A New Micro-Batch Approach for Partial Least Square Clusterwise Regression

Gaël Beck, Hanane Azzag, Stéphanie Bougeard, Mustapha Lebbah, Ndèye Niang

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 144 (2018) 239–250

Procedia Computer Science

www.elsevier.com/locate/procedia

INNS Conference on Big Data and Deep Learning 2018

A New Micro-Batch Approach for Partial Least Square Clusterwise Regression

Beck Gaël[a,*], Azzag Hanane[a], Bougeard Stéphanie[b], Lebbah Mustapha[a], Niang Ndèye[c]

[a] Computer Science Lab of Paris 13 University, 99 Avenue Jean-Baptiste Clément Villetaneuse, Ile de France, 93430, France
[b] Anses, Department of Epidemiology, Ploufragan, 22440, France
[c] CEDRIC CNAM, 292 rue St Martin, Paris Cedex 03, 75141, France

Abstract

Current implementations of Clusterwise methods for regression when applied to massive data either have prohibitive computational costs or produce models that are difficult to interpret. We introduce a new implementation Micro-Batch Clusterwise Partial Least Squares (mb-CW-PLS), which is consists of two main improvements: (a) a scalable and distributed computational framework and (b) a micro-batch Clusterwise regression using buckets (micro-clusters). With these improvements, we are able to produce interpretable regression models with multicollinearity within a reasonable time frame.

Keywords: Clusterwise, PLS, Spark

1. Introduction

In modern data analysis, many problems belong to the regression family which consist of explaining one or more variable (known as the response) with respect to other observed variables (known as the explanatory variables). Many types of regression models exist to solve specific problems. A widely used regression method is Multivariate Linear Regression (MLR) where many explanatory variables are linked to a specific response variable by a parametric linear model [14]. MLR is most suited to problems where the number of observations is larger than the number of explanatory variables. On the other hand, when the number of explanatory variables is larger (e.g. high dimensional data), then there tends to be important collinearities between them which ensures that MLR is not effective.

Partial least squares regression (PLS) is a linear regression model with latent features for high-dimensional data [21, 12]. Standard PLS defines new components by maximizing the covariance between components from two different blocks of variables (data matrix **X** and its response matrix **Y**). In addition, standard PLS does not require any

* Corresponding author. Tel.: +33-659373517.
E-mail address: beck@lipn.univ-paris13.fr

10.1016/j.procs.2018.10.525

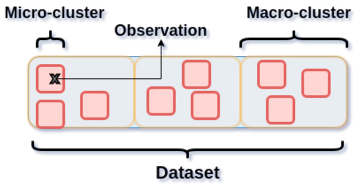


Fig. 1: Distinction between micro and macro clusters

distributional assumptions for the error distributions, since they need not be normal or other parametric distributions. Applying PLS to massive high-dimensional data is problematic because the data processing is computationally expensive. This has limited large-scale applications of PLS in practice. To reduce the computational burden, researchers usually apply a specific model design, such as the Clusterwise PLS [3, 21, 12].

In this paper, we address a Clusterwise problem where the response variables **y** are explained by the explanatory variables **x**, organized into clusters. The statistical model which results from simultaneously treating all observations (i.e. a single macro-cluster) may be of low prediction quality. To overcome this problem we use two nested levels of clustering (i.e. macro- and micro- clusters) illustrate in Figure 1 and compute one model per macro-cluster based on a micro-batch strategy where micro-clusters are shifted from one macro-cluster to another. A standard approach to obtain clusters within a regression framework is Clusterwise regression (which is also known as typological regression) [6, 18].

Clusterwise regression assumes that there is an underlying clustering structure of the observations and that each cluster can be revealed by the fit of a specific regression model based on micro-clusters. More formally, Clusterwise regression simultaneously looks for a partition of the observations into clusters which minimizes the overall sum of squared error. For Clusterwise methods, a crucial component is to describe the local relationships between the variables measured on the observations within the same cluster. This is handled in this paper by micro-batch approaches.

The remainder of the paper is organized in three sections. Section 2 summarizes previous results which are related with our work. In Section 3, the traditional PLS and the micro-batch version are presented. Section 4 focuses on the performance indicators including the prediction quality, execution times and scalability. Section 5 contains the conclusions and future perspectives of our work.

Used notations

In the following read, matrix are in bold upper case, vectors in bold lower case. Micro-clusters are those generate after being applied a clustering algorithm , here k-means. Macro-clusters are those generated for the Clusterwise purpose, their number will be equal to G. **X** and **Y** are respectively the data and response matrix. Response variables will use **y** and **x** will stand for the explanatory variables.

2. Related works

Existing Clusterwise methods seek clusters within a regression framework while simultaneously minimizing the sum of squared error computed over all the clusters. These methods can be viewed as extensions of the k-means clustering from unsupervised learning to the regression set-up. As in standard regression, ordinary least squares or maximum likelihood estimation can be used to get the quality of the regression coefficients. These k-means like algorithms are based on a least square error criterion have been proposed by [2, 7]. A multivariate regression for heterogeneous data which takes into account both the between- and the within-cluster variability has also been proposed [13].

