Sparse and Kernel OPLS feature extraction based on eigenvalue problem solving

Sergio Muñoz-Romero*, Jerónimo Arenas-García, Vanessa Gómez-Verdejo
Department of Signal Theory and Communications, Universidad Carlos III de Madrid, 28911 Leganés, Spain

Abstract

Orthonormalized partial least squares (OPLS) is a popular multivariate analysis method to perform supervised feature extraction. Usually, in machine learning papers OPLS projections are obtained by solving a generalized eigenvalue problem. However, in statistical papers the method is typically formulated in terms of a reduced-rank regression problem, leading to a formulation based on a standard eigenvalue decomposition. A first contribution of this paper is to derive explicit expressions for matching the OPLS solutions derived under both approaches and discuss that the standard eigenvalue formulation is also normally more convenient for feature extraction in machine learning. More importantly, since optimization with respect to the projection vectors is carried out without constraints via a minimization problem, inclusion of penalty terms that favor sparsity is straightforward. In the paper, we exploit this fact to propose modified versions of OPLS. In particular, relying on the ℓ_1 norm, we propose a sparse version of linear OPLS, as well as a non-linear kernel OPLS with pattern selection. We also incorporate a group-lasso penalty to derive an OPLS method with true feature selection. The discriminative power of the proposed methods is analyzed on a benchmark of classification problems. Furthermore, we compare the degree of sparsity achieved by our methods and compare them with other state-of-the-art methods for sparse feature extraction.

Keywords: Partial least squares, orthonormalized PLS, *lasso* regularization, feature extraction, sparse kernel representation

1. Introduction

Most data analysis methods are able to deal efficiently with data in one or a few dimensions. However, when they are applied to real world problems that involve high-dimensional patterns, numerical and overfitting problems easily emerge. In these cases, a previous feature extraction stage,

☆ This work was partly supported by MINECO projects TEC2011-22480 and PRI-PIBIN-2011-1266.

Corresponding author. Tel.: +34 91 624 6251; fax: +34 91 624 8749.
Email addresses: smunoz@tsc.uc3m.es (Sergio Muñoz-Romero), jarenas@tsc.uc3m.es (Jerónimo Arenas-García), vanessa@tsc.uc3m.es (Vanessa Gómez-Verdejo)

which allows to reduce data dimensionality and to remove collinearity among variables, is crucial to appropriately and efficiently apply these data analysis techniques. For this reason, feature extraction techniques, and in particular Multivariate Analysis (MVA) methods [1], have been successful in many different applications of machine learning, such as biomedical engineering [2, 3], remote sensing [4, 5], or chemometrics [6], among others.

Multivariate Analysis (MVA) aggregates a family of methods that build a new set of features by extracting highly correlated projections of data representations in input and output spaces. Well-known representatives of these methods are Principal Component Analysis (PCA) [7], Partial Least Squares (PLS) approaches [1, 8], or Canonical Correlation Analysis (CCA) [9]. The PCA algorithm creates a new data representation space by finding the directions of largest input variance, providing an optimum set of features in terms of mean squared reconstruction error. Unlike other MVA methods, PCA works in an unsupervised manner, i.e., it only considers the input data and does not take into account any target data. The Partial Least Squares (PLS) approach refers to a family of techniques which, in its general form, project both input and output variables to a new space, with the general objective of maximizing the covariance of the projected expressions. In CCA, the aim is to find linear projections of the input and output data to maximize correlation between the projected data sets. Thus, in contrast to PLS, CCA is accounting for correlation rather than covariance.

In this paper, we focus on a fourth MVA method known as Orthonormalized PLS (OPLS) [10], that has also been referred to in the literature as semipenalized CCA [6], multilinear regression (MLR) [11], or reduced-rank regression (RRR) [12, 13]. OPLS is known to be optimum in the mean square error sense for performing multilinear regression [14, 15]; thus, it is very competitive as a pre-processing step in classification and regression problems [4, 15, 16].

Although MVA techniques allow to reduce the data dimensionality, making it easier to handle large dimensional datasets when irrelevant or noisy features are present, the new projected data result from a combination of all original features, including uninformative variables. This behavior is quite undesirable, as it is stated in the *bet-on-sparsity* principle [17], and it would be desirable to obtain a solution consisting only of relevant features. In this way, not only more accurate solutions are usually achieved, but also more interpretable ones.

An direct way to perform feature selection is by favoring sparse solutions which automatically assign null coefficients to variables that are irrelevant for the task. For this reason, since Tibshirani proposed the *lasso* method as a way to induce sparsity [18], many researchers have focused their work on ℓ_1-norm minimization approaches. Lasso method includes an ℓ_1-norm regularization term in the minimization problem to favor sparse solutions. The ease of this technique to remove irrelevant features has brought on a large number of research papers on the topic during previous years, not only in classification and regression problems [19, 20, 21], but also proposing sparse extensions of MVA techniques, such as, the sparse PCA and CCA methods of [22, 23] and [24], respectively. A sparse OPLS has been introduced in [2]; unfortunately, this method does not guarantee orthogonality of the projected input data, thus convergence to the standard OPLS solution is not assured when the sparsity constraints are removed.

Despite the variety of MVA methods described above (both sparse and non-sparse), all of them deal with linear projections, which prevents them from exploiting non-linear relationships among the variables. To address this issue, several authors have proposed kernel variants [25, 27] where the input and/or output data are mapped by a non-linear function into a high-dimensional space in which ordinary linear MVA is performed on the transformed data. Most MVA methods have been reformulated into a kernel framework: kernel PCA [28], kernel CCA [29], kernel PLS [30], and kernel OPLS [15]. The main advantage of these kernel extensions relies on the fact that one obtains the flexibility of non-linear expressions while still solving only linear equations. For this reason, kernel MVA (kMVA) methods have been applied to a wide variety of fields characterized by non-linear relations, including remote sensing data analysis [4, 5], functional magnetic resonance imaging [31], or facial expression recognition [32], among others. On the down side, direct formulations of kernel MVA scale quadratically with the number of training data, making them unfeasible (or at least impractical) for data sets containing just a few thousands of patterns. Furthermore, unless appropriately regularized, these methods can easily overfit the training data [25, 27]. To counter these undesired properties, several sparse kMVA methods have been proposed, see e.g., [33, 34, 15, 16]. Note that when referring to sparse kMVA methods, pattern selection instead of variable selection is generally assumed.

In this paper, we address the issue of sparsity in linear and kernel OPLS. We do so by recurring to an OPLS formulation that places optimization constraints on the regression coefficients instead of the projection vectors, and leads to a standard eigenvalue decomposition (EVD) problem. This formulation, to which we will refer as EVD-OPLS, is well-known in the statistics community [12] but has not been so-widely applied in the machine learning field. The EVD formulation opens the door to modified versions of OPLS that impose additional constraints on the projection vectors, a fact that we will exploit to implement sparse versions of linear and non-linear OPLS.

More specifically, the main contributions of this paper are:

- We derive explicit expressions that show the equivalence between the EVD solution to OPLS and the solution based on a generalized eigenvalue (GEV) problem, that is more common in the machine learning field (see, e.g., [4, 13, 15, 35]). We will discuss that when the number of target variables is less than the input data dimensionality, the EVD formulation is more efficient in computational terms.

- A sparse variation of linear OPLS, which is based on the EVD formulation and the addition of an ℓ_1 regularization term. Although we can find in the literature attempts to use EVD to obtain sparse OPLS solutions [36, 37], they are based on the Procrustes solution. We will show that the schemes of [36, 37] present some convergence problems, and that they may fail to progress at all if the ℓ_1 term is removed.

- In a similar way, we propose an OPLS extension that incorporates group-lasso penalty to enforce sparsity in the projection matrix row-wise. As before, the solution we propose here avoids the problems associated to methods that rely on the Procrustes solution [36], and leads to more discriminative projections.
- Finally, we extend the EVD solution to the kernel framework. Previous proposals of ℓ_1 sparse OPLS have only been proposed in the input space, so the proposed sparse Kernel OPLS approach is, to the best of our knowledge, completely novel.

The rest of the paper is organized as follows: In the next section we review in detail the EVD and GEV formulations of the OPLS problem, demonstrate that they lead to the same solution, and analyze them in terms of computational efficiency. Then, in Sections III and IV we exploit the flexibility of the EVD-OPLS formulation to derive sparse extensions of OPLS in the linear and kernel cases, respectively. In Section V, the experimental assessment of the new methods will be carried out in a benchmark of classification problems and in a face recognition task, analyzing the discriminative power of the extracted features and the degree of sparsity achieved by the solutions. Finally, Section VI presents the main conclusions of our work.

2. Orthonormalized Partial Least Squares

In this section, we review two different implementations of OPLS and derive the expressions that characterize the matching between these solutions. To the best of our knowledge, this connection has not been established before, and is therefore a first contribution of the paper. Both OPLS formulations will also be compared in terms of computational requirements. Before that, we briefly review the notation that will be used in the paper.

Let us assume a supervised learning scenario, where the goal is to learn relevant features from input data using a set of N training data $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$, for $i = 1, \ldots, N$, where $\boldsymbol{x}_i \in \Re^n$ and $\boldsymbol{y}_i \in \Re^m$ are considered as the input and output vectors, respectively. Therefore, n and m denote the dimensions of the input and output spaces. In classification problems, $\boldsymbol{y}_i$ will be used to denote the class membership of the ith pattern, e.g., using 1-of-C encoding [38]. For notational convenience, we define the input and output data matrices: $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ and $\mathbf{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$. It will be assumed throughout the paper that these matrices are centered to remove any correlation between variables produced by a shift of their centers of mass [25]. Sample estimations of the input and output data covariance matrices, as well as of their cross-covariance matrix, can be calculated as $\mathbf{C_{XX}} = \mathbf{XX}^\top$, $\mathbf{C_{YY}} = \mathbf{YY}^\top$ and $\mathbf{C_{XY}} = \mathbf{XY}^\top$, where we have neglected the scaling factor $\frac{1}{N}$, and superscript $^\top$ denotes vector or matrix transposition.

The extracted input data features are calculated as $\mathbf{X}' = \mathbf{U}^\top\mathbf{X}$, where $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n_f}]$ is a projection matrix with projection vectors arranged columnwise[1], and $n_f < n$ is the number of extracted features. The goal of OPLS is to find the projection vectors so that the projected data best approximate the output data in a mean square error (MSE) sense; i.e., OPLS minimizes the following loss function [14],

$$\mathcal{L}(\mathbf{W}, \mathbf{U}) = \|\mathbf{Y} - \mathbf{W}\mathbf{U}^\top\mathbf{X}\|_F^2, \tag{1}$$

where $\mathbf{W}$ is an $m \times n_f$ matrix of regression coefficients that can alternatively be seen as a projection matrix for the output data, $\|\mathbf{A}\|_F = \text{Tr}\{\mathbf{AA}^T\}$ denotes the Frobenius norm of matrix $\mathbf{A}$, and $\text{Tr}\{\cdot\}$ is the trace operator. Note that the above problem is different from standard least squares regression since matrix $\mathbf{U}$ imposes a representation bottleneck [14]. Note also that the solution to (1) is not unique since, e.g., $\mathbf{W}$ can compensate any scaling of matrix $\mathbf{U}$. In the next two subsections we will pay attention to different constraints that can be used to make OPLS solution unique.

2.1. OPLS as a generalized eigenvalue decomposition problem

In this subsection we review the solution to the OPLS problem that is more frequently found in the machine learning literature where OPLS is typically seen as a feature extraction method and the goal is to find a solution for $\mathbf{U}$ (see e.g., [5, 10, 39]).

Developing the Frobenius norm in (1), the objective cost function can be written down as

$$\mathcal{L}(\mathbf{W}, \mathbf{U}) = \text{Tr}\{\mathbf{C_{YY}}\} - 2\,\text{Tr}\{\mathbf{W}^\top\mathbf{C}_{\mathbf{XY}}^\top\mathbf{U}\} + \text{Tr}\{\mathbf{U}^\top\mathbf{C_{XX}}\mathbf{U}\mathbf{W}^\top\mathbf{W}\}. \tag{2}$$

As we have already said, the arguments that minimize this cost function are not unique. However, it can be seen that the optimal $\mathbf{W}$ is uniquely determined for fixed $\mathbf{U}$ as the solution to the LS problem stated in (1)

$$\mathbf{W} = \mathbf{C}_{\mathbf{XY}}^\top\mathbf{U}\left(\mathbf{U}^\top\mathbf{C_{XX}}\mathbf{U}\right)^{-1}. \tag{3}$$

Introducing this expression into (2), and after some algebraic manipulations, the objective cost function can be expressed as a function of $\mathbf{U}$ only:

$$\mathcal{L}(\mathbf{U}) = \text{Tr}\{\mathbf{C_{YY}}\} - \text{Tr}\{\left(\mathbf{U}^\top\mathbf{C_{XX}}\mathbf{U}\right)^{-1}\mathbf{U}^\top\mathbf{C_{XY}}\mathbf{C}_{\mathbf{XY}}^\top\mathbf{U}\}. \tag{4}$$

[1]Note that $\mathbf{U}$ is not a projection operator in a rigorous mathematical sense, since it maps data from ^{n} to ^{n_f}, and therefore does not satisfy the idempotent property of projection operators. However, the columns of $\mathbf{U}$ span the subspace of ^{n} where the data are projected, and it is in this sense that we refer to $\mathbf{U}$ and $\boldsymbol{u}_i$ as projection matrix and vectors respectively, and to $\mathbf{X}$ as projected data. This nomenclature has been widely used in the machine learning field, particularly in works dealing with feature extraction methods.

