




Multiblock PLS: Block dependent prediction modeling for Python

Andreas Baum¹ and Laurent Vermue¹

¹ Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads 324, DK-2800 Kgs. Lyngby, Denmark

DOI: [10.21105/joss.01190](https://doi.org/10.21105/joss.01190)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Submitted: 06 January 2019

Published: 09 February 2019

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Introduction

Partial Least Squares (PLS) regression is a statistical method for supervised multivariate analysis. It relates two data blocks \mathbf{X} and \mathbf{Y} to each other with the aim of establishing a prediction model. When deployed in production, this model can be used to predict an outcome \mathbf{y} from a newly measured feature vector \mathbf{x} . PLS is popular in chemometrics, process control and other analytic fields, due to its striking advantages, namely the ability to analyze small sample sizes and the ability to handle high-dimensional data with cross-correlated features (where Ordinary Least Squares regression typically fails). In addition, and in contrast to many other machine learning approaches, PLS models can be interpreted using its latent variable structure just like principal components can be interpreted for a PCA analysis.

Multivariate data is often structured in blocks, e.g. $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_i$. This could mean that one has obtained data from two different analytic methodologies for a similar set of samples, which may indicate two totally independent feature spaces. In such cases it is often important to understand how each data block contributes to the prediction of \mathbf{Y} . Examples for data measured in blocks could be the following.

1. It can be of interest to relate patient clinical records to data obtained through different high-throughput omics measurements. These data could typically be structured in blocks referring to genomics, transcriptomics, proteomics, metabolomics etc.
2. Spectroscopic methods are useful to predict and assure food quality parameters. When measuring food samples by several different spectroscopic methods, e.g. by applying near infrared and UV-vis spectroscopy, it is meaningful to combine the data blocks to obtain reliable prediction models.
3. A process utilizing fermentation technology is typically carried out in several sequential production phases, i.e. seed and main fermentation phase. If sensor data is available for all production phases it is meaningful to include these as individual data blocks when establishing prediction models for quality control parameters, such as product yield.

Several Data Fusion approaches were proposed to establish combined prediction models from such multiblock data (Li, Wu, & Ngom, 2016). One of the proposed methods is Multiblock-PLS (MB-PLS) (Westerhuis, Kourti, & MacGregor, 1998). It is closely related to PLS regression, but instead of obtaining an interpretative model for the entire (concatenated) data matrix \mathbf{X} one obtains model parameters for each individual data block \mathbf{X}_i . Furthermore, it provides a relative importance measure, i.e. expressing how

much each block \mathbf{X}_i contributes to the prediction of \mathbf{Y} . Subsequently, this information can be used to recognize block specific patterns in the data.

At the current stage software packages for MB-PLS exist for Matlab (<http://www.models.life.ku.dk/MBToolbox>) and R (Bougeard & Dray, 2018). In the following sections we give a brief introduction to the statistical method and its implementation. The package is distributed under the BSD-3-Clause license and made available at <https://github.com/DTUComputeStatisticsAndDataAnalysis/MBPLS> together with several introductory Jupyter notebook examples. It is also available as pip installable Python package (<https://pypi.org/project/mbpls/>) and comes with a Read-the-Docs documentation (<https://mbpls.readthedocs.io>).

Methods

The MB-PLS package can be utilized for PLS and MB-PLS regression. The statistical background is briefly introduced in the following. More detailed information is given in the `mbpls` help of the Python package (<https://mbpls.readthedocs.io/en/latest/mbpls.html>).

PLS

PLS was introduced by S. Wold, Ruhe, Wold, & III (1984) and aims at finding a suitable subspace projection \mathbf{w} which maximizes co-variance between a so called score vector \mathbf{t} and a response vector \mathbf{y} that will yield a least squares solution. The formal PLS criterion for univariate responses \mathbf{y} is given in eq. 1.

$$\operatorname{argmax}_{\mathbf{w}} \left(\operatorname{cov}(\mathbf{t}, \mathbf{y}) \mid \min \left(\sum_{i=1}^I \sum_{j=1}^J (\mathbf{x}_{ij} - \mathbf{t}_i \mathbf{w}_j)^2 \right) \wedge \|\mathbf{w}\| = 1 \right) \quad (1)$$

This procedure is typically repeated to find K latent variables (LV). In each latent variable step, the score vector \mathbf{t}_k is subsequently projected onto its respective matrix \mathbf{X}_k to find the loading vector \mathbf{p}_k (eq. 2). Once found, \mathbf{X}_k is deflated by the explained variance (eq. 3) and the next latent variable $k + 1$ can be calculated using \mathbf{X}_{k+1} .

$$\mathbf{p}_k = \mathbf{X}_k \mathbf{t}_k \quad (2)$$

$$\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{t}_k \mathbf{p}_k^T \quad (3)$$