To detect categorical differences in underlying regression models on the other hand, Spath [18] developed Clusterwise Regression (CR) which clusters the data points based on the underlying regression model. Related methods exist within the mixture and latent class framework [6]. Other related methods are the principal component regression (PCR) [4] and partial least square regression (PLS) [19, 15] which have also been proposed to deal with multicollinearity, small sample size, or large number of variables. Specifically, PLS reduces the explanatory variables to those which are maximally related (in terms of squared covariance) as possible to the objective function.

In the framework of component-based path-modeling methods, several Clusterwise methods have been applied in the marketing field (for an early review, refer to [16]). In [9] the authors propose the widely-used finite-mixture PLS (FIMIX-PLS) which assume multivariate normally distributed data. In [10], fuzzy Clusterwise generalized structured component analysis (FCGSCA) is proposed. In [8] the authors propose REBUS-PLS which came from the hierarchical clustering based on a similarity measure defined from the residuals coming from the same models. [17] proposed PLS-IRRS which identifies homogeneous clusters that have similar residual values.

In the field of multigroup analysis where the groups of observations are known *a priori* Clusterwise simultaneous component analysis (CW-SCA) seeks clusters among groups of observations rather than among observations [5]. It is worth noting that likelihood-based methods are relevant for data exploration or modeling but are unable to be utilized for prediction as dependent values are needed to compute the likelihood.

3. Micro-Batch PLS for Clusterwise

In this section we present a new mb-CW-PLS algorithm based on combining clustering and PLS. We first review the classical PLS algorithm and then we focus on our proposition.

3.1. Partial Least Square (PLS)

We use the common notation where scalars are defined as italic lower case (x, y), vectors are in bold lower case $(\mathbf{x}, \mathbf{y})$ and matrices as bold upper case $(\mathbf{X}, \mathbf{Y})$.

PLS regression is a technique that generalizes and combines features from principal component analysis and multiple regression. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$ be N vector observations $\mathbf{x_i} = (x_{i1}, \ldots, x_{ip}) \in \mathfrak{R}^p$, described by the vector response variables $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ where $\mathbf{y}_i = (y_{i1}, \ldots, y_{iq}) \in \mathfrak{R}^q$. Each pair of explanatory and response variables in the model is denoted by the concatenated vector $\mathbf{z}_i = (\mathbf{x}_i; \mathbf{y}_i)$. The matrices $\mathbf{Y}$ and $\mathbf{X}$ are assumed to be pre-whitened (i.e., the sum of each variable is zero and its norm is one). The superscript T denotes the matrix transpose operation, e.g. $\mathbf{X}^T$ and $\mathbf{I}$ the identity matrix.

PLS regression is particularly well-suited when the matrix of explanatory variables $\mathbf{X}$ has more features p than observations, and when there is multicollinearity among the $\mathbf{X}$ values. The goal of PLS regression is to predict $\mathbf{Y}$ from $\mathbf{X}$ and to describe their common structure. When $\mathbf{Y}$ is a vector and $\mathbf{X}$ is sufficiently regular, this goal could be accomplished using ordinary multiple regression. PLS solves the problem that arises when number of observations (N) is much lower than the number of variables (p).

PLS regression performs a simultaneous decomposition of $\mathbf{X}$ and $\mathbf{Y}$ with the constraint that these components capture as much as possible of the covariance between $\mathbf{X}$ and $\mathbf{Y}$. More formally, $\mathbf{X}$ and $\mathbf{Y}$ are decomposed as follows:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E}, \; x_{ik} = \sum_{j=1}^{r} t_{ij} p_{kj} + e_{ik} \tag{1}$$

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^T + \mathbf{F}, \; y_{im} = \sum_{j=1}^{r} u_{ij} q_{mj} + f_{im} \tag{2}$$

where $\mathbf{T}$ and $\mathbf{U}$ are the r-dimensional latent representations of $\mathbf{X}$ and $\mathbf{Y}$ of size of $N \times r$, $\mathbf{P}$ and $\mathbf{Q}$ are the loading matrices with size of $p \times r$ and $q \times r$, $\mathbf{E}$ and $\mathbf{F}$ are the residual matrices. Thus PLS implements the following optimization problem:

$$(\mathbf{p}, \mathbf{q}) = \text{argmax}_{||\mathbf{p}||=||\mathbf{q}||=1} \; cov(\mathbf{T}, \mathbf{U})$$

where $\mathbf{p}, \mathbf{q}$ are columns of the $\mathbf{P}, \mathbf{Q}$ respectively

Once the decomposition of the $\mathbf{X}$ and $\mathbf{Y}$ is carried out, we can continue with computing the regression coefficients. The latter are defined as the minimizers of the residual error

$$\text{argmin}_{\mathbf{B}}||\mathbf{Y} - \mathbf{XB}||^2$$

where $\mathbf{B}$ is a p by q regression coefficient matrix. Suppose there is a linear relationship such that

$$\mathbf{U} = \mathbf{TD} + \mathbf{H}$$

where $\mathbf{D}$ is an $r \times r$ diagonal matrix, then

$$\mathbf{Y} = \mathbf{TC}^T + \mathbf{F}^*$$

where $\mathbf{C}^T = \mathbf{DQ}^T$ and $\mathbf{F}^* = \mathbf{HQ}^T + \mathbf{F}$. Based on the previous expressions, the response matrix is expressed as

$$\mathbf{Y} = \mathbf{XP}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{CT} + \mathbf{F}^*\mathbf{EP}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{CT}.$$

The PLS regression coefficient is thus

$$\mathbf{B} = \mathbf{P}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{C}^T$$

For the brevity, we have discussed the PLS where the variables form a single macro-cluster. When the data consist of non-homogeneous macro-clusters, it is necessary to decompose the regression according to these macro-clusters, for example, using the Clusterwise techniques described in Section 2. These techniques have many bottlenecks as the scalability for a large number of observations and variables. One of the main objectives of this paper is to integrate the micro-batch processing into the Clusterwise PLS to resolve these scalability issues.

