (1936), 'Relations between two sets of variates', *Biometrika* **28**(3-4), 321.

Ji, G., Yang, Z. & You, W. (2011), 'Pls-based gene selection and identification of tumor-specific genes', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **41**(6), 830–841.

Kraemer, N. & Sugiyama, M. (2011), 'The degrees of freedom of partial least squares regression', *Journal of the American Statistical Association* **106**(494). MR2847952

Krishnan, A., Williams, L. J., McIntosh, A. R. & Abdi, H. (2011), 'Partial least

squares (pls) methods for neuroimaging: A tutorial and review', *NeuroImage* **56**(2), 455 – 475.

Lafaye de Micheaux, P., Liquet, B. & Sutton, M. (2017), 'A Unified Parallel Algorithm for Regularized Group PLS Scalable to Big Data', *ArXiv e-prints*.

Lê Cao, K.-A., Rossouw, D., Robert-Granié, C. & Besse, P. (2008), 'Sparse PLS: Variable Selection when Integrating Omics data', *Statistical Application and Molecular Biology* **7**((1):37). MR2457048

LeCun, Y. & Cortes, C. (2010), 'MNIST handwritten digit database'. http://yann.lecun.com/exdb/mnist/

Liang, F., Shi, R. & Mo, Q. (2016), 'A split-and-merge approach for singular value decomposition of large-scale matrices', *Statistics And Its Interface* **9**(4), 453–459. MR3553373

Lin, D., Cao, H., Calhoun, V. D. & Wang, Y.-P. (2014), 'Sparse models for correlative and integrative analysis of imaging and genetic data', *Journal of Neuroscience Methods* **237**, 69 – 78.

Lindgren, F. & Rännar, S. (1998), 'Alternative partial least squares (pls) algorithms', *Perspectives Drug Discovery and Design* pp. 105–113.

Liquet, B., Lafaye de Micheaux, P., Hejblum, B. & Thiébaut, R. (2016), 'Group and sparse group partial least square approaches applied in genomics context', *Bioinformatics* **32**, 35–42.

Liu, J. & Calhoun, V. D. (2014), 'A review of multivariate analyses in imaging genetics', *Frontiers in Neuroinformatics* **8**(29).

Lockhart, R., Taylor, J., Tibshirani, R. J. & Tibshirani, R. (2014), 'A significance test for the lasso', *Ann Stat* **42**(2), 413–468. MR3210970

Lorenzi, M., Gutman, B., Hibar, D. P., Altmann, A., Jahanshad, N., Thompson, P. M. & Ourselin, S. (2016), Partial least squares modelling for imaging-genetics in Alzheimer's disease: Plausibility and generalization, *in* '2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)', pp. 838–841.

Lütkepohl, H. (2005), *New introduction to multiple time series analysis*, Springer-Verlag, Berlin. MR2172368

Mackey, L. W. (2009), Deflation methods for sparse pca, *in* D. Koller, D. Schuurmans, Y. Bengio & L. Bottou, eds, 'Advances in Neural Information Processing Systems 21', Curran Associates, Inc., pp. 1017–1024.

Mardia, K. V., Kent, J. T. & Bibby, J. M. (1979), *Multivariate analysis / K.V. Mardia, J.T. Kent, J.M. Bibby*, Academic Press London; New York. MR0560319

McIntosh, A. R., Bookstein, F. L., Haxby, J. V. & Grady, C. L. (1996), 'Spatial pattern analysis of functional brain images using partial least squares', *NeuroImage* **3**(3), 143–157.

Meyer, C. D. (2000), *Matrix Analysis and Applied Linear Algebra*, SIAM. MR1777382

Netrapalli, P., Jain, P. & Sanghavi, S. (2015), 'Phase retrieval using alternating minimization', *IEEE Transactions on Signal Processing* **63**(18), 4814–4826. MR3385838

Nguyen, D. & Rocke, D. (2002), 'Tumor classification by partial least squares using microarray gene expression data', *Bioinformatics* **18**(1), 39–50.

Nicole Kraemer, M. L. B. (2018), *plsdof: Degrees of Freedom and Statistical Inference for Partial Least Squares Regression*. R package version 0.2-8. https://CRAN.R-project.org/package=plsdof

Nielsen, F. A. (2002), Neuroinformatics in Functional Neuroimaging, PhD thesis, Technical University of Denmark, Lyngby.

Palermo, R. E., Patterson, L. J., Aicher, L. D., Korth, M. J., Robert-Guroff, M. & Katze, M. G. (2011), 'Genomic analysis reveals pre- and postchallenge differences in a rhesus macaque aids vaccine trial: Insights into mechanisms of vaccine efficacy', *Journal of Virology* **85**(2), 1099–1116.

Phatak, A. & de Jong, S. (1997), 'The geometry of partial least squares', *Journal of Chemometrics* **11**(4), 311–338.

