PPLS: An efficient parallel algorithm for Partial Least-Squares

Ruy Luiz Milidiú[1]
Raúl Rentería[1]

[1] Departamento de Informática
Pontifícia Universidade Católica PUC-Rio
Rua Marquês de São Vicente 225, Gávea, CEP 22453-900
Rio de Janeiro — Brasil
milidiu@inf.puc□rio.br
renteria@inf.puc□rio.br

Abstract—
PPLS, a parallel version for the Partial Least-Squares algorithm, is introduced in this article. The proposed algorithm is restricted to the case of only one dependent variable for the regression model. Unlike other approaches, such as the ones that calculate the PLS factors through Hebbian learning, PPLS is exact and does not depend on approximation or convergence criteria. In our experiments with a small data set, the algorithm shows a Speedup greater than 3 for the first 4 machines in a computer cluster architecture.

Keywords— **Partial Least-Squares, PLS, parallelism, chemometrics, factor analysis.**

I. Introduction

The PLS algorithm [GK86] has been widely used as a chemometric tool for Near-Infrared spectral analysis [HT88a, HT88b, BK87]. The simplicity of the technique and robustness of the generated model, also make the partial least-squares approach a powerful tool for factor analysis, being applied to many other areas such as process monitoring, marketing analysis and image processing [AM99, MMR99]. In this paper, we propose PPLS, a parallel version of the Partial Least-Squares algorithm restricted to one dependent variable. Compared to other parallel implementations [HK98, MMT94, HM98] that use Neural Networks, ours is not based on any convergence rule such as Hebbian learning, making its use simpler.

In our experiments with a small data set, the PPLS approach shows an efficiency above 74% when using four nodes of our computer cluster.

In section 2, we present the classical non-parallel PLS algorithm. In section 3, our parallel approach PPLS is presented, and, in section 4, its parallel performance is tested.

II. PLS Algorithm

A. Modeling

Partial Least Squares (PLS) is a multivariate statistical method, based on the use of factors, which is aimed at prediction [GK86]. The goal is to predict the values of a set of variables y based on the observed values of a set of variables x. As an example, x may be formed by the values of a time series window and y be taken as the value of a single future observation, or as the values of a set of future points in the same time series.

The construction of a PLS model requires a set of observation samples (patterns) and also their respective future values. Let X be the matrix containing in its rows the patterns of observations and Y be the matrix containing in its rows the effective values to be predicted.

The PLS method is a modeling procedure that simultaneously estimates underlying factors in both X and Y. These factors are then used to define a subspace in X that is more adequate to model Y. With PCR [BK87], the rotation defined by the eigenvectors is used to find a subspace in X that subsequently is used to model Y. The approach taken by PLS is very similar to that of Principal Component Analysis (PCA) [LMP97], except that factors are chosen to describe the variables in Y as well as in X. This is accomplished by using the columns of the Y matrix to estimate the factors for X. At the same time, the columns of X are used to estimate the factors of Y. The resulting models are

$$X = TP + E$$

$$Y = UQ + F$$

where the elements of T and U are called the scores of X and Y, respectively, and the elements of P and Q are called the loadings. The matrices E and F are the errors associated with modeling X and Y with the PLS model.

The T factors are not optimal for estimating the columns of X as is the case with PCA , but are rotated so as to simultaneously describe the Y matrix.

In the ideal situation, the sources of variation in X are exactly equal to the sources of variation in Y, and the factors for X and Y are identical. In real applications, X varies in ways not correlated to the variation in Y, and therefore $t \neq u$

Let X and Y be the following matrices

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} \text{ and } Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

At each one of the γ nodes, n/γ samples are stored after partitioning the original X and Y matrices into the following blocks

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_\gamma \end{bmatrix} \text{ and } Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_\gamma \end{bmatrix}$$

where blocks X_γ and Y_γ go to node γ.

The calculation of each set of factors $\{w, b, p\}$ can be acomplished by combining the two algorithms presented in figures 3 and 4. They describe the computation to be performed in nodes 1 to γ and node 0 respectively.

```
Parallel PLS for node $i$

$R_i = X_i^\top Y$
send $R_i$ to node 0

receive $w$ from node 0
$t_i = X_i w$
$b_i = t_i^\top Y_i$
$d_i = t_i^\top t_i$
send $b_i, d_i$ to node 0

$p_i = X_i^\top t_i$
send $p_i$ to node 0

// Calculation of the residuals
receive p from node 0
receive b from node 0
$X_i = X_i - t_i p^\top$
$Y_i = Y_i - b \cdot t_i$
```

Fig. 3. $\{w, b, p\}$ calculation for node i, for $i = 1, \ldots, \gamma$

After each iteration through the codes just introduced, the top node 0 will have the set $\{w, b, p\}$ corresponding to the regression for the first factor. To calculate the next one, it is only necessary to execute the same steps again.