The minimization of $\mathcal{L}(\mathbf{U})$ is equivalent to the maximization of the second trace in the above expression, i.e., a ratio trace maximization problem (see, e.g., [40, 41]). Obviously, the optimizer of (4) is not unique, since e.g., multiplying $\mathbf{U}$ by a constant does not affect the value of $\mathcal{L}(\mathbf{U})$. The minimizer of (4) can alternatively be found by solving the following constrained optimization problem:

$$\begin{aligned} \max_{\mathbf{U}} \quad & \mathrm{Tr}\{\mathbf{U}^\top \mathbf{C}_{\mathbf{XY}} \mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}\} \\ \text{s.t.:} \quad & \mathbf{U}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{U} = \mathbf{I} \end{aligned} \tag{5}$$

This solution to OPLS can be efficiently obtained by solving the following generalized eigenvalue decomposition problem:

$$\mathbf{C}_{\mathbf{XY}} \mathbf{C}_{\mathbf{XY}}^\top \boldsymbol{u} = \lambda \mathbf{C}_{\mathbf{XX}} \boldsymbol{u} \tag{6}$$

We denote as $\boldsymbol{\Lambda}_{\mathsf{GEV}}$ the diagonal matrix containing the n_f largest generalized eigenvalues of (6) arranged in decreasing order, whereas $\mathbf{U}_{\mathsf{GEV}}$ is a matrix whose columns are the corresponding n_f leading generalized eigenvectors. Note that any matrix $\mathbf{U}_{\mathbf{R}} = \mathbf{U}_{\mathsf{GEV}}\mathbf{R}$, where $\mathbf{R}$ is a rotation matrix, is also a solution to (5). However, $\mathbf{U}_{\mathsf{GEV}}$ has the nice property that any subset containing just the first $n'_f < n_f$ columns of the matrix is also an OPLS solution for the selected number of dimensions. In other words, by using $\mathbf{U}_{\mathsf{GEV}}$ the extracted features are ordered according to their relevance for the regression problem (first feature accounts for the maximum information that can be summarized with a single variable, and so on), whereas this is not true for the rotated matrix $\mathbf{U}_{\mathbf{R}}$.

Once $\mathbf{U}_{\mathsf{GEV}}$ is obtained, it is straightforward to calculate the corresponding regression coefficients using (3)

$$\mathbf{W}_{\mathsf{GEV}} = \mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}_{\mathsf{GEV}}, \tag{7}$$

where we have also used the fact that the columns in $\mathbf{U}_{\mathsf{GEV}}$ are $\mathbf{C}_{\mathbf{XX}}$-orthonormal, i.e., $\mathbf{U}_{\mathsf{GEV}}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{U}_{\mathsf{GEV}} = \mathbf{I}$ (see (5)).

A further interesting property of the above OPLS solution can be obtained by first noting that for $\mathbf{U}_{\mathsf{GEV}}$ it is satisfied

$$\mathbf{C}_{\mathbf{XY}} \mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}_{\mathsf{GEV}} = \mathbf{C}_{\mathbf{XX}} \mathbf{U}_{\mathsf{GEV}} \boldsymbol{\Lambda}_{\mathsf{GEV}} \tag{8}$$

Then, if we premultiply both terms of (8) by $\mathbf{U}_{\mathsf{GEV}}^\top$, and since $\mathbf{W}_{\mathsf{GEV}} = \mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}_{\mathsf{GEV}}$ and $\mathbf{U}_{\mathsf{GEV}}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{U}_{\mathsf{GEV}} = \mathbf{I}$, we arrive at $\mathbf{W}_{\mathsf{GEV}}^\top \mathbf{W}_{\mathsf{GEV}} = \boldsymbol{\Lambda}_{\mathsf{GEV}}$, i.e., the columns of $\mathbf{W}_{\mathsf{GEV}}$ are orthogonal.

2.2. OPLS as an eigenvalue decomposition problem

In the statistics community, the minimization of (1) is usually seen as a reduced-rank regression problem [12] leading to a standard eigenvalue decomposition that provides a solution for the regression matrix $\mathbf{W}$. However, this formulation has not been so often applied in the machine learning literature, where the objective is to extract the most relevant features from the input data (i.e., to find the projection matrix $\mathbf{U}$).

To start with, we will express the optimal $\mathbf{U}$ associated to a given regression matrix in closed form. We take first derivatives of (2) with respect to $\mathbf{U}$

$$\frac{\partial \mathcal{L}(\mathbf{U},\mathbf{W})}{\partial \mathbf{U}} = -2\mathbf{C_{XY}}\mathbf{W} + 2\mathbf{C_{XX}}\mathbf{U}\mathbf{W}^\top\mathbf{W}.$$

Setting the derivatives to zero, and solving out for $\mathbf{U}$, we obtain the following closed-form expression for computing the optimum projection matrix associated to any given $\mathbf{W}$:

$$\mathbf{U} = \mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\mathbf{W}\left(\mathbf{W}^\top\mathbf{W}\right)^{-1} \tag{9}$$

Replacing this expression back in (2), and after some algebraic manipulations, it is possible to express the OPLS cost function in terms of $\mathbf{W}$ only:

$$\mathcal{L}(\mathbf{W}) = \mathrm{Tr}\{\mathbf{C_{YY}}\} - \mathrm{Tr}\{\left(\mathbf{W}^\top\mathbf{W}\right)^{-1}\mathbf{W}^\top\mathbf{C_{XY}^\top}\mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\mathbf{W}\}. \tag{10}$$

The minimization of $\mathcal{L}(\mathbf{W})$ can be carried out by solving the following constrained maximization problem:

$$\begin{aligned} \max_{\mathbf{W}} \quad & \mathrm{Tr}\{\mathbf{W}^\top\mathbf{C_{XY}^\top}\mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\mathbf{W}\} \\ \text{s.t.:} \quad & \mathbf{W}^\top\mathbf{W} = \mathbf{I} \end{aligned} \tag{11}$$

whose solution can be obtained via the standard eigenvalue problem

$$\mathbf{C_{XY}^\top}\mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\boldsymbol{w} = \lambda\boldsymbol{w}. \tag{12}$$

In the following, we denote as $\boldsymbol{\Lambda}_{\mathsf{EVD}}$ the diagonal matrix containing the n_f largest eigenvalues of $\mathbf{C_{XY}^\top}\mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}$ sorted in decreasing order, whereas the columns of $\mathbf{W}_{\mathsf{EVD}}$ contain the corresponding eigenvectors. As before, it should be noted that any rotated version of $\mathbf{W}_{\mathsf{EVD}}$ is also a minimizer of (10), but $\mathbf{W}_{\mathsf{EVD}}$ has the property that any subset containing its $n_f' < n_f$ first columns is the OPLS solution for the selected number of projections.

Using (9), we can obtain the projection vectors associated to the regression matrix $\mathbf{W}_{\mathsf{EVD}}$ as

$$\mathbf{U}_{\mathsf{EVD}} = \mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\mathbf{W}_{\mathsf{EVD}}, \tag{13}$$

where in simplifying we used the fact that $\mathbf{W}_{\mathsf{EVD}}^\top\mathbf{W}_{\mathsf{EVD}} = \mathbf{I}$. As with the classical OPLS solution, it is possible to show that the solution we have derived in this subsection leads also to orthogonal projected data. To see this, let us first explicitly write down the eigenvalue problem satisfied by the regression matrix

$$\mathbf{C_{XY}^\top}\mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\mathbf{W}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{EVD}}\boldsymbol{\Lambda}_{\mathsf{EVD}}. \tag{14}$$

Now, we can premultiply both terms of (14) by $\mathbf{W}_{\mathsf{EVD}}^\top$, which leads to

$$\mathbf{W}_{\mathsf{EVD}}^\top\mathbf{C_{XY}^\top}\mathbf{C_{XX}^{-1}}\mathbf{C_{XY}}\mathbf{W}_{\mathsf{EVD}} = \boldsymbol{\Lambda}_{\mathsf{EVD}} \tag{15}$$

where we have used again orthonormality of the columns of $\mathbf{W}_{\mathsf{EVD}}$ to simplify the right-hand-side term. If we further note that, according to (13), $\mathbf{C}_{\mathbf{XY}}\mathbf{W}_{\mathsf{EVD}} = \mathbf{C}_{\mathbf{XX}}\mathbf{U}_{\mathsf{EVD}}$, we arrive at

$$\mathbf{U}_{\mathsf{EVD}}^{\top}\mathbf{C}_{\mathbf{XX}}\mathbf{U}_{\mathsf{EVD}} = \mathbf{\Lambda}_{\mathsf{EVD}}. \tag{16}$$

which demonstrates the orthogonality condition of the projected input data.

OPLS solution can be obtained in a batch manner (i.e., all projection vectors in $\mathbf{U}_{\mathsf{EVD}}$ are computed at once) by solving the eigenvalue problem (14) followed by (13). We conclude this subsection by presenting an algorithm that sequentially computes projection vectors $\boldsymbol{u}_i$ (i.e., the columns of $\mathbf{U}_{\mathsf{EVD}}$), and that will be the basis for sequential OPLS versions with constraints that we will present in later sections. The sequential algorithm works by iteratively going (for $i = 1, \ldots, n_f$) through the following three steps:

S1) Obtain the leading eigenvector of the symmetric matrix $\mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C}_{\mathbf{XY}}$, to produce the regression coefficients vector $\boldsymbol{w}_i$. Computation of $\boldsymbol{w}_i$ can be efficiently implemented, e.g., using the power method [42].

S2) Obtain the corresponding projection vector $\boldsymbol{u}_i$ as the minimizer of cost function (1) particularized for $n_f = 1$ and $\mathbf{W} = \boldsymbol{w}_i$, i.e.,

$$\begin{aligned} \boldsymbol{u}_i &= \arg\min_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{w}_i, \boldsymbol{u}) = \arg\min_{\boldsymbol{u}} \|\mathbf{Y} - \boldsymbol{w}_i\boldsymbol{u}^{\top}\mathbf{X}\|_F^2 \\ &= \arg\min_{\boldsymbol{u}} \|\boldsymbol{w}_i^{\top}\mathbf{Y} - \boldsymbol{u}^{\top}\mathbf{X}\|_F^2 = \mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C}_{\mathbf{XY}}\boldsymbol{w}_i \end{aligned} \tag{17}$$

S3) Deflate the cross-covariance matrix $\mathbf{C}_{\mathbf{XY}}$ according to

$$\mathbf{C}_{\mathbf{XY}} \leftarrow \mathbf{C}_{\mathbf{XY}} - \mathbf{C}_{\mathbf{XX}}\boldsymbol{u}_i\boldsymbol{w}_i^{\top}. \tag{18}$$

This deflation scheme can be better understood by noticing that it is equivalent to deflating the output data matrix, removing from it the best prediction (in the least-squares sense) that can be achieved using the current projections of the input data, i.e.,

$$\mathbf{Y} \leftarrow \mathbf{Y} - \boldsymbol{w}_i\boldsymbol{u}_i^{\top}\mathbf{X}. \tag{19}$$

2.3. Equivalence between the GEV and EVD solutions to OPLS

It is easy to see that since the solutions to (5) and (11) are different minima of the same cost function, they should both provide the same value of $\mathcal{L}(\mathbf{W}, \mathbf{U})$. In this paper, we derive explicit expressions that show the equivalence between the OPLS solutions obtained using the GEV formulation, $\{\mathbf{U}_{\mathsf{GEV}}, \mathbf{W}_{\mathsf{GEV}}\}$, or recurring to the EVD problem, $\{\mathbf{U}_{\mathsf{EVD}}, \mathbf{W}_{\mathsf{EVD}}\}$. To the best of our knowledge, this connection has not been established before, and is therefore a first contribution of this work.

To simplify the presentation we provide below the existing relationships between the OPLS solutions derived in the previous subsections (the proof is given in Appendix A):

$$\begin{aligned} \mathbf{\Lambda}_{\mathsf{EVD}}, &= \mathbf{\Lambda}_{\mathsf{GEV}} \qquad (= \mathbf{\Lambda}), \\ \mathbf{U}_{\mathsf{EVD}} &= \mathbf{U}_{\mathsf{GEV}}\mathbf{\Lambda}^{1/2}, \\ \mathbf{W}_{\mathsf{EVD}} &= \mathbf{W}_{\mathsf{GEV}}\mathbf{\Lambda}^{-1/2}. \end{aligned} \tag{20}$$

Therefore, since $\mathbf{\Lambda}$ is diagonal, this implies that the columns of $\mathbf{U}_{\mathsf{GEV}}$ and $\mathbf{U}_{\mathsf{EVD}}$ have the same direction, and differ only in a scaling factor.

Table 1 summarizes the main equations and properties of the two alternative solutions to OPLS we have reviewed, to which we will refer in the following as GEV-OPLS and EVD-OPLS.

Table 1: Most relevant equations and properties of the GEV and EVD-OPLS solutions.