Algorithms to perform PLS regression include the Nonlinear Iterative **P**artial Least Squares (NIPALS) (S. Wold et al., 1984), **U**Niversal **P**artial Least Squares (UNIPALS) (Dunn III, Scott, & Glen, 1989), Kernel UNIPALS (Lindgren, Geladi, & Wold, 1993; Rännar, Geladi, Lindgren, & Wold, 1995; Rännar, Lindgren, Geladi, & Wold, 1994) and SIMPLS algorithm (de Jong, 1993). While NIPALS represents an iterative approach, the other algorithms are based on Singular Value Decomposition (SVD). All the above mentioned algorithms are implemented in the MB-PLS package. Benchmark results and comparisons to other Software packages are provided below.

MB-PLS

MB-PLS can be understood as an extension of PLS to incorporate several data blocks $\mathbf{X}_1, \dots, \mathbf{X}_I$, which all share a common sample dimension. The prediction accuracy does not deviate from normal PLS, if all data blocks were concatenated into a single block, but the

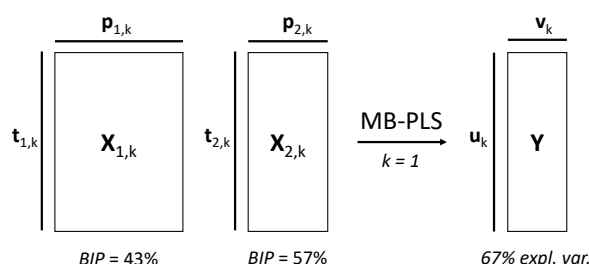


Figure 1: The figure illustrates the extraction of a single LV ($k = 1$). MB-PLS offers extra exploratory features for each block, i.e. block scores, block loadings and block importances (BIP).

advantage of MB-PLS is to gain extra model interpretability concerning the underlying block structure of the data. For each LV one obtains extra block scores, block loadings and block importances (BIP). The extraction of a single LV using MB-PLS is illustrated in figure 1. The results are read in a fashion that 67% variance in Y are explained by the first LV. The two blocks X_1 and X_2 contribute to the prediction of the 67% with their relative BIPs, 43% and 57%, respectively. More important blocks result in more influential block loadings and contribute stronger to the prediction of Y . Hence, interpretation of patterns among block scores with high importance are recommended.

To assert that the BIP is a meaningful indicator it is necessary to standardize the data prior to MB-PLS analysis. When standardization is employed all features in all blocks have a variance of 1. For post-hoc interpretation of the loadings an inverse transformation is carried out to ensure straight forward interpretation of the results.

Software and Implementation

The package is written in pure Python 3. In its core it builds on Numpy and Scipy for efficient data handling and fast mathematical operations of big data-sets. To achieve a fast implementation all algorithms using SVD employ Scipy's partial SVD capability, i.e. by only calculating the first singular value at each PLS iteration. Multiple matrix multiplications use the optimized Numpy multi-array multiplication. In addition, the MB-PLS implementation can handle missing data without prior imputation based on the sparse NIPALS algorithm by H. Martens & Martens (2001). The overall code design follows the structure and philosophy of Scikit-learn (Pedregosa et al., 2011). Therefore, objects are instantiated in a Scikit-learn manner and can be accessed with the same methods, i.e. fit, predict, transform and score. Furthermore, Scikit-learn's base classes and validation methods are incorporated. As a result, all objects are fully compatible to Scikit-learn and, thus, allow the use of model selection functions, e.g. cross validation and grid search, as well as a processing pipeline. For exploratory analysis, each fitted model contains a custom plot method that the fitted model attributes in a meaningful manner using Matplotlib, which allows a straight forward evaluation of the MBPLS results without requiring any additional coding.

Benchmark

To compare the four algorithms, the run-times are analyzed for different data-set sizes with two basic shapes, i.e. non-symmetric shapes with more samples than variables ($N > P$) or vice versa ($N < P$) and symmetric shapes ($N = P$). To simulate the multiblock and multivariate behaviour, each data-set is split into two X -blocks with the size $N \times \frac{P}{2}$ and