3.2. New model: Micro-Batch Clusterwise PLS (mb-CW-PLS)

3.2.1. Overview

Our proposed approach mb-CW-PLS (micro-batch Clusterwise Partial Least Squares) has the following properties:

- Nested clustering: we combine the PLS with two levels of clustering (macro- and micro-clustering) and micro-batch optimization approaches to create a new Clusterwise method that can find the underlying structure of the observations and provide each macro-cluster of observations with its own set of regression coefficients.
- Micro-Batch processing (divide and conquer strategies): rather than move a single observation from a macro-cluster to another to calculate the regression models in a cross validation approach, we move an entire micro-cluster of points, which we call the micro-cluster shift. These micro-clusters are computed using the k-means clustering, where k is set as the ratio of dataset size to the desired micro-cluster size.
- Scalability: we use a distributed framework based on Apache Spark/Scala to accelerate the initialization process and to compute the regression models. For each cross validation, we distribute initializations over the slave processes. Each slave selects the optimal initialization and submits it to the master process, which in turns selects the optimal one among all the results received from the slaves. This decreases execution times inversely proportional to the number of slaves.

- Usability: the Spark/Scala implementation requires the configuration of a small number of hyper parameters, and can be utilized in a distributed system or even on a stand-alone terminal with minimal effort, which enlarges the scope of usability of the PLS.

3.2.2. Mathematical details

We assume that the N observations are clustered into K micro-clusters (or buckets), $C = \{C_1, \ldots, C_k, \ldots, C_K\}$ where $C_k = \{\mathbf{x}_i, \phi(\mathbf{x}_i) = k\}$. Denote ϕ as the assignment function defined as follow : $\phi : \mathbf{x}_i \in \Re^p \to \{1, \ldots, k, \ldots, K\}$ (eg. euclidean distance). The partition C of the micro-clusters is carried as an initialization before running the PLS models. This step can be done using clustering approaches such as k-means.

Denote a second partition level $\mathcal{P} = \{P_1, ..., P_g, ..., P_G\}$ of the micro-clusters set C into G macro-clusters (and $K \ll G$). Therefore G is the number of regression models that will be considered ($PLS_1, ..., PLS_g, ..., PLS_G$). The second level assignment function Φ is defined as: $\Phi : \{C_1, \ldots, C_k, \ldots, C_K\} \to \{1, \ldots, g, \ldots, G\}$.

We introduce Micro-Bach Clusterwise PLS methods (mb-CW-PLS) by assuming two phases of clustering: in the first level the N observations is clustered in K fixed micro-clusters, and in the second level, the K micro-clusters are grouped into G macro-clusters where each macro-cluster has a specific PLS regression model. Therefore the mb-CW-PLS algorithm searches for an optimal partition of the K micro-clusters into G macro-clusters as well as the corresponding set of regression coefficient matrices $(\mathbf{B}_1, ..., \mathbf{B}_G)$ that minimize the overall error described in Equation 3:

$$\begin{aligned} L(\mathcal{P}, \mathbf{B}) &= \sum_{g=1}^{G} \sum_{\mathbf{x}_i \in P_g} \|\mathbf{Y}_g - \mathbf{X}_g \mathbf{B}_g\|^2 \\ &= \sum_{g=1}^{G} \sum_{C_k \in P_g} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{y}_i - \mathbf{x}_i \mathbf{b}_g^T\|^2 \end{aligned} \tag{3}$$

where $\mathbf{X}g$ and $\mathbf{Y}_g$ denote the data matrices of the gth macro-cluster respectively of $\mathbf{X}$ and $\mathbf{Y}$, i.e.

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 = \begin{bmatrix} x_{11} & \ldots & x_{1p} \\ & \vdots & \\ x_{N_1 1} & \ldots & x_{N_1 p} \end{bmatrix} \\ \vdots \\ \mathbf{X}_g = \begin{bmatrix} x_{11} & \ldots & x_{1p} \\ & \vdots & \\ x_{N_g 1} & \ldots & x_{N_g p} \end{bmatrix} \\ \vdots \\ \mathbf{X}_G = \begin{bmatrix} x_{11} & \ldots & x_{1p} \\ & v \ldots & \\ x_{N_G 1} & \ldots & x_{N_G p} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \phi(\mathbf{x}_1) & \Phi(C_{\phi(\mathbf{x}_1)}) \\ \vdots & \vdots \\ \phi(\mathbf{x}_N) & \Phi(C_{\phi(\mathbf{x}_N)}) \end{bmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 = \begin{bmatrix} y_{11} & \ldots & y_{1q} \\ & \vdots & \\ y_{N_1 1} & \ldots & y_{N_1 q} \end{bmatrix} \\ \vdots \\ \mathbf{Y}_g = \begin{bmatrix} y_{11} & \ldots & y_{1q} \\ & \vdots & \\ y_{N_g 1} & \ldots & y_{N_g y} \end{bmatrix} \\ \vdots \\ \mathbf{Y}_G = \begin{bmatrix} y_{11} & \ldots & y_{1q} \\ & \vdots & \\ y_{N_G 1} & \ldots & y_{N_G q} \end{bmatrix} \end{bmatrix}$$