	GEV-OPLS (Subsec. 2.1)	**EVD-OPLS (Subsec. 2.2)**
Eigenvalue problem	$\mathbf{C_{XY}}\mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{U}_{\mathsf{GEV}} = \mathbf{C_{XX}}\mathbf{U}_{\mathsf{GEV}}\mathbf{\Lambda}$	$\mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C_{XY}}\mathbf{W}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{EVD}}\mathbf{\Lambda}$
	(dimension n)	(dimension m)
Orthonormality conditions	$\mathbf{U}_{\mathsf{GEV}}^{\top}\mathbf{C_{XX}}\mathbf{U}_{\mathsf{GEV}} = \mathbf{I}$	$\mathbf{U}_{\mathsf{EVD}}^{\top}\mathbf{C_{XX}}\mathbf{U}_{\mathsf{EVD}} = \mathbf{\Lambda}$
	$\mathbf{W}_{\mathsf{GEV}}^{\top}\mathbf{W}_{\mathsf{GEV}} = \mathbf{\Lambda}$	$\mathbf{W}_{\mathsf{EVD}}^{\top}\mathbf{W}_{\mathsf{EVD}} = \mathbf{I}$
$\mathbf{U}$, $\mathbf{W}$ relationship	$\mathbf{W}_{\mathsf{GEV}} = \mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{U}_{\mathsf{GEV}}$	$\mathbf{U}_{\mathsf{EVD}} = \mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C_{XY}}\mathbf{W}_{\mathsf{EVD}}$

Although the GEV-OPLS formulation has been typically used in machine learning papers, it is arguable that the EVD-OPLS formulation offers some important advantages also in this context. In particular, the main advantages of EVD-OPLS that we will exploit in subsequent sections are:

- The dimension of the eigenvalue problems (8) and (14) are n and m, respectively, meaning that EVD-OPLS is computationally more efficient for the common case $m < n$ (i.e., the number of target variables is smaller than the dimensionality of the input data).

- EVD-OPLS facilitates the introduction of constraints on the projection matrix. Indeed, since $\mathbf{U}_{\mathsf{EVD}}$ is given as the solution to a least-squares problem, additional constraints can be easily imposed by modifying (1). For instance, we could favor sparsity on the projection vectors by adding a lasso penalization term. Constraining the GEV-OPLS projection vectors is not as obvious, since $\mathbf{U}_{\mathsf{GEV}}$ is obtained as the solution to the generalized eigenvalue problem (8). Note, that obtaining sparse projection vectors implicitly carries out feature selection on the original data representation, whereas no obvious advantage is derived from sparsity on the model regression coefficient matrix $\mathbf{W}$ (that could be more easily implemented using the GEV-OPLS formulation).

In the next subsection, we compare the computational complexity of the GEV-OPLS and EVD-OPLS formulations. Then, in subsequent sections of the paper, we will rely on the EVD-OPLS formulation to derive sparse solutions both for the linear and non-linear cases.

2.4. Computational complexity

To compare the computational requirements of GEV-OPLS and EVD-OPLS, in this subsection we carry out an empirical comparison of computational complexity of the two solutions. In order to make the comparison fair, we compute first the least-squares solution of the regression problem, $\mathbf{W}_{\mathsf{LS}} = \mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C}_{\mathbf{XY}}$, which has a computational complexity of $\mathcal{O}(n^3)$. In this way, we can rewrite (8) and (14) as the following eigenvalue decomposition problems:

$$\text{GEV-OPLS}: \;\; \mathbf{W}_{\mathsf{LS}}\mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \tag{21}$$

$$\text{EVD-OPLS}: \;\; \mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{W}_{\mathsf{LS}}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda} \tag{22}$$

As we have already discussed, the GEV and EVD formulations require matrices of size $n \times n$ and $m \times m$, respectively, implying eigenvalue decomposition steps with complexity $\mathcal{O}(n^3)$ and $\mathcal{O}(m^3)$ for GEV-OPLS and EVD-OPLS. Note that once the eigenvalue problem of EVD-OPLS is solved, the projection matrix can be straightforwardly computed as $\mathbf{U}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{LS}}\mathbf{W}_{\mathsf{EVD}}$.

To illustrate how computational requirements scale for both methods, we have created an artificial problem according to the following regression model

$$\mathbf{Y} = \sin(\pi\mathbf{M}\mathbf{X} + 1) + \boldsymbol{\Xi},$$

where $\mathbf{X}$ and $\boldsymbol{\Xi}$ are $n \times N$ and $m \times N$ matrices containing the input data and observation noise. The elements of these matrices are independently drawn from Normal distributions with mean zero and standard deviations 0.7 and $5 \cdot 10^{-2}$, respectively for $\mathbf{X}$ and $\boldsymbol{\Xi}$. Finally, $\mathbf{M}$ is an $m \times n$ matrix that contains the parameters of the model, which are independently taken from a uniform distribution between 0 and 1.

Fig. 1 displays the execution times of GEV-OPLS and EVD-OPLS for $N = 5000$ and different values of m and n. All experiments have been carried out on an Intel Core i7 CPU 870, running at 2.93 GHz, and with 8 GB of RAM. As expected, GEV-OPLS computational time increases very fast with n, whereas EVD-OPLS execution time shows just a slight increment, mostly because of the additional time required for calculating $\mathbf{W}_{\mathsf{LS}}$. The opposite behavior is observed as the output dimensionality m increases. These results support our conclusion that EVD-OPLS is a more efficient implementation for the common case in which the input dimensionality exceeds the number of target variables (i.e., $n > m$).

3. Sparse OPLS

In this section, we propose a novel OPLS solution that imposes sparsity on the projection vectors. In this way, the method will not just carry out feature extraction, but also a selection of the most

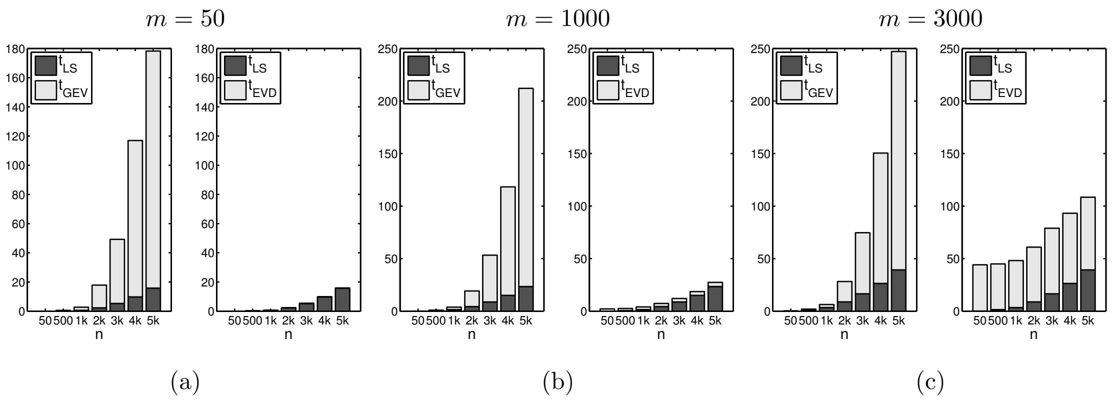


Figure 1: Time in seconds required by the GEV-OPLS (21) and EVD-OPLS (22) implementations. The subplots show the time required for the computation of the least-squares regression model (t_{LS}) and for the solution of the generalized and standard eigenvalue problems (t_{GEV} and t_{EVD}, respectively) for $N = 5000$ and different values of n and m.

relevant variables to solve the problem. This allows more interpretable solutions that involve only a few of the original variables, which is a desirable property of machine learning algorithms in many contexts. To derive our sparse OPLS method (SOPLS) we will rely on the EVD formulation, i.e., we will use constraint $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$ along the derivations.

It is well-known that adding ℓ_1-regularization (known as *lasso*) yields sparse solutions by shrinking to zero the most irrelevant coefficients of the solution. In our approach, we will rely on the elastic net [43] implementation that solves the LS problem subject to both ℓ_1 and ℓ_2-regularization. Thus, we modify the OPLS problem (1) into the minimization of

$$\mathcal{L}_{\mathsf{reg}}(\mathbf{W}, \mathbf{U}) = \|\mathbf{Y} - \mathbf{W}\mathbf{U}^\top\mathbf{X}\|_F^2 + \lambda_1\|\mathbf{U}\|_1 + \lambda_2\|\mathbf{U}\|_F^2, \tag{23}$$

subject to $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$. Here, λ_1 and λ_2 are parameters that control the amount of regularization, and $\|\mathbf{U}\|_1$ is the ℓ_1-norm of matrix $\mathbf{U}$, i.e., the sum of absolute values of all matrix components.

To solve this problem, we rely on an algorithm similar to those in [22, 36], based on the iterative application of the two following steps:

1) $\mathbf{W}-$step: For fixed $\mathbf{U}$, minimize (23) subject to $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$.

   When (23) is minimized with respect to $\mathbf{W}$ only, both regularization terms can be ignored. Therefore, this step reduces to the minimization of the LS cost subject to constraint $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$, thus becoming similar to EVD-OPLS, with the difference that $\mathbf{W}$ is optimized for a generic $\mathbf{U}$, i.e., without assuming (9). In Appendix B we show that the solution of this problem is given by the eigenvalue decomposition

$$\mathbf{C}_{\mathbf{X}\,\mathbf{Y}}^\top\mathbf{C}_{\mathbf{X}\,\mathbf{Y}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}, \tag{24}$$

where $\mathbf{C}_{\mathbf{X}'\mathbf{Y}} = \mathbf{U}^\top \mathbf{C}_{\mathbf{XY}}$. Note that the dimension of the matrix that needs to be analyzed is m, as for the standard EVD-OPLS problem.

2) $\mathbf{U}-$step: For fixed $\mathbf{W}$, minimize (23) with respect to $\mathbf{U}$ only.

   There are several efficient methods to solve this *elastic net* problem. We refer the reader to [44] and [45] for good summaries on optimization methods under ℓ_1-regularization. In the experiments section, we will use the implementation provided by MOSEK 6.0[2], although any other elastic net implementation could also be considered.

Preliminar experiments showed us that initialization of the algorithm is not critical, and we simply initialize $\mathbf{U}$ for the first iteration as the identity matrix. As a stopping mechanism, we use $\text{Tr}\{\mathbf{\Lambda}^{(k)} - \mathbf{\Lambda}^{(k-1)}\} \leq \delta$, where the superscripts denote the iteration index and δ is a small constant. In plain words, the algorithm stops when the difference between the eigenvalues of the $\mathbf{W}-$step of two consecutive iterations is smaller than an arbitrary constant.

It is also worth mentioning that the $\mathbf{U}-$ step can be modified to impose sparsity constraints on entire rows of $\mathbf{U}$ rather than on each isolated component, similarly to what is done in the group *lasso* [46]. The latter approach requires more memory and is computationally more involved. However, it offers the additional advantage that all projection vectors are restricted to use the same variables from the input data representation, and thus favors a real feature selection, since it forces that the same original feature is either removed or retained from all projections.

An important difference exists between our approach and the algorithm in [36]. Given the singular value decomposition $\mathbf{C}_{\mathbf{X}'\mathbf{Y}} = \mathbf{PDQ}^\top$, where $\mathbf{D}$ is a diagonal matrix containing the singular values, and $\mathbf{P}$ and $\mathbf{Q}$ contain the left and right singular vectors, respectively, the output of the $\mathbf{W}-$step of our algorihtm would be $\mathbf{W} = \mathbf{Q}$, whereas the orthogonal Procrustes solution of [36] would output a rotated version $\mathbf{W} = \mathbf{QP}^\top$. This means that in the absence of regularization (i.e., $\lambda_1 = \lambda_2 = 0$), the algorithm in [36] does not in general converge to the OPLS solution, but to a rotated version of the OPLS projection matrix. As we have already discussed, this is not an irrelevant issue because the true OPLS solution guaranties that extracted projections are ordered according to their relevance, i.e., the first $n_f' < n_f$ features contain as much information as possible for that number of variables in the sense of minimizing (23). This property does not hold for rotated solutions. Furthermore, in Appendix C we show that, unlike our solution, the algorithm in [36] strongly depends on initialization, and that in the absence of regularization the algorithm may not progress at all.

[2]`http://www.mosek.com`.

3.1. Sequential implementation of sparse OPLS using deflation

Similar to the sequential implementation of EVD-OPLS, we can derive a sequential algorithm that implements the sparse OPLS feature extraction scheme we have just described. The sequential algorithm works by first extracting the pair $\{\boldsymbol{u}_i, \boldsymbol{w}_i\}$ that minimizes (23) for $n_f = 1$, and then deflating the cross-covariance matrix. These two steps are repeated until the desired number of features is reached. Extraction of the pair $\{\boldsymbol{u}_i, \boldsymbol{w}_i\}$, for $i = 1, \ldots, n_f$, is carried out by iterating the $\mathbf{W}-$ and $\mathbf{U}-$steps we have just described. Note that since at each step we are solving a unidimensional problem, the solution to the $\mathbf{W}-$step can be obtained simply as

$$\boldsymbol{w}_i = \frac{\mathbf{C}_{\mathsf{X}\,\mathbf{Y}}^{\top}}{\|\mathbf{C}_{\mathsf{X}\,\mathbf{Y}}\|}, \tag{25}$$

where $\mathbf{C}_{\mathsf{X}\,\mathbf{Y}} = \boldsymbol{u}_i^{\top}\mathbf{C}_{\mathbf{XY}}$.