IV. Performance Analysis

For the performance analysis of PPLS, two metrics are used [CG91, Fos95, JáJ92]: *Speedup* and *Efficiency*. The

```
Parallel PLS for top node 0

for $i = 1$ to $\gamma$
    receive $R_i$ from node $i$
$R = \sum_{i=1}^{\gamma} R_i$
$w = R^\top / ||R||$

for $i = 1$ to $\gamma$
    send $w$ to node $i$
for $i = 1$ to $\gamma$
    receive $b_i$ from node $i$
    receive $d_i$ from node $i$
$b = \sum_{i=1}^{\gamma} b_i / \sum_{i=1}^{\gamma} d_i$

for $i = 1$ to $\gamma$
    receive $p_i$ from node $i$
$p = \sum_{i=1}^{\gamma} p_i$
$p = p/||p||$

// Calculation of the residuals
for $i = 1$ to $\gamma$
    send p to node $i$
    send b to node $i$
```

Fig. 4. $\{w, b, p\}$ calculation for node 0

first one indicates the parallel computation gain obtained at each node added for the model calibration step. The second metric gives an idea of how much each node contributes to the global processing.

During the tests made to estimate the *Speedup* and *Efficiency* of PPLS, we have used up to 22 computers of a cluster composed of IBM 400MHz PC's with 32MB of RAM each one. All the computers were conected through a 10MB/s network switch. The algorithm was coded in C using the mpich v1.2.0 MPI library [Int] for the logical communication between the nodes. Regarding the *Speedup* estimation, the computation required by node 0 was performed by one of the γ processors available. Also, for the comparison to the case of one processor we used a competitive sequential algorithm, restricted to the case of only one variable.

For our experiments, the first 5340 observations of the data set D provided by the Santa Fé Competition [Ins92] was employed. This data was synthetically generated by numerically integrating the motion equations for a damped driven particle. For the construction of the X matrix we have used a sliding window with a 300 points width over the series. The next point, rigth after the window, was set as the Y to be predicted. In that way we are trying to predict a future value of

(t and u are columns of T and U, respectively). However, when both matrices are used to estimate factors, the factors for the X and Y matrices have the following relationship:

$$u = bt + \epsilon$$

where b is termed the inner relationship between u and t and is used to calculate subsequent factors if the intrinsic dimensionality of X is greater than one. Geometrically, this states that the vectors u and t are approximately equal except for their lengths.

The main advantage of PLS is that it incorporates more information in the model-building phase.

B. The Algorithm

The classical Partial Least-Squares algorithm, as described in [GK86], can be decomposed in the following steps:

1. given a data set for training, a regression model is built. This is the calibration or training step;
2. given an independent data set, called test set, predictions are made using the model that has just been built.

In the following subsections these two steps are explained.

B.1 Model Training

The sequential PLS algorithm, as described in figure 1, uses as the training set the $n \times m$ matrix X, and the n dimensional vector Y. Observe that X contains the n observations of m independent variables. Y contains the corresponding values for the dependent variable.

At each iteration, the following factors are calculated regarding the regression model between the scores for X and Y:

1. the weights w_i;
2. the regression coefficient b_i for the inner relation;
3. the loadings represented by p_i.

As already stated, there is a key difference between PLS and other regression methods [GK86] such as PCR. Both methods construct a regression on principal components, however the model constructed by PLS also uses information from the dependent variable Y to bias the principal components. In fact, as we can see in the second and third lines of figure 1, the weighting factor w_i being an eigenvector of $X^\top Y Y^\top X$, provides a better quality for the prediction step.

B.2 Prediction Step

Given a trained model, obtained as described in the previous section, one can make predictions by using an independent data set X. Figure 2 shows the algorithm for this step. It should be noticed that the number of desired factors for the prediction is indicated by the variable k.

A common procedure, when determining the optimal number of factors k to be used in the prediction, consists in calculating a statistic for the lack of model accuracy called PRESS [GK86] (Prediction Residual Sum of Squares). This kind of method, called cross-validation [BK87], uses an independent data set X with an already known variable Y. PRESS is calculated for each value of k, and the one that yields the minimum PRESS indicates the recommended number of factors to be chosen.

```
Sequential PLS
for  i = 1 to m
       $R = Y^\top X$
       $w_i = R^\top / ||R||$

       $t_i = X w_i$
       $b_i = t_i^\top Y / t_i^\top t_i$

       $p_i = X^\top t_i$
       $p_i = p_i / ||p_i||$

       $X = X - t_i p_i^\top$
       $Y = Y - b_i \cdot t_i$
end
```

Fig. 1. Sequential PLS algorithm.

```
PLS Prediction
for  i = 1 to k
       $t_i = X w_i$
       $y = y + b_i \cdot t_i$
       $X = X - t_i p_i^\top$
end
```

Fig. 2. Sequential PLS algorithm for the prediction of y.