The classical Clusterwise PLS does not provide estimators in reasonable time for large N. Based on the traditional sequential algorithm, each observation $\mathbf{x}_i$ is assigned to its optimal cluster and the overall error is updated whenever one observation switches cluster. In order to overcome this problem and to ensure that the error decreases monotonically at each iteration of the algorithm, we propose to use a new sequential micro-batch algorithm. Each micro-cluster C_k is assigned to its optimal macro-cluster P_g and the overall error is updated whenever a micro-cluster switches to a different macro-cluster. Therefore the main idea, instead moving a single observation $\mathbf{x}_i$ from its original macro-cluster to other macro-clusters, we move the entire micro-cluster C_k which contains $\mathbf{x}_i$ and then re-compute the regression model. Starting with an initial partition of macro-clusters $\mathcal{P}^{(0)} = \{P_0^{(0)}, \ldots, P_G^{(0)}\}$, the algorithm constructs iteratively a sequence $\{\mathcal{P}^{(s)}, \mathbf{B}^{(s)}\}$, $s > 0$, in the following way:

- For each $g \in \{1 \ldots, G\}$, $\mathcal{P}^{(0)}$ is given by the least square estimators of the PLS regression using the points of the macro-cluster P_g.
- Given $(\mathcal{P}^{(s)}, \mathbf{B}_g^{(s)})$, then for each macro-cluster $P_g \in \{P_1, \ldots, P_G\}$,

$$\mathcal{P}_g^{(s+1)} = \{(\mathbf{X}_i, \mathbf{Y}_i) : \|\mathbf{Y}_i - \mathbf{X}_i\mathbf{B}_{g_1}^{(s)}\|^2 < \|\mathbf{Y}_i - \mathbf{X}_i\mathbf{B}_{g_2}^{(s)}\|^2, \forall g_1 \neq g_2\}. \tag{4}$$

  The resulting $\mathbf{B}^{(s+1)}$ are the PLS estimators using the data partitioned by $\mathcal{P}^{(s+1)}$. The sequence $\{(\mathcal{P}^{(s)}, \mathbf{B}^{(s)})\}_{s\geq 0}$ is such that

$$L(\mathcal{P}^{(s)}, \mathbf{B}^{(s)}) \geq L(\mathcal{P}^{(s+1)}, \mathbf{B}^{(s+1)}),\ \forall s \geq 0$$

  and so it is convergent.
- Repeat above step for all micro-clusters C_k which are re-assigned to their optimal macro-cluster P_g. This guarantees that the overall error decreases monotonically at each change in assignment.

The mb-CW-PLS algorithm finds simultaneously an optimal partition of the fixed K micro-clusters (buckets) into $\mathcal{P} = \{P_1, ..., P_g, ..., P_G\}$, $P_g = \{C_k, \Phi(C_k) = g$ and $\forall \mathbf{x}_i \in C_k\ \phi(x_i) = k\}$ and the regression models associated to each macro-cluster g. Finally the best cross-validated try is selected based on the Root Mean Squared Error (RMSE) [20] score. This metric evaluate the prediction accuracy of the fitted regression,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$$

where $\hat{y}_i$ is the predicted value, y_i the true value and N the number of data points. This method is summarized in the Algorithm 1.

3.3. Implementations specificities

In order to get an efficient distributed implementations, we decided to apply the Clusterwise logic through Spark framework. As explained previously in order to apply a regression in the faster way, data should be inside the machine which perform linked operations. Then rather than to distribute piece of data through nodes in order to avoid shuffling, we put the entire dataset on each node using the *sc.broadcast(dataset)* function. Once this step done, we distribute simultaneously every initializations which correspond to $CV \times INIT$ homogeneously over nodes. It allows to each node to perform sequentially their $\frac{CV \times INIT}{nodes}$ Clusterwise initializations, via a *mapPartitions* functions where the number of Spark partitions is $CV \times INIT$. The *Map* part achieved we *aggregateByKey* obtained results where the *Key* is a Cross-Validation index. We decided to use the non-parametric k-nearest neighbors method to assign observations to their closest macro-cluster P_g in order to apply the associated PLS_g. Finally, based on RMSE scores, the best model is selected over Cross-Validations.