Table 2 provides the pseudocode for the sequential sparse algorithm that we have just described. Note that, in the table, subscript i is used to index the projection vectors (i.e., $i = 1, \ldots, n_f$), whereas superscript k indexes the iterative application of $\mathbf{W}-$ and $\mathbf{U}-$steps that are needed to converge to each projection vector. Different convergence criteria can be used for step 2.2.3 of the algorithm. In the experimental section we will monitor the cosine distance

$$d_{\mathsf{cos}}\left(\boldsymbol{u}_i^{(k)}, \boldsymbol{u}_i^{(k-1)}\right) = \frac{\boldsymbol{u}_i^{(k)\top}\boldsymbol{u}_i^{(k-1)}}{\|\boldsymbol{u}_i^{(k)}\|\|\boldsymbol{u}_i^{(k-1)}\|}, \tag{26}$$

and use as a stopping criterion $d_{\mathsf{cos}}\left(\boldsymbol{u}_i^{(k)}, \boldsymbol{u}_i^{(k-1)}\right) > 1 - \delta$, where δ is a tolerance parameter. Other possibilities would consist of monitoring the cosine distance between the regression coefficient vectors, or the eigenvalue of the $\mathbf{W}-$step.

Table 2: Pseudocode for the sequential SOPLS with deflation.

```
1.- Inputs: centered matrices $\mathbf{X}$ and $\mathbf{Y}$, $n_f$, $\lambda_1$, $\lambda_2$
2.- For $i = 1, \ldots, n_f$
    2.1.- Initialize $\boldsymbol{u}_i^{(1)} = \mathbf{1}. * \delta_i$ ‡
    2.2.- For $k = 1, 2, \ldots$
        2.2.1.- Update $\boldsymbol{w}_i^{(k)}$ using (25)
        2.2.2.- Update $\boldsymbol{u}_i^{(k)}$ by solving the *elastic net* problem (23) for $n_f = 1$
        2.2.3.- If convergence criterion is met, output current values as $\{\boldsymbol{u}_i, \boldsymbol{w}_i\}$, otherwise back to 2.2.
    2.3.- Deflate cross-covariance matrix: $\mathbf{C}_{\mathbf{XY}} \leftarrow \mathbf{C}_{\mathbf{XY}} - \mathbf{C}_{\mathbf{XY}}\boldsymbol{u}_i\boldsymbol{w}_i^{\top}$
3.- Outputs: $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n_f}]$, $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n_f}]$
```

‡ Projection vector $\boldsymbol{u}_i$ is initialized as a vector with its ith component equal to 1, and all other components equal to 0.

3.2. Sparse OPLS with group lasso penalty

To conclude the section, we illustrate how the sparse OPLS method we have just proposed can be easily extended to impose sparsity in the projection matrix in a row-wise manner. In this way, the algorithm performs a true feature selection since when all elements of a row are set to zero, the corresponding feature will not be used for any of the projection vectors.

In order to do so, we will incorporate a group lasso penalty to (1), leading to

$$\mathcal{L}_{\text{reg}}(\mathbf{W}, \mathbf{U}) = \|\mathbf{Y} - \mathbf{W}\mathbf{U}^\top\mathbf{X}\|_F^2 + \sum_{i=1}^{n} \lambda_i \|\boldsymbol{u}^i\|_2, \tag{27}$$

being $\boldsymbol{u}^i$ the ith row of the matrix projection $\mathbf{U}$ and λ_i the penalization term associated to each input variable. It can be easily seen that the optimization of this new regularized cost can also be carried out iterating updates for W and U. In fact, only the $\mathbf{U}-$step needs to be modified to account for the new cost function (27). In [37] the solution of (27) is carried out relying on the Procrustes solution to solve the $\mathbf{W}-$step; thus, the proposed solution suffers from the same drawbacks than the approach in [36]. These problems, as well as the advantages of our formulation, will be analyzed in the experimental section.

4. Sparse Kernel OPLS extensions

Since relationships among variables are usually non-linear, in this section we will also pay attention to kernel extensions of the OPLS algorithm [5, 15], and show how this method can also benefit from an eigenvalue decomposition formulation similar to the one proposed for EVD-OPLS. After describing the straightforward kernel extension of the OPLS method (KOPLS), we will focus on a reduced KOPLS (rKOPLS) approach proposed in [15], which forces sparsity of the solution *a priori*, and solves some practical problems inherent to the standard KOPLS method. As for the linear case, our EVD-rKOPLS formulation enjoys two main advantages: improved efficiency with respect to CPU cost, and possibility to impose additional constraints on the projection vectors. We exploit this second property to derive a sparse formulation of rKOPLS at the end of the section.

Along this section, we consider that input data $\mathbf{X}$ are mapped into some Reproducing Kernel Hilbert Space (RKHS) through a mapping function $\boldsymbol{\phi}(\boldsymbol{x}) : \mathrm{R}^n \to \mathcal{F}$, where the target space is normally very high- or even infinite-dimensional. Training data are stacked together in matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N)]$, so that now the n_f projections of the input data are given by $\boldsymbol{\Phi}' = \mathbf{U}^T\tilde{\boldsymbol{\Phi}}$, where $\tilde{\boldsymbol{\Phi}}$ is the centered version of $\boldsymbol{\Phi}$ and $\mathbf{U}$ is the projection matrix of size $\dim(\mathcal{F}) \times n_f$. Thereby, the OPLS cost function (1) can be rewritten in the feature space as,

$$\mathcal{L}_{\mathcal{F}}(\mathbf{W}, \mathbf{U}) = ||\mathbf{Y} - \mathbf{W}\mathbf{U}^T\tilde{\boldsymbol{\Phi}}||_F^2. \tag{28}$$

In order to solve the above problem for the usual case in which dimension of $\mathcal{F}$ is infinite, we will use the Representer's Theorem [25], that states that projection vectors can be expressed as a linear combination of the mapped input data, $\mathbf{U} = \tilde{\mathbf{\Phi}}\mathbf{A}$, with $\mathbf{A} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_{n_f}]$ and $\boldsymbol{\alpha}_i$ parameterizes the ith projection vector. Introducing this expression in (28), we get

$$\mathcal{L}_{\mathcal{F}}(\mathbf{W}, \mathbf{A}) = ||\mathbf{Y} - \mathbf{W}\mathbf{A}^T\mathbf{K}_x||_F^2, \tag{29}$$

where $\mathbf{K}_x = \tilde{\mathbf{\Phi}}^T\tilde{\mathbf{\Phi}}$ is the centered kernel matrix, that involves only inner products in $\mathcal{F}$. Different kernel functions to build up the kernel matrices and the centering process of these matrices are explained in detail in [25].

We can see that (29) is formally equivalent to (1). Thus, KOPLS formulations based on generalized and standard eigenvalue decompositions (GEV- and EVD-KOPLS, respectively) can be easily obtained by replacing $\mathbf{U}$ by $\mathbf{A}$ and $\mathbf{X}$ by $\mathbf{K}_x$ in the linear formulations. We remark that the computational savings of the EVD formulation can be even more important in this case, since the size the GEV-KOPLS matrix decomposition problem increases with N, while EVD-KOPLS still involves the decomposition of an $m \times m$ matrix.

KOPLS requires the inversion of matrix $\mathbf{K}_x\mathbf{K}_x$, which is usually ill-conditioned, so that some sort or regularization is needed. Furthermore, when dealing with large data sets, computational and memory requirements to handle kernel matrices usually make unfeasible to work with this method. For these reasons, in the next subsection we turn our attention to the reduced complexity KOPLS method (rKOPLS) of [15] that is able to overcome these limitations.

4.1. Reduced KOPLS as an eigenvalue decomposition problem

The rKOPLS formulation, which we borrow from [15], is given by $\mathbf{U} = \tilde{\mathbf{\Phi}}_R\mathbf{B}$, where $\mathbf{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{n_f}]$ is the coefficient matrix of the reduced model and $\tilde{\mathbf{\Phi}}_R$ is a matrix containing a subset of R training data ($R < N$) selected randomly[3]. Introducing the new expression for $\mathbf{U}$ into (29), we get the following objective function:

$$\mathcal{L}_{\mathcal{F}}(\mathbf{W}, \mathbf{B}) = ||\mathbf{Y} - \mathbf{W}\mathbf{B}^\top\mathbf{K}_R||_F^2, \tag{30}$$

where $\mathbf{K}_R = \tilde{\mathbf{\Phi}}_R^\top\tilde{\mathbf{\Phi}}$ is a kernel matrix of size $R \times N$. In other words, while KOPLS projection vectors are obtained as a linear combination of all training data ($\mathbf{U} = \tilde{\mathbf{\Phi}}\mathbf{A}$), rKOPLS *a priori* enforces sparsity by expressing the projection vectors as linear combinations of a reduced set of the training data. We should emphasize the differences between the 'sparsity' concept in the linear and kernel algorithms: whereas for the linear case sparsity is induced over the original variables of the data, in KOPLS we

[3]Here we recur to the random selection strategy that was used in [15], but more sophisticated strategies, such as Nyström subsampling, could be used [26] as well, both for rKOPLS and for the sparse version that we derive in the next subsection. A more careful selection of the subset $\tilde{\mathbf{\Phi}}_R$ usually results in improved accuracy for a fixed value of R at the cost of a more expensive training phase.

refer to the capability of the methods to express the solution in terms of a reduced set of training data, and implies mostly a computational benefit (both during the train and test phases). It is also important to note that since kernel matrix $\mathbf{K}_R$ still involves all available training data, rKOPLS results in a more powerful approximation than mere subsampling.

A solution to (30) based on a standard generalized eigenvalue decomposition (GEV-rKOPLS) was given in [15]. Alternatively, in this paper we propose to reformulate the problem as a standard eigenvalue decomposition problem. Again, derivation of the EVD-rKOPLS solution is straightforward given the similarities between (1) and (30): we just need to replace $\mathbf{U}$, $\boldsymbol{u}_i$, $\mathbf{C}_{\mathbf{XX}}$ and $\mathbf{C}_{\mathbf{XY}}$ respectively by $\mathbf{B}$, $\boldsymbol{\beta}_i$, $\mathbf{K}_R\mathbf{K}_R^\top$ and $\mathbf{K}_R\mathbf{Y}^\top$. Then, a batch EVD-rKOPLS algorithm can be obtained by going through the following three steps:

1. $\mathbf{W}_{\mathsf{LS}} = \left(\mathbf{K}_R\mathbf{K}_R^\top\right)^{-1}\mathbf{K}_R\mathbf{Y}$
2. $\mathbf{Y}\mathbf{K}_R^\top\mathbf{W}_{\mathsf{LS}}\mathbf{W}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{EVD}}\boldsymbol{\Lambda}_{\mathsf{EVD}}$
3. $\mathbf{B}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{LS}}\mathbf{W}_{\mathsf{EVD}}$

Parameter R acts as a sort of regularizer, making $\mathbf{K}_R\mathbf{K}_R^\top$ full rank. It also dictates the computational and memory requirements of the algorithm. In case of $R = N$, the KOPLS solution would be recovered. Table 3 summarizes the main characteristics of KOPLS (see [5]), GEV-rKOPLS and EVD-rKOPLS in terms of computational and memory requirements. Note that the proposed EVD-rKOPLS approach is normally more efficient than the other two solutions in time and storage terms.

Table 3: Computational time and memory requirements comparison.

	GEV-KOPLS	GEV-rKOPLS	EVD-rKOPLS
Kernel matrix dimensions	$N \times N$	$R \times N$	$R \times N$
Memory requirement	$\mathcal{O}(N^2)$	$\mathcal{O}(R^2)$	$\mathcal{O}(R^2)$
GEV/EVD problem complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(R^3)$	$\mathcal{O}(m^3)$

4.2. Sparse rKOPLS

Standard KOPLS solution is usually given by a dense projection matrix $\mathbf{A}$. Thus, to extract features for new data, it is necessary to calculate the kernels between these new data and all the training patterns. The rKOPLS algorithm alleviates this problem by imposing sparsity *a priori* on the number of kernels to be computed, a fact which implies computational and memory savings; but it randomly selects the vectors in $\mathcal{F}$ that span the solution. Therefore, rKOPLS does not guarantee the selection of the most representative training data for the expansion, neither that the most sparse representation is achieved.

Trying to provide a solution to this problem, in this subsection we add an ℓ_1-regularization term in the rKOPLS objective function to further induce sparsity of the solution in terms of the $\boldsymbol{\beta}_i$ vectors. In this way, the method automatically selects the most representative patterns in $\tilde{\boldsymbol{\Phi}}_R$ and reduces the number of kernels that need to be computed for projecting new data.

The novel sparse rKOPLS approach, to which we will refer as SrKOPLS, is given by the minimization of

$$\mathcal{L}_{\mathcal{F}} = ||\mathbf{Y} - \mathbf{W}\mathbf{B}^\top\mathbf{K}_R||_F^2 + \lambda_1\|\mathbf{B}\|_1. \tag{31}$$

Imposing sparsity on matrix $\mathbf{B}$ has beneficial effects with respect to generalization, as we will see in the experiments section. Furthermore, more compact solutions can be expected, i.e., SrKOPLS solution will reduce the number of kernels needed for feature extraction.

To minimize (31), we need to recur to an EVD formulation that imposes the usual constraint $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$, so that minimization with respect to $\mathbf{B}$ can be done without constraints. Then, we can use again the algorithms in Section 3, just replacing matrices $\mathbf{U}$, $\boldsymbol{u}_i$, $\mathbf{C}_{\mathbf{XX}}$ and $\mathbf{C}_{\mathbf{XY}}$, respectively by $\mathbf{B}$, $\boldsymbol{\beta}_i$, $\mathbf{K}_R\mathbf{K}_R^\top$ and $\mathbf{K}_R\mathbf{Y}^\top$.