III. PPLS: Parallel PLS Algorithm

Since the calibration step dominates the time consumption of the Partial Least-Squares methodoly, only the parallelization effort of this step will be detailed here. Furthermore, the data sets usually involved in the prediction step are very small when compared with the training step. Hence, the sequential algorithm can be used with no significant loss of performance.

Given $\gamma+1$ machines, denoted as nodes 0 to γ, and n samples (x_i, y_i) $(1 \leq i \leq n)$, with $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$, we want to build a linear model for regression of Y upon X. Node 0 is a machine with a distinctive function: it coordinates and integrates the computation performed by machines 1 to γ.

the series given its past 300 obervations, using for the model calibration step the 5040×300 matrix X and the 5040×1 vector Y. Figure 5 shows an overview of the behaviour of the D time series.

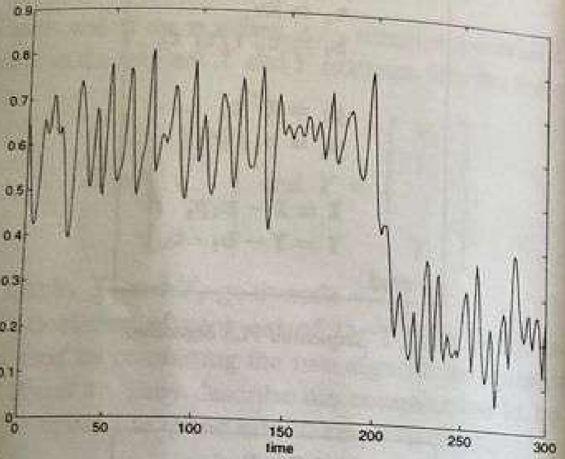

Fig. 5. Part of the data series D using for experiments.

TABLE I
PPLS PERFORMANCE

Nodes	Time in sec.	Speedup	Efficieny
1	111.778	1.000	1.000
2	60.545	1.846	0.923
3	44.646	2.503	0.834
4	37.312	2.995	0.748
5	33.410	3.345	0.669
6	31.038	3.601	0.600
7	29.773	3.754	0.536
8	29.395	3.802	0.475
9	32.852	3.402	0.378
10	38.129	2.931	0.293
12	43.800	2.552	0.232
14	51.025	2.190	0.182
15	55.143	2.027	0.155
18	67.158	1.664	0.118
20	75.224	1.485	0.099
21	79.316	1.409	0.088

Table I shows the performance of PPLS regarding time, speedup and efficiency when calculating all the 300 factors. The same measures are plotted on figures 6, 7 and 8 for a better analysis.

As we can see in figure 7, our algorithm showed a Speedup greater than 3 for 4 nodes. In addition, we obtained an Efficiency above 74% for the same configuration, indicating that our parallel aproach is efficient.

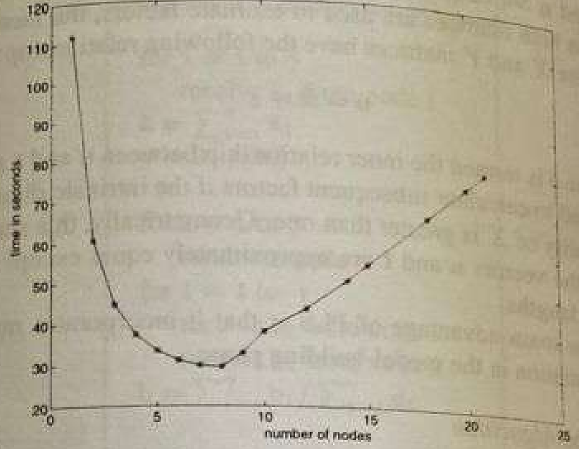

Fig. 6. Running time of PPLS.

It should be noted that these results depend on the size of the data set used. In fact, using a bigger set will give a better performance for the algorithm as the communication between the nodes do not depends on the amount of samples (or the number of lines of X) on each one.

V. CONCLUSION

In this work, we presented PPLS, a parallel version of the Partial Least-Squares algorithm. The examined model is restricted to only one dependent variable. Nevertheless, it can be used in many practical applications [MMR99, HK98]. Among the main caracteristics of PPLS we can highlight the following:

1. the calculation of the PLS factors does not depend on any kind of convergence criteria;
2. the running time of PPLS regarding the communication beteween the nodes does not depend on the number of samples used, making the algorithm suitable for huges data sets.

Besides, our experiments show a good performance, since the observed efficiency is above 74%, when running PPLS in 4 nodes.

The simplicity and low cost of this distributed version of PPLS recomend its use in production environnments.