Algorithm 1 mb-CW-PLS Clusterwise algorithm

procedure CLUSTERWISE($\mathbf{X}, \mathbf{G}, \mathbf{CV}, \mathbf{INIT}$)
// Generate the cross validated dataset
$\cup_{i=1}^{\mathbf{CV}} \mathbf{X_i} = \mathbf{X}$
// These steps are distributed over nodes
for i := 1 to **CV do**
$\mathbf{X_{curr}} = \cup_{i \neq j}^{\mathbf{CV}} \mathbf{X_i}$
for j := 1 to **INIT do**
// Generate randomly filled classes
$\cup_{g=1}^{\mathbf{G}} \mathbf{X_g} = \mathbf{X_{curr}}$
Choose randomly x where $x \in \mathbf{X_{curr}}$
for u := 1 to **G do** Apply PLS_g on X_u with $\mathbf{x} \in X_u$
for u := 1 to **G do** Apply PLS_g on X_u with $\mathbf{x} \notin X_u$
Choose the class b with the best regression score (least error)
Add $\mathbf{x}$ to $\mathbf{X}_b$
Select the best initialization **b-init**
Apply corresponding model on the test set
Select the best model among **CV** through RMSE

Package availability. The mb-CW-PLS method described in this article are implemented in Spark/Scala and will be available on Clustering4Ever github repository at *https://github.com/Clustering4Ever/Clustering4Ever*

4. Numerical experiments

4.1. Prediction accuracy vs micro-cluster sizes

In contrast to the regular PLS regression, micro-batch Clusterwise PLS provides G regression models corresponding to G macro-clusters. The iterative cross validation process is as follows: macro clusters are initiated by assigning them randomly each point, for standard Clusterwise, or each micro-cluster C_k for micro-batch Clusterwise. Then the PLS is applied to each macro-cluster, once it is achieved, one observation or one micro-cluster C_k is moved inside all other macro-cluster P_g in order to compute new PLS regressions. Therefore we run one regression per macro-cluster with and without a specific observation or micro-cluster given $2 \times G$ regressions. Once least squares have been compared over all combinations, the point or the micro-cluster C_k is assigned to the macro-cluster P_g which yields the minimal error. For our experiments different datasets listed in Table 1 of various sizes have been used. Most of them came from UCI repository [11].

Dataset	n	$\mathbf{X}$ size, p	$\mathbf{Y}$ size, q
Yacht Hydrodynamics	308	6	1
Forest Fire	517	10	1
Wine Quality Red	1599	11	1
Wine Quality White	4898	11	1
SimData	400	10	3

Table 1: Overview of data sets.

The evolution of the RMSE in Table 3 with the value of micro-cluster size N_k shows that we can efficiently decrease the RMSE with our approach (mb-CW-PLS). We observe a property of Clusterwise methods that increasing number of macro-clusters G reduces the RMSE score. On the other hand, a lower G allows to use larger values of N_k. In Table 3, N/A entry indicates that we cannot execute the algorithms under these conditions due to risk of generate empty class which will falsify quality of results. First we seek for the optimal number of macro-clusters G, then we search for the optimal size of the micro-clusters (buckets) N_k. The first performance criterion is the Root Mean Square Error of prediction as evaluated with a ten-fold cross-validation procedure. Not surprisingly, mb-CW-PLS always improve the Root Mean Square Error (RMSE) of prediction while taking into account the cluster size N_k.

We observe a property of Clusterwise methods that an increasing number of macro-clusters G reduces the RMSE score. On the other hand, a lower G allows to use larger values of N_k. In Table 3, N/A entry indicates that the choice of tuning parameters does not lead to a well-defined solution e.g. if we set high value of N_k we could inconveniently moving every micro-cluster in a macro-cluster to others macro-cluster. However every macro-cluster have to be present in order to compare score of the different generated models. If this happened, then we re-tried with a different initialization, but a threshold of the number of attempts is fixed before considering tuning parameters ill-posed. The number of clusters K in the k-means clustering is $K = N/N_k$, the number of cross validation classes is $CV = 10$, the number of initializations is $INIT = 20$ and the number of k nearest neighbors for PLS model assignation is 20. Note that for $N_k = 1$ is equivalent to the standard CW-PLS.

Figure 2 illustrates results with the three output dimensions $\mathbf{Y}$ from SimData dataset. We observe that the predicted values (in red) are closer to the true values (in blue) for the mb-CW-PLS than with CW-PLS. The corresponding RMSE is referenced in Table 3.

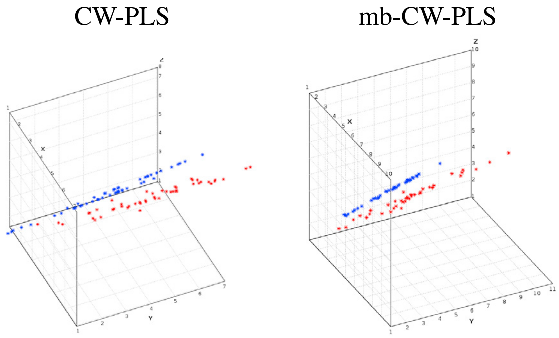


Fig. 2: Comparison of Clusterwise regression CW-PLS vs mb-CW-PLS for SimData 's $\mathbf{Y}$. True responses are in blue and predicted one are in red

4.2. Comparison to existing regression methods

To compare our PLS and mb-CW-PLS regressions to other regression methods, we took the experimental data sets with a single response variable ($q = 1$). We then fit models for the OLS, Ridge regression, LASSO and Random Forest using code from the Smile library (*https://haifengl.github.io/smile/*). 500 replicates of 90% random sampled are taken as training data, then the RMSE based on the remaining 10% test data is computed. The RMSEs are shown in Table 2. We observe that PLS provides at least as well the the OLS, Ridge regression and the LASSO, and less well than the Random Forest for the Forest Fire, Yacht Hydrodynamics and Wine Quality Red. The proposed mb-CW-PLS outperforms these methods, in some cases by a substantial margin, in terms of the RMSE. Furthermore, for the SimData which has 3 response variables ($q = 3$), the PLS and mb-CW-PLS are able to produce results, whereas the other regressions cannot provide results.