A batch formulation of the SrKOPLS algorithm would consist in the iterative application of the two following steps:

1) $\mathbf{W}-$step: For fixed $\mathbf{B}$, find $\mathbf{W}$ as the solution to the following eigenvalue decomposition problem $\mathbf{Y}\mathbf{K}_R^\top\,\mathbf{K}_R\,\mathbf{Y}^\top\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda}$, where $\mathbf{K}_R\; = \mathbf{B}^\top\mathbf{K}_R$.
2) $\mathbf{B}-$step: For fixed $\mathbf{W}$, solve the *elastic net* problem to minimize (31) with respect to $\mathbf{B}$ only.

For this batch formulation, the same initialization and stopping mechanism as for the linear solution can be used.

If a sequential implementation of SrKOPLS is preferred, at each step a unidimensional problem is solved followed by deflation of matrix $\mathbf{K}_R\mathbf{Y}^\top$. In this case, the solution to the $\mathbf{W}-$step can be simply computed as

$$\boldsymbol{w}_i = \frac{\mathbf{Y}\boldsymbol{k}_R^\top}{\|\mathbf{Y}\boldsymbol{k}_R^\top\|}, \tag{32}$$

where $\boldsymbol{k}_R\; = \boldsymbol{\beta}_i^\top\mathbf{K}_R$. As for the stopping criterion, we use also the same criterion that was applied for the linear SOPLS algorithm:

$$d_{\cos}\left(\boldsymbol{u}_i^{(k)}, \boldsymbol{u}_i^{(k-1)}\right) = \left|\frac{\boldsymbol{\beta}_i^{(k)\top}\mathbf{K}_{RR}\boldsymbol{\beta}_i^{(k-1)}}{\sqrt{\boldsymbol{\beta}_i^{(k)\top}\mathbf{K}_{RR}\boldsymbol{\beta}_i^{(k)}}\sqrt{\boldsymbol{\beta}_i^{(k-1)\top}\mathbf{K}_{RR}\boldsymbol{\beta}_i^{(k-1)}}}\right|, \tag{33}$$

requiring $d_{\cos}\left(\boldsymbol{u}_i^{(k)}, \boldsymbol{u}_i^{(k-1)}\right) > 1-\delta$, where δ is a tolerance parameter. Table 4 provides the pseudocode for the sequential implementation that we have just described.

Table 4: Pseudocode for the sequential SrKOPLS with deflation.

```
1.- Inputs: centered matrices $\mathbf{K}_R$ and $\mathbf{Y}$, $n_f$, $\lambda_1$
2.- For $i = 1, \ldots, n_f$
    2.1.- Initialize $\boldsymbol{\beta}_i^{(1)} = \mathbf{1}. * \delta_i$ ‡
    2.2.- For $k = 1, 2, \ldots$
        2.2.1.- Update $\boldsymbol{w}_i^{(k)}$ using (32)
        2.2.2.- Update $\boldsymbol{\beta}_i^{(k)}$ to solve the *elastic net* problem (31)
        2.2.3.- If convergence criterion is met, output current values as $\{\boldsymbol{\beta}_i, \boldsymbol{w}_i\}$, otherwise back to 2.2.
    2.3.- Deflate cross-covariance matrix: $\mathbf{Y}\mathbf{K}_R^\top \leftarrow \mathbf{Y}\mathbf{K}_R^\top - \boldsymbol{w}_i\boldsymbol{\beta}_i^\top\mathbf{K}_R\mathbf{K}_R^\top$
3.- Outputs: $\mathbf{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{n_f}]$, $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n_f}]$
```

‡ Projection vector $\boldsymbol{\beta}_i$ is initialized as a vector with its ith component equal to 1, and all other components equal to 0.

5. Experiments

In this section, we analyze the discriminative power of both sparse linear (SOPLS) and sparse non-linear (SKOPLS and SrKOPLS) solutions. For this purpose, we are going to evaluate the performance of these approaches over nine multi-class classification problems from the UCI machine learning repository[4]. Table 5 summarizes their main characteristics, being N_{train} and N_{test} the number of samples in both train and test datasets, respectively. To complete this study, we will also analyze the convergence of the proposed SOPLS solution to that of the OPLS when the sparsity constraint is removed. Finally, we also show the advantages of sparse solutions in a face recognition task.

Table 5: Main properties of the selected benchmark problems.

	N_{train}/N_{test}	n	m
arrhythmia	315 / 135	276	16
letter	10000 / 10000	16	26
mfeatures	1400 / 600	649	10
optdigits	3823 / 1797	64	10
pendigits	7494 / 3498	16	10
satellite	4435 / 2000	36	6
segment	1310 / 1000	18	7
vehicle	500 / 346	18	4
yeast	1038 / 446	8	10

[4] `http://archive.ics.uci.edu/ml`

Table 6: Overall Accuracy (OA) achieved by OPLS, P-SOPLS and SOPLS algorithms. Sparsity rates (SR) of P-SOPLS and SOPLS also are included.

	OPLS	P-SOPLS		SOPLS	
	OA(%)	OA(%)	SR(%)	OA(%)	SR(%)
arrhythmia	50,37	69,63	77.63	69,63	76.06
letter	84,89	84,85	11,33	**85,05**	10,94
mfeatures	97,83	98,33	38.64	98,33	31.55
optdigits	94,21	94,27	42,47	**95,05**	29,93
pendigits	92,08	91,68	39,58	**92,22**	43,06
satellite	85,7	85,90	17,22	**86,10**	27,22
segment	92,8	**95,60**	90,74	94,90	93,52
vehicle	78,32	77,17	25,93	**78,03**	1,85
yeast	58.52	**58.74**	35.94	58.27	23.44

5.1. Sparse Linear Feature Extraction

This subsection analyzes the capability of the proposed SOPLS approach against the standard OPLS method and the sparse OPLS algorithm proposed in [2], which is based on the Procrustes problem solution; for this reason, we will denote it as P-SOPLS (Procrustes Sparse OPLS).

To compute the solutions of the different approaches under study, OPLS method follows the steps described in Eqs. (13) and (14), P-SOPLS follows the procedure described in [36], and the proposed SOPLS approach uses the formulation detailed in Table 2, stopping its iterative process when either the cosine distance (26) achieves a tolerance level of $\delta = 10^{-12}$ or 500 iterations have been completed. To test the discrimination capability of the set of features provided for each feature extraction approach, a linear C-SVM has been trained using as inputs the maximum number of projections ($r = \text{rank}\{\mathbf{C_{XY}}\}$). The regularization parameter λ_1 of SOPLS and P-SOPLS approaches and the empirical cost parameter C of the SVM have both been adjusted by a 10-fold Cross Validation (CV) process over the training data. We have explored a rectangular grid taking 40 values logarithmically spaced between 10^{-4} and 10^{-1} for λ_1 and selecting C from the set of values $\{1, 10, 100, 1000\}$. We have checked that these intervals are sufficiently large to assure that the limits were not selected as a result of the CV.

It is important to note that problem *segment* is ill-conditioned ($\text{rank}\{\mathbf{C_{XX}}\} < n$) preventing the application of OPLS; for this reason, PCA has been applied as preprocessing step to reduce the input data dimension to $\text{rank}\{\mathbf{C_{XX}}\}$, after which OPLS algorithm can be applied. This was not necessary for the sparse approaches (P-SOPLS and SOPLS) since the included ℓ_1-regularizer makes possible to solve ill-conditioned problems without any preprocessing step.

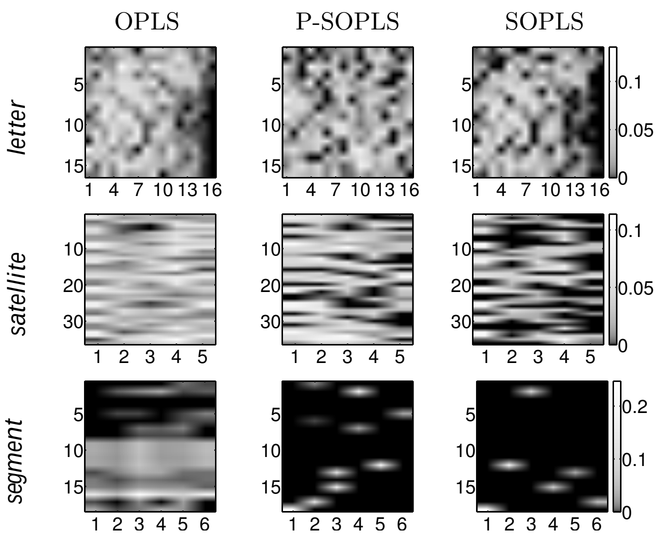


Figure 2: Representation of the projection matrix U $(n \times n_f)$ in OPLS, P-SOPLS, and SOPLS for three representative problems.

Table 6 shows the overall accuracy (OA) provided by these three feature selection techniques and the sparsity rate (SR) of the projections vectors, defined as the ratio between the number of zero coefficients and the total number of coefficients. When SOPLS features are used to train the C-SVM, OPLS is outperformed in all the datasets, whereas it improves or ties the P-SOPLS method in terms of OA.

Apart from its increased discrimination capability, the main advantage of the proposed SOPLS method relies on its sparse formulation that makes it easier to analyze which features do not contribute to the new projected ones. To carry out this analysis, Figure 2 depicts the projection matrices $\mathbf{U}$ obtained by OPLS, SOPLS, and P-SOPLS solutions in three representative problems. Looking at these figures, one can see that in problems presenting a high SR, such as *segment*, the feature extraction becomes close to feature selection, since most features are associated with just one of the original variables. In *satellite*, features 8, 31, 32 and 36 are removed from the first projection vectors (the most important ones) of the SOPLS algorithm.

Regarding the computational burden, implementation of the SOPLS and P-SOPLS methods involve basically the same operations, and differences in time should be mostly due to initialization and stopping criteria. In practice, we have observed that both methods require quite similar training times in all problems, whereas OPLS is obviously a more efficient solution (it solves just one EVD problem).

5.2. Convergence of sparse methods to the OPLS solution for $\lambda_1 = 0$

In this section, we compare the convergence of both SOPLS and P-SOPLS solutions to the standard OPLS if the sparsity constraint tends to zero ($\lambda_1 \to 0$). To carry out this analysis, we analyze the orthogonality of the projected data for block implementations of both SOPLS and P-SOPLS algorithms, being also a property which ease the selection of subsets of variables.

Figure 3 depicts the Frobenius distance between the covariance matrix of the projected data (when either SOPLS or P-SOPLS algorithms are used) and matrix $\mathbf{\Lambda}$ (the covariance of the projected data when using the OPLS algorithm).

As we expected, when λ_1 is close to zero, the projection matrix provided by the SOPLS method is orthogonal, tending its solution to that of OPLS; when λ_1 increases, the SOPLS solution drops most of their coefficients to zero, making SOPLS and OPLS solutions different. However, P-SOPLS algorithm does not present this desired behavior (as it is proved in Appendix C). Despite the addition of the ℓ_1-penalty reduces the orthogonality of the solution, if we pay attention to λ_1 values selected by the CV process (marked with an asterisk in Figure 3 curves), we can observe that the proposed SOPLS algorithm tends to select working points with more orthogonal solutions than those of P-SOPLS.

The advantage of these orthogonal features can be clearly seen in Fig. 4, where the overall accuracy against the number of used projections ($1 \leq n_f \leq r$) is displayed for the three methods under study: OPLS, SOPLS, and P-SOPLS. As we expected, the proposed SOPLS approach outperforms the P-SOPLS results when a bottleneck ($n_f < r$) is applied, showing significant advantages in eight out of the nine problems. This increased performance is due to the fact that the projections obtained by SOPLS are more orthogonal than those of P-SOPLS, as we discussed in Fig. 3.

5.3. OPLS with group-lasso penalty

This subsection analyzes the performance of the method proposed in Subsection 3.1 that incorporates a group-lasso penalty in the objective function (G-OPLS method). The method will be compared to the SRRR method from [37] that relies on the Procrustes solution and the standard OPLS, both from the point of view of discriminative power of the selected projections and with respect to the number of features that are selected by the method. Unlike in the previous subsections, these methods guarantee that unnecessary features are simultaneously removed from all projection vectors, thus carrying out a more practical feature selection.

Experimental settings for the G-OPLS and SRRR methods are as follows: a common penalty term is used for all features ($\lambda_1 = \cdots = \lambda_n$), and its value has been adjusted by a 10-fold CV process exploring 40 logarithmically spaced between 10^{-4} and 10^{-1}. As before, a linear C-SVM was used to test the discrimination power of any set of features.