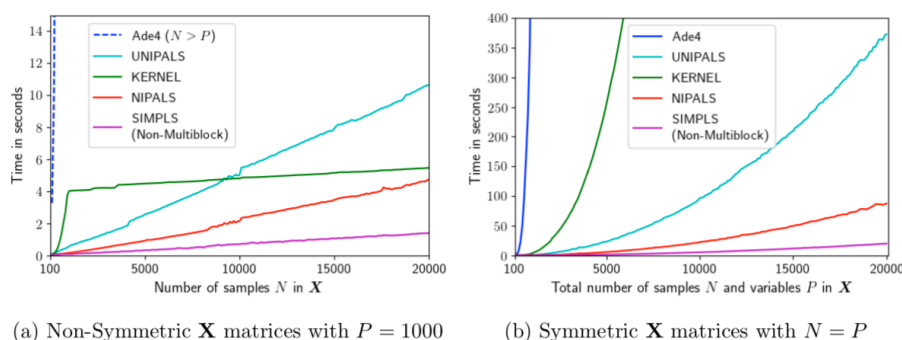


Figure 2: Comparison of run-times based on different data-set sizes

accompanied by a Y -block of size $N \times 10$. The data is randomly generated for each run, so that the obtained times exhibit worst-case behaviour, since there are no actual latent structures. All algorithms are set to find the first 20 LVs and are run three times for each data-set size on a machine with two Intel® Xeon® X5650 @ 2.67 GHz processors and 48 GB RAM.

As to be seen in both plots of figure 2 all algorithms implemented in the Python mbpls package substantially outperform the above mentioned R-package Ade4-MBPLS by Bougeard & Dray (2018), which was run on the same machine. In general NIPALS is the fastest multiblock algorithm that is only outperformed by the SIMPLS algorithm, which only supports single block PLS. However, figure 2a shows how the KERNEL algorithm performs progressively better in cases where $N \gg P$ or $N \ll P$. As to be seen in this plot, the runtime of this algorithm is a combination of an exponential part given by the right plot and dependent on $\min(N, P)$ and a linear part defined by $\text{diff}(N, P)$. Due to the exponential part, $\min(N, P)$ has to be considered carefully when choosing the KERNEL algorithm over NIPALS.

An important feature of this Python mbpls package is its invariance to shape rotations, i.e. it obtains the same run-times for both $N > P$ and $N < P$ given the same ratio $\frac{N}{P}$ and its respective inverse, which e.g. is not the case for the R-package.

Acknowledgement

The authors gratefully acknowledge the financial support through the BioPro (Innovationsfonden project nr. 10513) and DABAI (Innovationsfonden project nr. 10599 and 10577) project.

References

- Bougeard, S., & Dray, S. (2018). Supervised multiblock analysis in r with the ade4 package. *Journal of Statistical Software*, 86(1), 1–17. doi:[10.18637/jss.v086.i01](https://doi.org/10.18637/jss.v086.i01)
- de Jong, S. (1993). SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems*, 18(3), 251–263. doi:[10.1016/0169-7439\(93\)85002-X](https://doi.org/10.1016/0169-7439(93)85002-X)
- Dunn III, W. J., Scott, D. R., & Glen, W. G. (1989). Principal components analysis and partial least squares regression. *Tetrahedron computer methodology*, 2(6), 349–376. doi:[10.1016/0898-5529\(89\)90004-3](https://doi.org/10.1016/0898-5529(89)90004-3)

- Li, Y., Wu, F.-X., & Ngom, A. (2016). A review on machine learning principles for multi-view biological data integration. *Briefings in Bioinformatics*, 19(2), 325–340. doi:[10.1093/bib/bbw113](https://doi.org/10.1093/bib/bbw113)
- Lindgren, F., Geladi, P., & Wold, S. (1993). The kernel algorithm for pls. *Journal of Chemometrics*, 7(1), 45–59. doi:[10.1002/cem.1180070104](https://doi.org/10.1002/cem.1180070104)
- Martens, H., & Martens, M. (2001). *Multivariate analysis of quality: An introduction*. Chichester: Wiley.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.
- Rännar, S., Geladi, P., Lindgren, F., & Wold, S. (1995). A pls kernel algorithm for data sets with many variables and few objects. Part ii: Cross-validation, missing data and examples. *Journal of Chemometrics*, 9(6), 459–470. doi:[10.1002/cem.1180090604](https://doi.org/10.1002/cem.1180090604)
- Rännar, S., Lindgren, F., Geladi, P., & Wold, S. (1994). A pls kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm. *Journal of Chemometrics*, 8(2), 111–125. doi:[10.1002/cem.1180080204](https://doi.org/10.1002/cem.1180080204)
- Westerhuis, J. A., Kourti, T., & MacGregor, J. F. (1998). Analysis of multiblock and hierarchical pca and pls models. *Journal of Chemometrics*, 12(5), 301–321. doi:[10.1002/\(SICI\)1099-128X\(199809/10\)12:5<301::AID-CEM515>3.0.CO;2-S](https://doi.org/10.1002/(SICI)1099-128X(199809/10)12:5<301::AID-CEM515>3.0.CO;2-S)
- Wold, S., Ruhe, A., Wold, H., & III, W. J. D. (1984). The collinearity problem in linear regression. The partial least squares (pls) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3), 735–743. doi:[10.1137/0905052](https://doi.org/10.1137/0905052)