4.3. Comparison of prediction accuracy and execution times

Table 3 shows the RMSE scores and the execution times for CW-PLS and mb-CW-PLS. Let recall that CW-PLS is equivalent to mb-CW-PLS with $N_k = 1$. Figure 4a illustrates some of these results indicating that the higher we set G the better can be our results with a smaller standard deviation. This result is partially intuitive in that sense that the more specialized cluster we build, the more effective will be model build on them. The counter part of this strategy is a risk of overfitting. For most datasets, mb-CW-PLS presents similar RMSE scores to classical CW-PLS

Regression method	Forest Fire	Yacht Hydro-dynamics	Wine Quality Red	Sim Data
OLS	0.730 ±0.415	0.597 ±0.007	0.803 ±0.003	N/A
Ridge regression	0.729 ±0.415	0.596 ±0.007	0.803 ±0.003	N/A
LASSO	0.729 ±0.416	0.594 ±0.007	0.803 ±0.003	N/A
Random Forest	**0.715 ±0.362**	0.289 ±0.008	0.750 ±0.002	N/A
PLS	0.720 ±0.419	0.594 ±0.007	0.808 ±0.002	0.223 ±0.158
mb-CW-PLS $G = 6$	0.899 ±0.328	**0.226 ±0.039**	**0.003 ±0.000**	**0.016 ±0.000**

Table 2: Comparison of the different regression methods. The first row indicates the mean RMSE followed by the standard deviation. N/A entry indicates that we cannot execute the algorithms under these conditions

ones. Indeed execution times is much faster as Figure 3c highlight it, especially on bigger datasets. The micro-batch processing which produces the micro-clusters allows these decrease in execution time without sacrificing too much of the prediction accuracy as it is exposed on Figure 4b-4c. In some case as with the Wine Quality Red dataset we even observe better results. An interesting thing holds in the modest RMSE evolutions over different N_k values for $N_k > 1$.

4.4. Scalability

4.4.1. Scalability with the respect with the number of nodes

The results obtained in Table 3 were carried in a local environment (i.e. a stand-alone terminal). We now examine the performance of mb-CW-PLS in a true distributed computing set-up. To execute these experiments, we used Grid5000 [1] infrastructure which is one of the biggest French's laboratories clusters. We used a set-up with two times 8 core Intel Xeon E5-2630v3 or two times 12 cores AMD Opteron 6164 HE CPUs and 128 Gb RAM per node. Table 4 presents average time spend for four run executing a full Clusterwise workflow, which consist to train and test models to select best one. Total init $CV \times Init$ corresponds to the total number of initializations made. Figure 3a shows that the execution times decreases efficiently with the number of nodes.

In order to maximize efficiency, we recommend to have 2-3 times more Spark partitions than the total number of core among every nodes. This approach is not optimal if we have a large data set with few initializations as we do not utilize fully the distributed computational power of Spark. We can also observe that when the number of core nodes is exceeded by the number of initializations, the growth in execution time is almost linear with respect with the number of initializations. As our distributed set-up consists of nodes with 32 cores, a reduction in execution time is observed when the number of initializations is higher than 32.

4.4.2. Scalability with respect with the number of initializations

Figure 3b shows the linearity of the problem with the increase of the number of initializations with a set-up of 8 slave nodes. This results illustrate that initializations are well distributed among nodes which allows an efficient computations of the algorithm.

5. Discussions and conclusion

This work present a new Clusterwise PLS regression algorithm, which brings multiple improvements. First, the micro-batch processing facilitates a drastic reduction in execution times, keeping the same magnitude of prediction accuracy. Second, the distributed implementation enables the test with a large number of initializations in order to