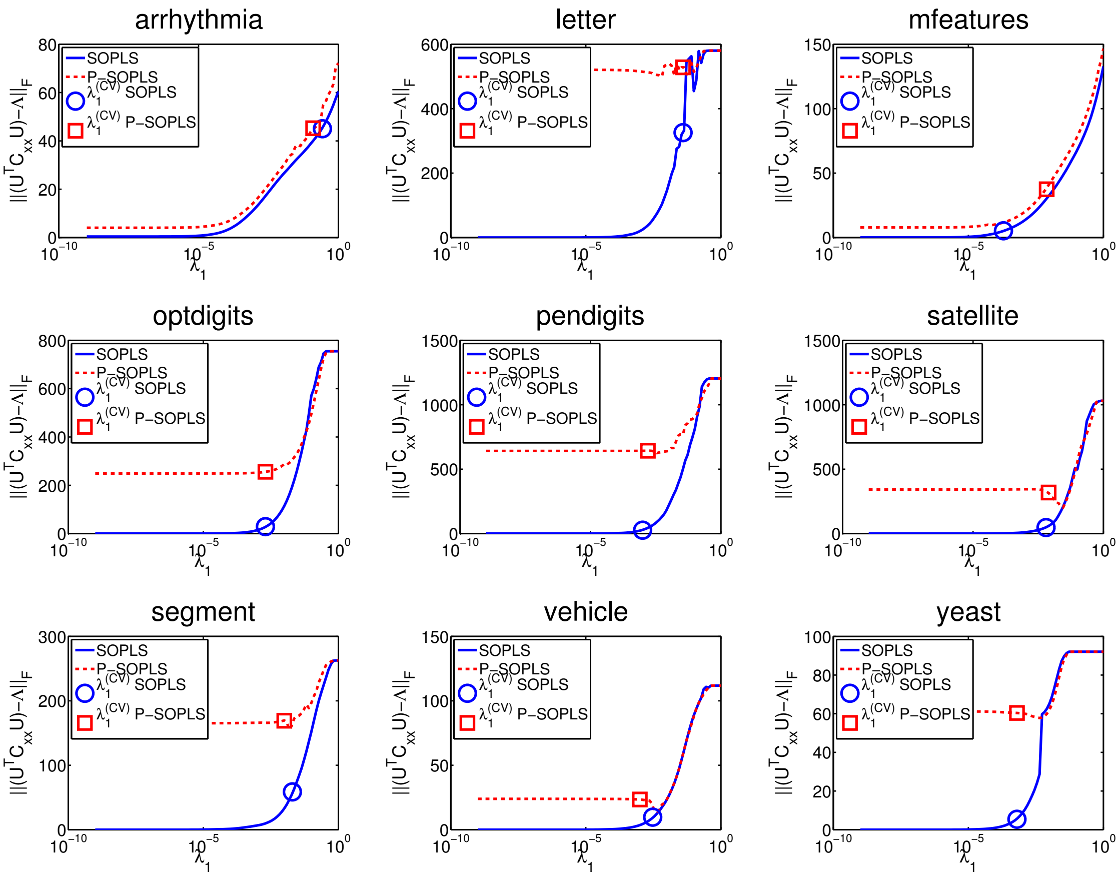


Figure 3: Frobenius distance between the covariance matrix of the projected data when either SOPLS or P-OPLS algorithm are used and matrix $\mathbf{\Lambda}$ (the covariance of the projected data when using the OPLS algorithm). Markers show the ℓ_1-norm penalty parameter (λ_1) selected by CV for both algorithms.

Fig. 5 displays the overall accuracy against the number of used projections for the three methods under study. It can be easily seen that G-OPLS outperforms SRRR when a bottleneck ($n_f < r$) is applied, showing significant advantages in most of the problems under consideration. This gain is a direct consequence of our solution imposing orthogonality of the projected data. The removed features rate (%) of the G-OPLS and SRRR methods is indicated in brackets in the legends of Fig. 5. In this case, both methods achieve exactly the same solution for $n_f = r$, and therefore the number of selected features is the same for both methods. For some of the problems, these methods were able to remove a good amount of the original features without any performance degradation, thus allowing us

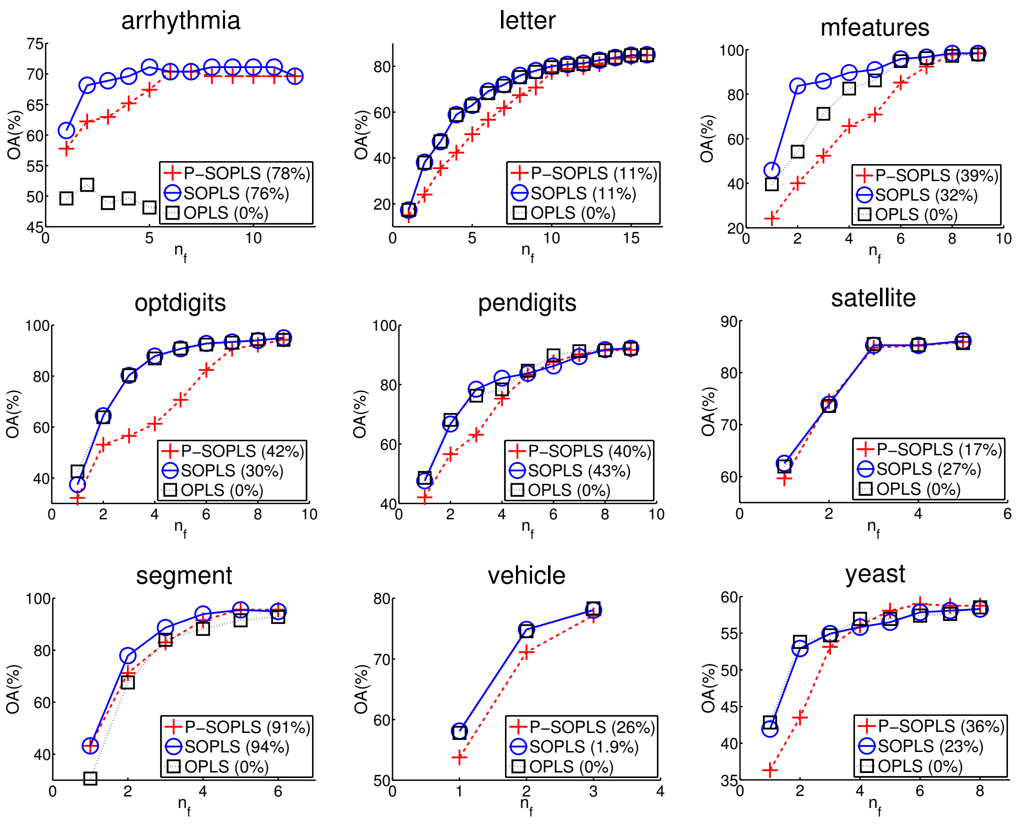


Figure 4: Overall Accuracy (OA) (%) provided by OPLS, SOPLS, and P-SOPLS algorithms for different number of features n_f. Sparsity rates (SR) achieved when all projections ($n_f = r$) are used is shown in the legend.

to identify irrelevant features that can be safely ignored.

5.4. Feature Extraction for Kernel Framework

This subsection studies the performance provided by non-linear OPLS extensions using kernel methods and the EVD formulation. To avoid computational problems of these formulations, their reduced versions (rKOPLS and SrKOPLS) will be also included in this analysis to be able to study the performance of these methods when they are dealing with large datasets. For this reason, we will test SKOPLS and KOPLS performances over seven medium and low size problems: *arrhythmia*, *mfeatures*, *optdigits*, *satellite*, *segment*, *vehicle*, and *yeast*; their reduced formulations (rKOPLS and SrKOPLS) will be analyzed over the same problems than the linear versions, except for problem *arrhythmia* where its reduced number of training samples prevents the application of the subsampling process.

For all methods under study, a Gaussian kernel with dispersion parameter σ,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{2\sigma^2}\right),$$

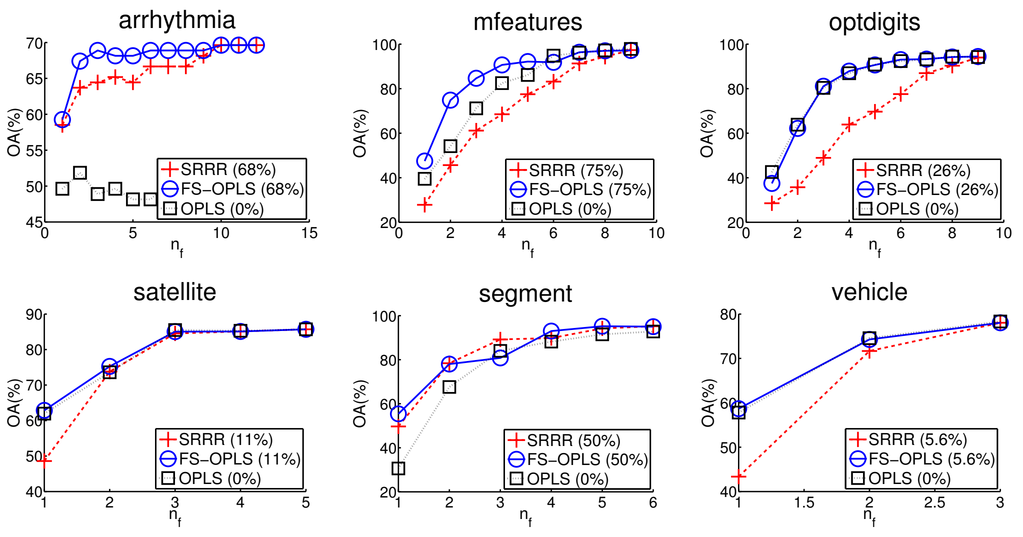


Figure 5: Overall Accuracy (OA) (%) provided by OPLS, G-OPLS, and SRRR algorithms for different number of features n_f. Removed features rate (%) achieved when all projections ($n_f = r$) are used is shown in the legend.

has been used. Once the new features of each method have been extracted, a linear C-SVM has been trained to measure the discriminative capability of each subset of projected features.

As in the previous subsection, free parameters have been adjusted by a 10-fold CV process, selecting parameter C of the SVM from the set of values $\{1, 10, 100, 1000\}$ and sweeping σ in the set $\{0.5, 1, 1.5, 2\} \times \sigma_0$, being σ_0 the median distance among all input data. In sparse methods (SKOPLS and SrKOPLS), regularization parameter λ_1 is cross-validated in the set of values $\{10^{-7}, 10^{-6}, 10^{-5}\}$ and the iterative process has fixed a stop criterion of $\delta = 10^{-12}$ with a maximum of 500 iterations.

Due to kernel matrices are usually ill-conditioned ($\text{rank}\{\mathbf{K}_x\} < N$), KOPLS needs to include an ℓ_2-regularization term to compute its solution. For this reason, an ℓ_2-penalty has been included in both KOPLS and SKOPLS methods, where the regularization parameter is selected by CV among the set of values $\{10^{-9}, 10^{-9}, \ldots, 10^{-1}\}$ and $\{10^{-12}, 10^{-9}, 10^{-7}, 10^{-5}\}$ for KOPLS and SKOPLS methods respectively.

Table 7 compares the performance of the proposed SKOPLS algorithm with those of KOPLS. It can be seen that SKOPLS presents improved or, at least, similar performance than the KOPLS method in all the problems, but *optdigits*. Moreover, due to sparse formulation of SKOPLS is intended to remove data from the projection vectors, this method has the additional advantage of reducing the complexity of the solution, providing SRs around 40% in *optdigits* and *satellite* or, even, 80% in *mfetures*, *segment* and *yeast*. These high sparsity rates can be translated into significant computational load reductions since in problems such as *yeast* only the 20% of the kernels has to be computed to obtain the projected data.

Table 7: Comparison between KOPLS and SKOPLS algorithms in terms of overall Accuracy (OA). In SKOPLS algorithm, the sparsity rate (SR) and the number of useful samples (N_u) to total training samples (N) rate are also displayed.

	KOPLS	SKOPLS		
	OA(%)	OA(%)	SR(%)	N_u/N (rate %)
arrhythmia	71.85	**73.33**	27.88	315/315 (100%)
mfeatures	96.33	**96.67**	86.03	878/1400 (62.71%)
optdigits	**98.33**	98.16	42.52	3809/3823 (99.63%)
satellite	**91.45**	**91.45**	44.86	4114/4435 (92.76%)
segment	**95.5**	**95.5**	75.78	847/1310 (64.65%)
vehicle	82.08	**83.53**	65	362/500 (72.4%)
yeast	58.3	**60.54**	94.31	244/1038 (23.51%)

Table 8: Overall Accuracy of rKOPLS and SrKOPLS algorithms for different training data subset sizes (R = 250, 500 and 1000).

		250		500		1000	
		rKOPLS	SrKOPLS	rKOPLS	SrKOPLS	rKOPLS	SrKOPLS
letter	OA	90.9	**91.44**	93.14	**93.38**	94.52	**94.55**
	SR	–	6.79%	–	9.35%	–	3.27%
mfeatures	OA	**98.31**	98.05	97.97	**98.53**	–	–
	SR	–	18.89%	–	13.92%	–	–
optdigits	OA	**97.45**	97.40	97.77	**98.01**	98.15	**98.17**
	SR	–	6.74%	–	18.13%	–	37.82%
pendigits	OA	97.76	**97.81**	98.17	**98.22**	98.14	**98.16**
	SR	–	10.90%	–	19.74%	–	10.73%
satellite	OA	**89.91**	89.78	**90.59**	90.42	91	**91.22**
	SR	–	18.24%	–	10.64%	–	24.91%
segment	OA	95.98	**96.11**	95.58	**95.75**	–	–
	SR	–	29.75%	–	50.77%	–	–
vehicle	OA	80.58	**81.96**	80.26	**81.56**	–	–
	SR	–	57.41%	–	76.39%	–	–
yeast	OA	56.93	**60.04**	56.77	**60.11**	–	–
	SR	–	44.20%	–	44.93%	–	–

Table 8 compares the efficient solutions of KOPLS (rKOPLS) and SKOPLS (SrKOPLS); due to the solution of these approaches depends on the subsampling process, Table 8 includes the overall accuracy (OA) resulting from averaging 10 independent runs. This efficient technique allows us to fix *a priori* sparsity rate or a initial randomly selected subset of training data (R). For this experiment, we display the solutions obtained with $R = 250$, $R = 500$ and $R = 1000$; for instance, in *vehicle* dataset, *a priori* sparsity rate of 50% is fixed for $R = 250$.