R Core Team (2017), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Rohlf, F. J. & Corti, M. (2000), 'Use of two-block partial least-squares to study covariation in shape', *Systematic Biology* **49**(4), 740–753.

Roon, P. V., Zakizadeh, J. & Chartier, S. (2014), 'Partial least squares tutorial for analyzing neuroimaging data', *The Quantitative Methods for Psychology* **10**(2), 200–215.

Rosipal, R. & Krämer, N. (2006), Overview and recent advances in partial least squares, *in* 'Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop', pp. 34–51.

S. E. Leurgans, R. A. Moyeed, B. W. S. (1993), 'Canonical correlation analysis when the data are curves', *Journal of the Royal Statistical Society. Series B (Methodological)* **55**(3), 725–740. MR1223939

Shen, H. & Huang, J. Z. (2008), 'Sparse principal component analysis via regularized low rank matrix approximation', *Journal of Multivariate Analysis* **99**(6), 1015 – 1034. MR2419336

Simon, N., Friedman, J., Hastie, T. & Tibshirani, R. (2013), 'A sparse-group lasso', *Journal of Computational and Graphical Statistics* **22**(2), 231–245. MR3173712

Sutton, M., Thiebaut, T. & Liquet, B. (2018), 'Sparse partial least squares with group and subgroup structure', *Statistics in Medicine* **37**(23), 3338–33356.

Tenenhaus, M. (1998), *La régression PLS: Théorie et Pratique*, Paris: Technip. MR1645125

ter Braak, C. J. F. & de Jong, S. (1998), 'The objective function of partial least squares regression', *Journal of Chemometrics* **12**(1), 41–54. MR2695725

Tibshirani, R. (1994), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society, Series B* **58**, 267–288. MR1379242

Tibshirani, R. J. & Taylor, J. (2011), 'The solution path of the generalized lasso', *Annals of Statistics* **39**(3), 1335–1371. MR2850205

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J. & Knight, K. (2005), 'Sparsity and smoothness via the fused lasso', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **67**(1), 91–108. MR2136641

Tibshirani, R., Tibshirani, R., Taylor, J., Loftus, J. & Reid, S. (2017), *selectiveInference: Tools for Post-Selection Inference*. R package version 1.2.4.

https://CRAN.R-project.org/package=selectiveInference

Tseng, P. (1988), Coordinate ascent for maximising nondifferentiable concave functions, Technical report, Massachusetts Institute of Technology. Laboratory for Information and Decision Systems.Cambridge MA.

Vinod, H. (1976), 'Canonical ridge and econometrics of joint production', *Journal of Econometrics* **4**(2), 147 – 166.

Vinzi, V., Trinchera, L. & Amato, S. (2010), 'Pls path modeling: from foundations to recent developments and open issues for model assessment and improvement', *Handbook of Partial Least Squares* pp. 47–82. MR2762257

Wegelin, J. A. (2000), A survey of partial least squares (pls) methods, with emphasis on the two-block case, Technical report, University of Washington.

Witten, D. M., Tibshirani, R. & Hastie, T. (2009), 'A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis', *Biostatistics* **10**(3), 515–534.

Wold, H. (1966), Estimation of principal components and related models by iterative least squares, *in* 'Multivariate Analysis', Academic Press, New York, Wiley, Dayton, Ohio, pp. 391–420. MR0220397

Wold, S., Ruhe, A., Wold, H. & Dunn, W. J. (1984), 'The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses', *SIAM Journal on Scientificic and Statistical Computing* **5**(3), 735–743.

Wold, S., Sjöström, M. & Eriksson, L. (2001), 'Pls-regression: a basic tool of chemometrics', *Chemometrics and Intelligent Laboratory Systems* **58**(2), 109 – 130.

Yee, T. W. (2018), *VGAM: Vector Generalized Linear and Additive Models*. R package version 1.0-6. https://CRAN.R-project.org/package=VGAM

Yee, T. W. & Wild, C. J. (1996), 'Vector generalized additive models', *Journal of the Royal Statistical Society. Series B (Methodological)* **58**(3), 481–493. http://www.jstor.org/stable/2345888 MR1394361

Yeniay, O. & Goktas, A. (2002), 'A comparison of partial least squares regression with other prediction methods', *Hacettepe Journal of Mathematics and Statistics* **31**(99), 99–101. MR1987061

Yuan, M. & Lin, Y. (2006), 'Model selection and estimation in regression with grouped variables', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68**(1), 49–67. MR2212574

Zeng, Y. & Breheny, P. (2017*a*), 'The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in r', *ArXiv e-prints*. https://arxiv.org/abs/1701.05936

Zeng, Y. & Breheny, P. (2017*b*), *The biglasso Package: A Memory- and Computation-Efficient Solver for Lasso Model Fitting with Big Data in R*. R package version 1.3. https://CRAN.R-project.org/package=biglasso