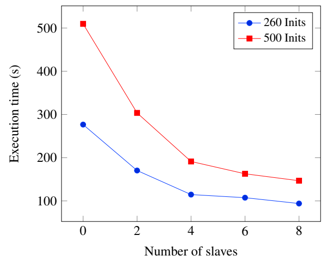
Dataset	CW-PLS	mb-CW-PLS $N_k = 5$	mb-CW-PLS $N_k = 10$	mb-CW-PLS $N_k = custom1$	mb-CW-PLS $N_k = custom2$
Forest Fire $G = 4$ 6 slaves INTEL	**0.06625** ±0.0132 26.3s	0.07771 ±0.00843 16.6s	0.08627 ±0.00952 15.6s	0.0857 ±0.00304 14.9s $N_k = 15$	0.08577 ±0.00728 15.1s $N_k = 20$
Yacht Hydrodynamics $G = 2$ 6 slaves INTEL	**0.0101** ±4.0E-4 15.6s	0.01852 ±3.7E-4 13.2s	0.01888 ±0.00138 12.8s	0.01821 ±0.00103 13.3s $N_k = 15$	0.01899 ±0.00182 13.2s $N_k = 20$
Yacht Hydrodynamics $G = 4$ 6 slaves INTEL	**0.01066** ±4.1E-4 15.0s	0.02002 ±8.2E-4 13.1s	0.01815 ±0.00146 13.1s	0.0167 ±8.0E-4 12.9s $N_k = 15$	0.01435 ±7.9E-4 12.8s $N_k = 20$
Yacht Hydrodynamics $G = 6$ 8 slaves AMD	**0.00146** ±5.0E-5 32.5s	0.01227 ±8.9E-4 28.2s	0.01233 ±6.7E-4 29.2s	N/A $N_k = 15$	N/A
Wine Quality Red $G = 4$ 8 slaves AMD	0.01269 ±4.8E-4 522.1s	**0.00969** ±1.0E-4 **141.6s**	0.00982 ±1.0E-5 **93.5s**	0.00989 ±7.0E-5 **63.1s** $N_k = 20$	0.00971 ±8.0E-5 **50.2s** $N_k = 40$
Wine Quality White $G = 4$ 8 slaves AMD	Not finished after 5h of computation	0.05567 ±3.4E-4 3878.7s	0.05545 ±4.0E-5 2032.0s	0.05489 ±3.9E-4 1059.2s $N_k = 20$	**0.05481** ±2.5E-4 **587.3s** $N_k = 40$
Sim Data $G = 2$ 8 slaves AMD	**0.02203** ±5.2E-4 40.5s	0.10793 ±0.00558 30.1s	0.10281 ±0.01839 28.7s	0.10541 ±0.00389 28.2 $N_k = 15$	N/A
Sim Data $G = 3$ 8 slaves AMD	**0.02383** ±0.00105 38.3s	0.09959 ±0.00473 30.0s	0.09464 ±0.00611 29.8s	0.09479 ±0.00274 28.7s $N_k = 15$	N/A
Sim Data $G = 4$ 8 slaves AMD	**0.02723** ±0.00156 38.2s	0.10054 ±0.00688 30.0s	0.08356 ±0.00497 29.3s	0.09095 ±0.00731 30.0s $N_k = 15$	N/A
Sim Data $G = 5$ 8 slaves AMD	**0.02951** ±0.00233 42.7s	0.0909 ±0.00376 30.5s	0.0894 ±0.00572 30.5s	0.0707 ±0.00606 31.7 $N_k = 15$	N/A
Sim Data $G = 6$ 8 slaves AMD	**0.02317** ±8.4E-4 36.0s	0.09274 ±0.0058 29.7s	0.07298 ±0.00552 32.0s	0.07436 ±0.00828 36.5 $N_k = 15$	N/A

Table 3: Comparison of CW-PLS and mb-CW-PLS. The first row is the mean RMSE followed by the standard deviations and the execution times. N/A indicates that we cannot execute the algorithms

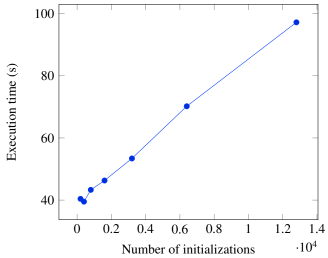
# total init	local	2 slaves	4 slaves	6 slaves	8 slaves
260	276.6	170.3	114.6	107.4	93.9
500	509.7	303.6	191.2	162.7	146.7

Table 4: Comparison of execution times with different number of initializations and slave processes for the Yacht Hydrodynamics using Intel setup.

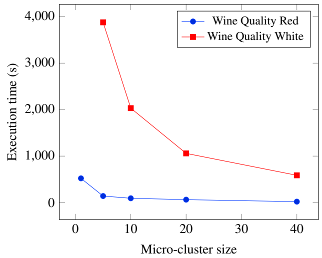
find the optimal cross-validated model. One key aspect of the algorithm is the number of regressions needed for one complete run which is $2 \times G \times CV \times INIT \times N \approx 1000 \times N$ leading to a quadratic time complexity. Our micro-cluster



(a) Evolution of the execution times with respect with the number of slaves. Set-up with Intel CPUs

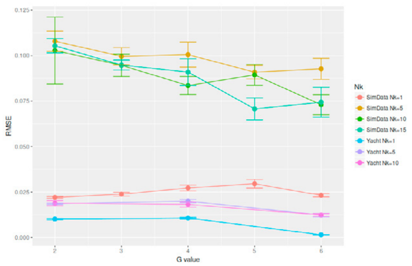


(b) Evolution of the execution times with respect with the number of initializations on Yacht dataset. Set-up with 8 AMD slaves

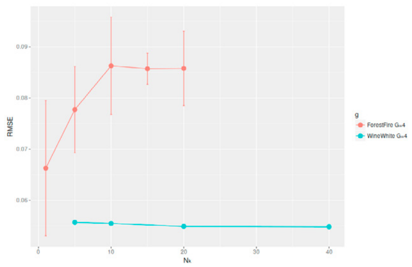


(c) Evolution of the execution times with respect with of the size of micro-cluster for different datatsets

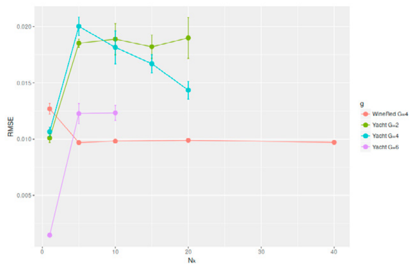
Fig. 3: Evolution of the execution times