Results show, for any value of R, that SrKOPLS approach tends to outperform rKOPLS in almost all the problems, leading us to the conclusion that SrKOPLS projections are more discriminative than those of the rKOPLS. Even in the case where the most aggressive subsampling is applied ($R = 250$), SrKOPLS improves rKOPLS accuracy in five out of eight problems and it is able to reduce, even more, the solution complexity; note that it provides SR around 30% in *segment* and close to 60% in *vehicle*.

We should emphasize that solving the KOPLS problem is not practical when dealing with large datasets of just a few thousands of patterns. As discussed for the linear sparse algorithms, our solution and the one based on Procrustes both required very similar CPU time, and the differences are mostly due to initialization and stopping criteria. Both solutions are considerably more computationally demanding that the rKOPLS method, since the latter does not need to iterate over several EVD problems.

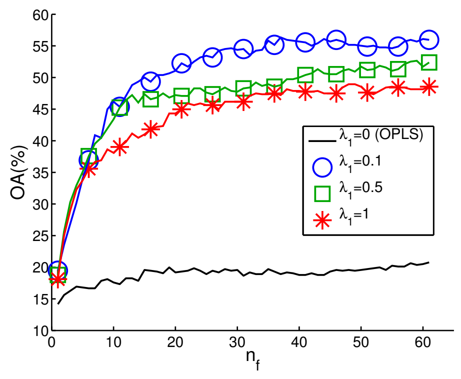
5.5. Sparse Feature Extraction for Face Recognition

In order to show the advantages of SOPLS over OPLS in a real problem, in this section we analyze the performance of these algorithms over a database of face images. In particular, this dataset is a preprocessed excerpt of "Labeled Faces in the Wild" (LFW)[5]. The complete dataset contains more than 13,000 images of faces from 1680 people. However, in order to work with a well-defined dataset, we have selected just the people with at least 20 available images. This results in a reduced dataset with 62 people, consisting of 2276 training images and 756 for testing purposes. The image size is 50×37 pixels, which are rearranged as a row vector of 1850 variables.

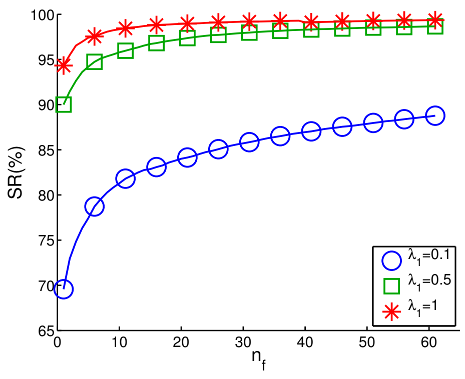
To study the advantages of the sparsity induced by the SOPLS approach, we train the algorithm with 3 different values of the penalty parameter, $\lambda_1 \in \{0.1, 0.5, 1\}$, in order to obtain solutions with different degrees of sparsity. As SOPLS stopping criterion, we have fixed the maximum number of iterations to 50 and the tolerance parameter δ to 10^{-5}. As in previous subsections, C-SVM is trained on the extracted features to evaluate the accuracy of the OPLS and SOPLS algorithms.

Figure 6 depicts the overall accuracy (OA, left) and sparsity rate (SR, right) of the OPLS and SOPLS solutions as a function of the number of extracted features. As expected, the sparsity rate

[5] `http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz` (233MB)



a) Overall Accuracy



b) Sparsity Rate

Figure 6: Evolution of OA and SR according to the number of projection vectors (n_f) obtained by OPLS ($\lambda_1 = 0$) and SOPLS. The behavior of SOPLS is analyzed for different values of λ_1.

grows for increasing λ_1. More importantly, we can also see that the introduction of ℓ_1 regularization leads to significantly higher accuracies. This advantage is due to the fact that in this application the original data representation has many redundant and irrelevant features, causing overfitting in the standard OPLS solution, a problem that is not suffered by sparse versions.

To analyze the advantage of the SOPLS solution from a point of view of its interpretability, Figure 7 displays the first 6 projection vectors obtained by the OPLS approach and SOPLS for different values of λ_1. We can observe that OPLS solution does not provide any useful information about the most relevant regions to classify the different faces; however, if we focus on the projection vectors provided by the SOPLS approach, mainly when a large value of λ_1 is used ($\lambda_1 = 1$), we can see how the non-zero coefficients are mostly concentrated on the eyes and mouth regions. For small values of λ_1 ($\lambda_1 = 0.1$), the location of the non-zero coefficients is not very informative; nevertheless, even in this case SOPLS avoids the overfitting problem and performs much better than standard OPLS.

6. Conclusions

Implementations of OPLS and KOPLS algorithms that are more frequently used in machine learning are based on the solution to a generalized eigenvector decomposition problem. In this paper, we have reviewed a formulation of OPLS that imposes constraints on the regression coefficients, leading to standard eigenvalue decomposition problems. We have argued in favor of these implementations for two main reasons: 1) the resulting algorithms require less memory and CPU resources, and 2) they allow the implementation of sparse OPLS and KOPLS algorithms by adding ℓ_1-regularization terms.

Exploiting this second advantage, we have proposed batch and sequential implementations of linear OPLS (SOPLS algorithm), as well as of non-linear extensions by means of kernel methods (SrKOPLS

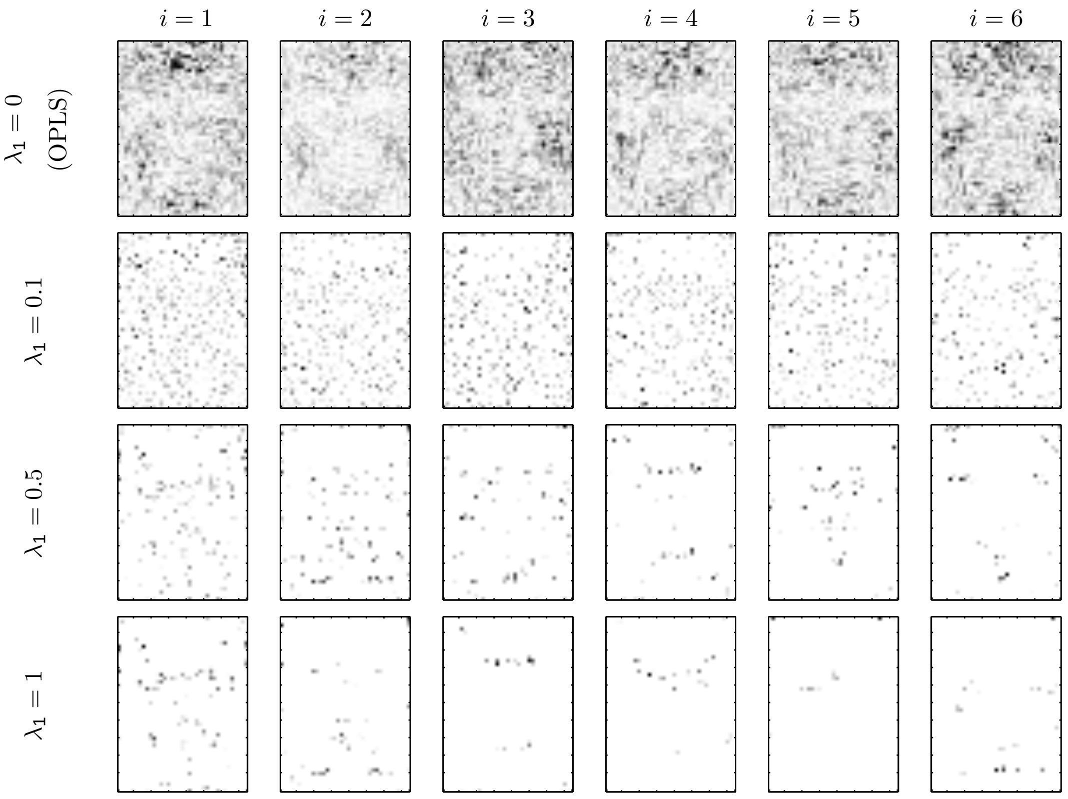


Figure 7: Six first projection vectors for different values of λ_1, with $\lambda_1 = 0$ corresponding to the OPLS algorihtm and $\lambda_1 > 0$ for the SOPLS algorithm.

algorithm). Numerical results on a benchmark of UCI datasets and in a face recognition task confirm the efficiency of our algorithms. Discriminative power superiority for iterative SOPLS is empirically proved and highly sparse projections are obtained, favoring interpretability of the solution. Regarding the SrKOPLS algorithm, it normally outperforms standard rKOPLS in most of the problems, with the additional advantage of obtaining even sparser solutions.

Further research is currently being carried out to incorporate group *lasso* concepts, to impose sparsity on entire rows of matrices $\mathbf{U}$ or $\mathbf{B}$, so that all projection vectors become dependent on the same group of original variables (linear case) or training data (kernel case).

Appendix A. Proof of Equations in (20)

We start by noting that since the columns of $\mathbf{U}_{\mathsf{EVD}}$ and $\mathbf{U}_{\mathsf{GEV}}$ span the same subspace of $\Re^n$, they should verify $\mathbf{U}_{\mathsf{EVD}} = \mathbf{U}_{\mathsf{GEV}}\mathbf{A}$, for some square and invertible matrix $\mathbf{A}$ of size n_f. Inserting this expression in (14), and taking into consideration that the columns in $\mathbf{U}_{\mathsf{GEV}}$ are $\mathbf{C}_{\mathbf{XX}}$-orthonormal, we get

$$\mathbf{A}^\top \mathbf{U}_{\mathsf{GEV}}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{U}_{\mathsf{GEV}} \mathbf{A} = \mathbf{A}^\top \mathbf{A} = \mathbf{\Lambda}_{\mathsf{EVD}}. \tag{A.1}$$

Since $\mathbf{\Lambda}_{\mathsf{EVD}}$ admits a Cholesky factorization, and this is unique, we necessarily have that $\mathbf{A} = \mathbf{A}^\top = \mathbf{\Lambda}_{\mathsf{EVD}}^{1/2}$ and

$$\mathbf{U}_{\mathsf{EVD}} = \mathbf{U}_{\mathsf{GEV}} \mathbf{\Lambda}_{\mathsf{EVD}}^{1/2}. \tag{A.2}$$

Next, we work towards showing the relationship between regression coefficient matrices. To this end, we can insert (13) into (14), obtaining $\mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{EVD}} \mathbf{\Lambda}_{\mathsf{EVD}}$. Also, if we use (A.2) with (7), we can easily show that $\mathbf{W}_{\mathsf{GEV}} \mathbf{\Lambda}_{\mathsf{EVD}}^{1/2} = \mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}_{\mathsf{EVD}}$. Using jointly these two last equations, it is straightforward to arrive at

$$\mathbf{W}_{\mathsf{EVD}} = \mathbf{W}_{\mathsf{GEV}} \mathbf{\Lambda}_{\mathsf{EVD}}^{-1/2}. \tag{A.3}$$

To conclude the proof, we need to show that $\mathbf{\Lambda}_{\mathsf{EVD}} = \mathbf{\Lambda}_{\mathsf{GEV}} = \mathbf{\Lambda}$, for which it is enough to use (A.3) together with condition $\mathbf{W}_{\mathsf{GEV}}^\top \mathbf{W}_{\mathsf{GEV}} = \mathbf{\Lambda}_{\mathsf{GEV}}$ for the classical OPLS solution. Recurring also to the orthonormality condition of the columns of $\mathbf{W}_{\mathsf{EVD}}$ we have

$$\mathbf{W}_{\mathsf{GEV}}^\top \mathbf{W}_{\mathsf{GEV}} = \mathbf{\Lambda}_{\mathsf{EVD}}^{1/2} \mathbf{W}_{\mathsf{EVD}}^\top \mathbf{W}_{\mathsf{EVD}} \mathbf{\Lambda}_{\mathsf{EVD}}^{1/2} = \mathbf{\Lambda}_{\mathsf{EVD}} = \mathbf{\Lambda}_{\mathsf{GEV}}. \tag{A.4}$$

Appendix B. Proof of Eq. (24)

Since the minimization of (23) is taken with respect to $\mathbf{W}$ only, we start by deleting regularization terms. Expanding the Frobenius norm of the least squares term, and exploiting the orthonormality constraint on the columns of $\mathbf{W}$, the objective function becomes

$$\begin{aligned} \mathcal{L}_{\mathsf{reg}}(\mathbf{W}) &= \mathrm{Tr}\{\mathbf{C}_{\mathbf{YY}}\} + \mathrm{Tr}\{\mathbf{X}'^\top \mathbf{X}'\} - \mathrm{Tr}\{\mathbf{W}^\top \mathbf{Y} \mathbf{X}'^\top\} \\ &\overset{c}{=} -\mathrm{Tr}\{\mathbf{W}^\top \mathbf{Y} \mathbf{X}'^\top\}, \end{aligned}$$

where $\mathbf{X}' = \mathbf{U}^\top \mathbf{X}$ is the projected data matrix and symbol $\overset{c}{=}$ means equal but for a constant term which does not depend on $\mathbf{W}$.