REFERENCES

[AM99] Michel Tenenhaus Alain Morineau, editor. *Les Méthodes PLS. Symposium International PLS'99*. Cisia-Ceresta, October 1999.

[BK87] Kenneth R. Beebe and Bruce R. Kowalski. An introduction to multivariate calibration and analysis. *Analytical Chemistry*, 59(17):1007–1017, September 1987.

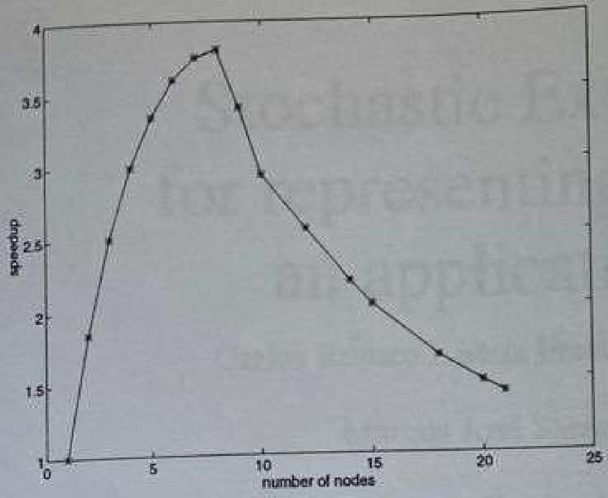

Fig. 7. Speedup for PPLS.

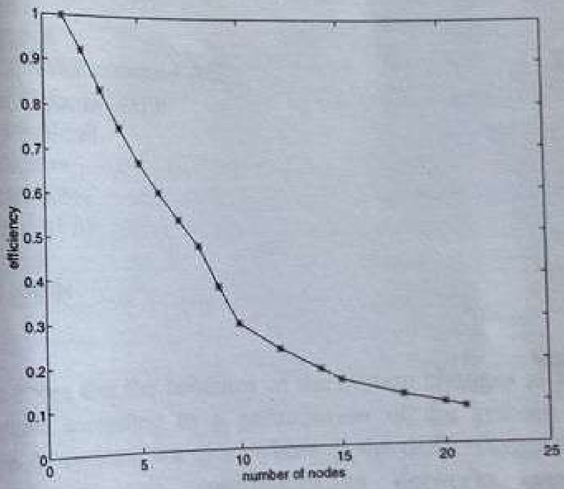

Fig. 8. Efficiency for PPLS.

[CG91] Nicholas Carriero and David Gelernter. *How to Write Parallel Programs: a First Course*. MIT Press, Cambridge, 1991.

[Fos95] Ian Foster. *Designing and Building Parallel Programs*. Addison-Wesley, Cambridge, 1995.

[GK86] Paul Geladi and Bruce R. Kowalski. Partial least squares regression: A tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

[HK98] Fredric M. Ham and Ivica Kostanic. A neural network architecture for partial least-squares regression (plsnet) with supervised adaptive modular hebbian learning. *Neural, Parallel & Scientific Computations*, 6:35–72, 1998.

[HM98] Fredric M. Ham and Thomas M. McDowall. Inverse model formulation of partial least-squares regression: a robust neural network approach. In Steven K. Rogers, David B. Fogel, and James C. Bezdekand Bruno Bosacchi, editors, *Applications and Science of Computational Intelligence*, volume 3390, pages 36–47. SPIE, March 1998.

[HT88a] David M. Haaland and Edward V. Thomas. Partial least-squares methods for spectral analyses. 1. relation to other quantitative calibration methods and the extraction of qualitative information. *Analytical Chemistry*, 60(11):1193–1202, June 1988.

[HT88b] David M. Haaland and Edward V. Thomas. Partial least-squares methods for spectral analyses. 2. application to simulated and glass spectral data. *Analytical Chemistry*, 60(11):1202–1208, June 1988.

[Ins92] Santa Fé Institute. ftp://ftp.santafe.edu/pub/time-series/ competition/, 1992.

[Int] Message Passing Interface. http://www-unix.mcs.anl.gov/mpi/.

[JáJ92] Joseph JáJá. *An Introduction to Parallel Algorithms*. Addison-Wesley Publishing Company, November 1992.

[LMP97] Ludovic Lebart, Alain Morineau, and Marie Piron. *Statistique Exploratoire Multidimensionnelle*. Dunod, Paris, 1997.

[MMR99] Ruy Milidiú, Ricardo Machado, and Raúl Rentería. Time series forecasting throught wavelets and a mixture of experts models. *Neurocomputing*, 28:145–156, 1999.

[MMT94] E. Malthouse, R. Mah, and A. Tamhane. Nonlinear partial least squares using neural networks. INCINC94 Chemometrics Conference., 1994.