(a) RMSE with standard deviation per G value



(b) RMSE with standard deviation per micro-cluster size



(c) RMSE with standard deviation per micro-cluster size

Fig. 4: Root Mean Square Error (RMSE)

approach reduce exclusively the number of regression from N to $\frac{N}{N_k}$, but we still compute PLS on the whole dataset preventing better time performance. In our future works we will examine methods to decrease the execution time and also to increase prediction quality. Unfortunately, an inevitable bottleneck is the impossibility to decrease the inner complexity of a regression and Clusterwise strategy is very regression greedy. One solution we desire to investigate will be to reduce to a computable size datasets keeping their inner structure in order to train the Clusterwise model. We have engage this work but due to number of existing sketching methods we need to further testing to find an optimal solution. Another important aspect holds in the affectation of a new observation to its closest macro-cluster which is solved here by the k-nearest neighbors. However others methods can be tested as for example comparison between the observation with macro-clusters centroid which is faster than k-nearest neighbors but potentially less accurate. As explained previously, an important number of initializations is required to guaranty quality due to their random nature. One option we are thinking about is relaunch an initialization over an achieved one in order to benefit from the Clusterwise process. We can also relaunch this process multiple times until we converge toward an optimal score. The computation overhead could be decrease by reducing the number of random initializations. Another quality result issue is linked to micro-clusters we generate, they came from a basic k-means version ,using an euclidean dissimilarity, in order to formed our micro-clusters, others clustering techniques could be used and compared to discover which one fit the best a specific need.

References

[1] Balouek, D., Carpen Amarie, A., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., Lèbre, A., Margery, D., Niclausse, N., Nussbaum, L., Richard, O., Pérez, C., Quesnel, F., Rohr, C., Sarzyniec, L., 2013. Adding virtualization capabilities to the Grid'5000 testbed, in: Ivanov, I.I.,

van Sinderen, M., Leymann, F., Shan, T. (Eds.), Cloud Computing and Services Science. Springer International Publishing. volume 367 of *Communications in Computer and Information Science*, pp. 3–20. doi:10.1007/978-3-319-04519-1_1.

[2] Bock, H., 1969. The equivalence of two extremal problems and its application to the iterative classification of multivariate data. Mathematisches Forschungsinstitut .

[3] Bougeard, S., Abdi, H., Saporta, G., Niang, N., 2017. Clusterwise analysis for multiblock component methods. Advances in Data Analysis and Classification , 1–29.

[4] Charles, C., 1977. Régression typologique et reconnaissance des formes. Ph.D. thesis. Université Paris IX.

[5] De Roover, K., Ceulemans, E., Timmerman, M.E., Vansteelandt, K., Stouten, J., Onghena, P., 2012. Clusterwise simultaneous component analysis for analyzing structural differences in multivariate multiblock data. Psychological Methods 17, 100.

[6] DeSarbo, W.S., Cron, W.L., 1988. A maximum likelihood methodology for clusterwise linear regression. Journal of classification 5, 249–282.

[7] Diday, E., 1976. Classification et sélection de paramètres sous contraintes. Rapport de recherche IRIA-LABORIA .

[8] Esposito Vinzi, V., Trinchera, L., Squillacciotti, S., Tenenhaus, M., 2008. Rebus-pls: A response-based procedure for detecting unit segments in pls path modelling. Applied Stochastic Models in Business and Industry 24, 439–458.

[9] Hahn, C., Johnson, M.D., Herrmann, A., Huber, F., 2002. Capturing customer heterogeneity using a finite mixture pls approach. Schmalenbach Business Review 54, 243–269.

[10] Hwang, H., Desarbo, W.S., Takane, Y., 2007. Fuzzy clusterwise generalized structured component analysis. Psychometrika 72, 181.

[11] Lichman, M., 2013. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

[12] Lohmöller, J.B., 2013. Latent variable path modeling with partial least squares. Springer Science & Business Media.

[13] Martella, F., Vicari, D., Vichi, M., 2015. Partitioning predictors in multivariate regression models. Statistics and Computing 25, 261–272.

[14] Montgomery, D.C., Peck, E.A., Vining, G.G., 2001. Introduction to linear regression analysis. .

[15] Preda, C., Saporta, G., 2005. Clusterwise pls regression on a stochastic process. Computational Statistics & Data Analysis 49, 99–108.

[16] Sarstedt, M., 2008. A review of recent approaches for capturing heterogeneity in partial least squares path modelling. Journal of Modelling in Management 3, 140–161.

[17] Schlittgen, R., Ringle, C.M., Sarstedt, M., Becker, J.M., 2016. Segmentation of pls path models by iterative reweighted regressions. Journal of Business Research 69, 4583–4592.

[18] Späth, H., 1979. Algorithm 39 clusterwise linear regression. Computing 22, 367–373.

[19] Vinzi, V.E., Lauro, C.N., Amato, S., 2005. Pls typological regression: algorithmic, classification and validation issues, in: New developments in classification and data analysis. Springer, pp. 133–140.

[20] Willmott, C.J., Matsuura, K., 2005. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate research 30, 79–82.

[21] Wold, H., 1985. Partial least squares. Encyclopedia of statistical sciences .