Big data in multi-block data analysis: An approach to parallelizing Partial Least Squares Mode B algorithm

Alba Martinez-Ruiz a,*, Cristina Montañola-Sales b,c

a Universidad Católica de la Santísima Concepción, Alonso de Ribera 2850, Concepción, Chile
b IQS-Universitat Ramon Llull (URL), Via Augusta, 350, 08017 Barcelona, Spain
c Barcelona Supercomputing Center, Centro Nacional de Supercomputación (BSC-CNS), Jordi Girona 29, 08034, Barcelona, Spain

* Corresponding author.
E-mail address: alba.martinez.ruiz@gmail.com (A. Martinez-Ruiz).

Abstract

Partial Least Squares (PLS) Mode B is a multi-block method and a tightly coupled algorithm for estimating structural equation models (SEMs). Describing key aspects of parallel computing, we approach the parallelization of the PLS Mode B algorithm to operate on large distributed data. We show the scalability and performance of the algorithm at a very fine-grained level thanks to the versatility of pbdR, a R-project library for parallel computing. We vary several factors under different data distribution schemes in a supercomputing environment. Shorter elapsed times are obtained for the square-blocking factor $16 \times 16$ using a grid of processors as square as possible and non-square blocking factors $1000 \times 4$ and $10000 \times 4$ using an one-column grid of processors. Depending on the configuration, distributing data in a larger number of cores allows reaching speedups of up to 121 over the CPU implementation. Moreover, we show that SEMs can be estimated with big data sets using current state-of-the-art algorithms for multi-block data analysis.

Keywords: Computer science, Computational mathematics
1. Introduction

Early, mathematicians and computer scientists explored methodologies and proposed techniques to process distributed matrices to optimize computing power and profit large computer systems (Golub and Van Loan, 1996). With the software-hardware infrastructures advances, examining large data sets is gradually more feasible. That is the reason why the use of parallel computing technologies has spread by leaps and bounds in many areas (Schmidberger et al., 2009, Pacheco, 2011). This situation makes the analysis of big volumes of data a major challenge of investigating the performance, efficiency, and effectiveness of statistical methods.

From an end-user perspective, the parallelization process of an algorithm is not an easy task. It requires considering many factors, such as data distribution and data processing schema, the understanding of how available computer architectures operate to find the best way to distribute both data and tasks, or determining the appropriate dimension of data blocks for distributing data. As a result, scientific communities and companies are making available computational platforms for parallel statistical analysis, parallel computing and big data endeavors with increasing swiftness. An example of that is the website of “CRAN Task View: High-Performance and Parallel Computing with R” (Eddelbuettel, 2016) which lists a set of R packages and tools to develop parallel R-based applications, the preferred software of the statistical community. The number of applications in the list has at least doubled in the last few years. Most of them provide support to MPI (Message Passing Interface) API which is the standard in parallel computing.

Among the different existing tools (Schmidberger et al., 2009) we would like to highlight snow (Rossini et al., 2007, Tierney et al., 2011), snowfall (Knaus, 2010), parallel (included in R since R 2.14.0) and its extension doParallel (Calaway et al., 2015), Rmpi (Yu, 2009), pbdR and MapReduce. snow and snowfall rely in the typical task parallelism provided by libraries with a Master/Worker approach. They use one function to perform reductions on a whole distributed data set in parallel. Both tools have been used in several applications. For instance, Deb and Srirama (2013) used snow to process bigger gene expression data sets by parallelizing the algorithm of K-Means clustering exploiting the multicore architecture of a desktop computer and Riddick et al. (2011) took advantage of snow package to make more efficient the process of multiple drug responses using Random Forest.

In contrast with this approach, Rmpi exposes MPI routines in R but leaves the parallelization task to the user. In this way, McLeod et al. (2007) used Rmpi to reduce computations by a factor of 30 in running the Durbin-Levinson and Trench algorithms for linear time series analysis and Lê Cao and Chabrier (2008) used Rmpi to faster the classification process of high dimensional data sets. Another example is Varsos et al. (2016), who took advantage of Rmpi to implement single program