We can next incorporate the constraints on $\mathbf{W}$ using Lagrange multipliers:

$$\mathcal{L}_{\mathbf{\Lambda}}(\mathbf{W}) = -\mathrm{Tr}\{\mathbf{W}^\top \mathbf{Y} \mathbf{X}'^\top\} + \mathrm{Tr}\{\left(\mathbf{W}^\top \mathbf{W} - \mathbf{I}\right) \mathbf{\Lambda}\},$$

which needs to be minimized with respect to $\mathbf{W}$ and maximized with respect to the Lagrange multipliers. To find the solution, we take derivatives of $\mathcal{L}_{\mathbf{\Lambda}}(\mathbf{W})$ with respect to $\mathbf{W}$:

$$\frac{\partial \mathcal{L}_{\mathbf{\Lambda}}}{\partial \mathbf{W}} = -\mathbf{C}_{\mathbf{X}\ \mathbf{Y}}^\top + \mathbf{W}\mathbf{\Lambda}.$$

Finally, setting this result to zero, we know that $\mathbf{C}_{\mathbf{X}\ \mathbf{Y}}^{\top} = \mathbf{W}\boldsymbol{\Lambda}$ holds at the solution. Multiplying both sides from the right by their transpose and by $\mathbf{W}$, and using the fact that $\mathbf{W}^{\top}\mathbf{W} = \mathbf{I}$, allows us to conclude that the optimal $\mathbf{W}$ is given by the solution to the following eigenvalue problem

$$\mathbf{C}_{\mathbf{X}\ \mathbf{Y}}^{\top}\mathbf{C}_{\mathbf{X}\ \mathbf{Y}}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda}', \tag{B.1}$$

where $\mathbf{C}_{\mathbf{X}\ \mathbf{Y}} = \mathbf{U}^{\top}\mathbf{C}_{\mathbf{XY}}$ and $\boldsymbol{\Lambda}' = \boldsymbol{\Lambda}^2$.

Appendix C. Proof of initialization dependency by applying orthogonal Procrustes solution

We base this proof on the discussion from [22], where it is claimed that one of the properties a good sparse method should achieve is to reduce to the non-sparse solution (in our case, standard OPLS) when sparsity constraints are removed. Here, we prove the solution presented in [36] does not progress in the iterative process and thus it does not converge to the OPLS solution when we set to zero the ℓ_1-norm penalization and matrix $\mathbf{W}$ is initialized with an unitary matrix (i.e. $\mathbf{W}\ ^{(0)}\mathbf{W}^{(0)} = \mathbf{W}^{(0)}\mathbf{W}\ ^{(0)} = \mathbf{I}_m$ and thus $\mathbf{W}^{^{-1}(0)} = \mathbf{W}\ ^{(0)}$, implying $m < n$ and $n_f = m$) as it is the case in [36] which initializes the algorithm with the eigenvectors of $\mathbf{C}_{\mathbf{YY}}$. Note that, since the orthonormality of $\mathbf{W}$ is required (i.e. $\mathbf{W}^{\top}\mathbf{W} = \mathbf{I}$), an unitary matrix is a reasonable choice for initialization. It should be noticed that the identity matrix is a classical initialization in these cases.

To start with, [36] performs a $\mathbf{U}-$step that provides $\mathbf{U}^{(0)} = \mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C}_{\mathbf{XY}}\mathbf{W}^{(0)}$, from which $\mathbf{W}^{(1)}$ is updated according to the Procrustes solution, i.e. $\mathbf{W}^{(1)} = \mathbf{Q}\mathbf{P}^{\top}$, where $\mathbf{Q}$ and $\mathbf{P}$ are the left and right eigenvector of $\mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{U}^{(0)}$:

$$\mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{U}^{(0)} = \mathbf{QDP}^{\top}. \tag{C.1}$$

Replacing the value of $\mathbf{U}^{(0)}$, (C.1) can be rewritten as

$$\mathbf{C}\mathbf{W}^{(0)} = \mathbf{QDP}^{\top}, \tag{C.2}$$

where $\mathbf{C} = \mathbf{C}_{\mathbf{XY}}^{\top}\mathbf{C}_{\mathbf{XX}}^{-1}\mathbf{C}_{\mathbf{XY}}$ is a symmetric matrix.

With this result, we will show next that $\mathbf{W}^{(1)} = \mathbf{W}^{(0)}$, and therefore that the algorithm does not progress to the standard OPLS solution when sparsity constraints are removed. Before that, we make some linear algebra to facilitate the derivations.

Multiplying both sides of (C.2) from the right by their transposes, we obtain

$$\mathbf{QD}^2\mathbf{Q}^{\top} = \mathbf{C}\mathbf{W}^{(0)}\mathbf{W}\ ^{(0)}\mathbf{C}.$$

If we premultiply from the left instead, we get

$$\mathbf{PD}^2\mathbf{P}^{\top} = \mathbf{W}\ ^{(0)}\mathbf{C}\mathbf{C}\mathbf{W}^{(0)},$$

so that the following equalities hold:

$$\mathbf{Q} \quad = \mathbf{C}\mathbf{W}^{(0)}\mathbf{W}^{\;\;(0)}\mathbf{C}\mathbf{Q}\mathbf{D}^{-2}, \tag{C.3}$$

$$\mathbf{P} \quad = \mathbf{W}^{\;\;(0)}\mathbf{C}\mathbf{C}\mathbf{W}^{(0)}\mathbf{P}\mathbf{D}^{-2}. \tag{C.4}$$

Now, introducing (C.3) and (C.4) into the expression for $\mathbf{W}^{(1)}$, and using the fact that $\mathbf{W}^{\,^{-1}(0)} = \mathbf{W}^{\;\;(0)}$, we obtain

$$\begin{aligned}
\mathbf{W}^{(1)} &= \mathbf{Q}\mathbf{P}^{\top} \\
&= \mathbf{C}\mathbf{W}^{(0)}\mathbf{W}^{\;\;(0)}\mathbf{C}(\mathbf{Q}\mathbf{D}^{-4}\mathbf{P}^{\top})\mathbf{W}^{\;\;(0)}\mathbf{C}\mathbf{C}\mathbf{W}^{(0)} \\
&= \mathbf{C}\mathbf{W}^{(0)}\mathbf{W}^{\;\;(0)}\mathbf{C}(\mathbf{C}\mathbf{W}^{(0)})^{-4}\mathbf{W}^{\;\;(0)}\mathbf{C}\mathbf{C}\mathbf{W}^{(0)} \\
&= \mathbf{C}\mathbf{C}\mathbf{C}^{-4}\mathbf{C}\mathbf{C}\mathbf{W}^{(0)} \\
&= \mathbf{W}^{(0)},
\end{aligned}$$

and the demonstration concludes.

References

[1] H. Wold, "Estimation of principal components and related models by iterative least squares," in *Multivariate Analysis*. Academic Press, 1966, pp. 391–420.

[2] M. A. J. van Gerven, Z. C. Chao, and T. Heskes, "On the decoding of intracranial data using sparse orthonormalized partial least squares," *Journal of Neural Engineering*, vol. 9, no. 2, pp. 26 017–26 027, 2012.

[3] L. K. Hansen, "Multivariate strategies in functional magnetic resonance imaging," *Brain and Language*, vol. 102, no. 2, pp. 186–191, 2007.

[4] J. Arenas-García and G. Camps-Valls, "Efficient kernel orthonormalized PLS for remote sensing applications," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, pp. 2872–2881, 2008.

[5] J. Arenas-García and K. B. Petersen, "Kernel multivariate analysis in remote sensing feature extraction," in *Kernel Methods for Remote Sensing Data Analysis*, G. Camps-Valls and L. Bruzzone, Eds. Wiley, 2009.

[6] M. Barker and W. Rayens, "Partial least squares for discrimination," *Journal of Chemometrics*, vol. 17, no. 3, pp. 166–173, 2003.

[7] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.

[8] H. Wold, "Non-linear estimation by iterative least squares procedures," in *Research Papers in Statistics*. Wiley, 1966, pp. 411–444.

[9] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, pp. 321–377, 1936.

[10] K. Worsley, J. Poline, K. Friston, and A. Evans., "Characterizing the response of pet and fMRI data using multivariate linear models (MLM)," *Neuroimage*, vol. 6, pp. 305–319, 1998.

[11] M. Borga, T. Landelius, and H. Knutsson, "A unified approach to PCA, PLS, MLR and CCA," Institutionen för Systemteknic, Linköping, Sweden, Tech. Rep. LiTH-ISY-R-1992, 1997.

[12] G. C. Reinsel and R. P. Velu, *Multivariate reduced-rank regression: theory and applications*. New York, NY: Springer, 1998.

[13] F. D. Torre, "A least-squares framework for component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1041–1055, 2012.

[14] S. Roweis and C. Brody, "Linear heteroencoders," Gatsby Computational Neuroscience Unit, UK, Tech. Rep. GCNU TR 1999-002, 1999.

[15] J. Arenas-García, K. B. Petersen, and L. K. Hansen, "Sparse kernel orthonormalized PLS for feature extraction in large data sets," *Advances in Neural Information Processing Systems 19*, Cambridge, MA: MIT Press, 2007.

[16] C. Dhanjal, S. R. Gunn, and J. Shawe-Taylor, "Efficient sparse kernel feature extraction based on partial least squares," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1347–1361, 2009.

[17] J. Friedman, T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "[consistency in boosting]: Discussion," *The Annals of Statistics*, vol. 32, no. 1, pp. 102–107, 2004.

[18] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[19] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *Journal of Machine Learning Research*, vol. 3, pp. 1229–1243, 2003.

[20] Z. J. Xiang and P. J. Ramadge, "Fast lasso screening tests based on correlations," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*. IEEE, 2012, pp. 2137–2140.

[21] M. Dyar, M. Carmosino, E. Speicher, M. Ozanne, S. Clegg, and R. Wiens, "Comparison of partial least squares and lasso regression techniques as applied to laser-induced breakdown spectroscopy of geological samples," *Spectrochimica Acta Part B: Atomic Spectroscopy*, 2012.

[22] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, pp. 265–286, 2006.

[23] D. Meng, Q. Zhao, and Z. Xu. "Improve robustness of sparse PCA by L_1-norm maximization," *Pattern Recognition*, vol. 45, no. 1, pp. 487-497, 2012.

[24] D. Hardoon and J. Shawe-Taylor, "Sparse canonical correlation analysis," *Machine Learning*, vol. 83, no. 3, pp. 331–353, 2011.

[25] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[26] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, 2001.

[27] J. Arenas-Garcıa, K. B. Petersen, G. Camps-Valls, and L. K. Hansen, "Kernel multivariate analysis framework for supervised subspace learning," *IEEE Signal Process. Mag.*, pp. 16–29, 2013.

[28] B. Scholkopf, A. Smola, and K.-R. Muller, "Non linear component analysis as kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[29] P. L. Lai and C. Fyfe, "Kernel and nonlinear canonical correlation analysis," *International Journal of Neural Systems*, vol. 10, no. 5, pp. 365–377, 2000.

[30] R. Rosipal and L. J. Trejo, "Kernel partial least squares regression in reproducing kernel hilbert space," *Journal of Machine Learning Research*, vol. 2, pp. 97–123, 2001.

[31] D. Hardoon, J. Mourao-Miranda, M. Brammer, and J. Shawe-Taylor, "Unsupervised analysis of fMRI data using kernel canonical correlation," *NeuroImage*, vol. 37, no. 4, pp. 1250–1259, 2007.

[32] W. Zheng, X. Zhou, C. Zou, and L. Zhao, "Facial expression recognition using kernel canonical correlation analysis (KCCA)," *IEEE Trans. Neural Networks*, vol. 17, no. 1, pp. 233–238, 2006.

[33] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor, "Primal space sparse kernel partial least squares regression for large scale problems," in *Proc. 2004 IEEE International Joint Conference on Neural Networks*, 2004.

[34] K. B. M. Momma, "Sparse kernel partial least squares regression," in *Proc. of Conference on Learning Theory (COLT 2003)*, 2003, pp. 216–230.

[35] D. Huang and F. D. Torre, "Bilinear kernel reduced rank regression for facial expression synthesis," in *Proc. European Conf. Computer Vision (ECCV)*, 2010. pp.364–377.

[36] M. Gerven and T. Heskes, "Sparse orthonormalized partial least squares," in *Proc. Benelux Conf. on Artificial Intelligence*, 2010.

[37] L. Chen and J. Z. Huang, "Sparse Reduced-Rank Regression for Simultaneous Dimension Reduction and Variable Selection," Journal of American Statistical Association, vol. 107, no. 500, pp. 1533–1545, 2012.

[38] C. Bishop, *Neural Networks for Pattern Recognition*. New York, NY: Oxford University Press, 1995.

[39] L. Sun, S. Ji, S. Yu, and J. Ye, "On the equivalence between canonical correlation analysis and orthonormalized partial least squares," in *Proc. of the 21st International Joint Conference on Artificial Intelligence*, 2009. pp. 1230–1235.

[40] T. T. Ngo, M. Bellalij, and Y. Saad, "The trace ratio optimization problem," *SIAM Rev.*, vol. 54, pp. 545–569, 2012.

[41] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Trans. Neural Networks*, vol. 20, pp. 729–735, 2009.

[42] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012.

[43] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301–320, 2005.

[44] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Convex optimization with sparsity-inducing norms," in *Optimization for Machine Learning*, MIT Press, 2011, pp. 19–53.

[45] G. X. Yuan, K. W. Chang, C. J. Hsieh, and C. J. Lin, "A comparison of optimization methods and software for large-scale L1-regularized linear classification," *Journal of Machine Learning Research*, vol. 11, pp. 3183–3234, 2010.

[46] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv preprint arXiv:1001.0736*, 2010.