Computing (2017) 99:219–234
DOI 10.1007/s00607-016-0487-8

Parallel regressions for variable selection using GPU

Lauro Cássio Martins de Paula[1] · Anderson S. Soares[1] · Telma W. L. Soares[1] · Arlindo R. G. Filho[1] · Clarimar J. Coelho[1] · Alexandre C. B. Delbem[1] · Wellington S. Martins[1]

Received: 20 March 2015 / Accepted: 3 February 2016 / Published online: 17 February 2016

Abstract This paper proposes a parallel regression formulation to reduce the computational time of variable selection algorithms. The proposed strategy can be used for several forward algorithms in order to select uncorrelated variables that contribute for a better predictive capability of the model. Our demonstration of the proposed method include the use of Successive Projections Algorithm (SPA), which is an iterative forward technique that minimizes multicollinearity. SPA is traditionally used for variable selection in the context of multivariate calibration. Nevertheless, due to the need of calculating an inverse matrix for each insertion of a new variable in the model calibration, the computational performance of the algorithm may become impractical as the matrix size increases. Based on such limitation, this paper proposes a new strategy called Parallel Regressions (PR). PR strategy was implemented in the SPA to avoid the matrix inverse calculation of original SPA in order to increase the computational performance of the algorithm. It uses a parallel computing platform called Compute Unified Device Architecture (CUDA) in order to exploit a Graphics Processing Unit, and was called SPA-PR-CUDA. For this purpose, we used a case study involving a large data set of spectral variables. The results obtained with SPA-PR-CUDA presented 37× times better performance compared to a traditional SPA implementation. Additionally, when compared to traditional algorithms we demonstrated that SPA-PR-CUDA may be a more viable choice for obtaining a model with a reduced prediction error value.

Keywords Multivariate calibration · Variable selection · GPU · SPA

✉ Anderson S. Soares
anderson@inf.ufg.br

Lauro Cássio Martins de Paula
laurocassio@inf.ufg.br

[1] Federal University of Goiás, Goiânia, Goiás, Brazil

Mathematics Subject Classification 68W10

1 Introduction

The connection of laboratory instruments with computers have produced a large amount of complex data [1,2]. Increasingly huge databases containing a large amount of information such as the spectral analysis, chromatographic, molecular structures, and their properties are available nowadays [3]. Mathematical models can be used to extract the relation between independent and response variables from a large volume of data.

Regression is a technique that relates dependent variables (response variables) with independent variables (explanatory variables). The most widely used estimation method of the relationship between variables is the Least Squares method [4]. The Least Squares or Ordinary Least Squares (OLS) method can find the best fit for a data set by minimizing the sum of squared differences between the estimated response value and the observed data (such differences are called residue), estimating the unknown parameters in a Multivariate Regression Model (MLR).

The major limitation in regression is the presence of collinear variables. This problem is known as multicollinearity [5], which results in a ill-conditioned regression. MLR and other multivariate regression methods can load a large amount of data from a database into large matrices in a computer memory. Such matrices usually contain many more columns than rows [6,7]. It should be remarked that variables in such matrices are often very correlated in which two or more variable or measurements on the same group of elements vary together [8]. Furthermore, the inversion of the matrix of variables is mathematically unstable when the variables are highly correlated [5]. In this scenario, there are at least two problems to be overcome: (i) the inversion of the highly correlated matrix; and (ii) the computational time spent to inversion of large matrices. Both limitations of matrix inversion can be solved using variable (or feature) selection methods, which helps to reduce the correlation among the variables.

Variable selection algorithms have been used in order to avoid the ill-conditioning regression [9]. Search algorithms such as Genetic Algorithms (GA) and the Successive Projections Algorithm (SPA) have also been used for variable selection in the regression [10,11]. For instance, Paula et al. [12] proposed a modified metaheuristic (Firefly Algorithm) for variable selection in a multivariate calibration problem. Furthermore, Soares et al. [13] presented an optimized SPA implementation for variable selection in Near-Infrared spectrometric analysis. However, in all cases the high computational cost of the search grows considerably as the number of variables increases [11].

Although our proposition can be used for any “forward” algorithm in order to select uncorrelated variables that contribute for a better predictive capability of the model, we choose the SPA[1] algorithm to demonstrate the advantages of our proposition.

Successive Projections Algorithm is an iterative “forward” technique used for variable selection and multicollinearity reduction in different applications such as

[1] SPA is a variable-selection technique that has attracted increasing interest in the community in the past 10 years [14].

multivariate calibration [14,15] and multivariate classification [16–18]. Indeed, SPA is a successful technique for variable selection in multiple applications [11,13,19]. Nevertheless, its major limitation consists on the calculation of the inverse matrix every time a new variable is inserted into the subset of candidate variables [19].

Because of such limitation, this paper proposes a new strategy called Parallel Regressions (PR), which seems very useful since it can measure the effect of a variable inclusion into a model without calculating an inverse matrix. The use of PR also can produce some reduction of the computational time calculations. On the other hand, the inclusion of variables has the effect of increasing the running time.

Therefore, a new parallel implementation of the SPA is also proposed. This new version is implemented exploiting features of Compute Unified Device Architecture (CUDA) on a Graphics Processing Unit (GPU), and it was called SPA-PR-CUDA. In addition, SPA-PR-CUDA includes a code optimization in the mathematical formulation. The results showed a significant reduction of the computational time compared to original SPA. It was possible to note that SPA-PR-CUDA may be twelve times faster than traditional SPA implementations. It is important to note that all outcomes were obtained by averaging twenty executions.

The study case includes the variable selection obtained from NIR (Near-InfraRed) spectrum for determination of protein concentration on wheat. In the original data set, 1090 correlated variables are available. The challenge is to select a minimal subset uncorrelated that minimizes the error between predicted and real protein concentration on wheat.

The paper is organized as follows. Section 2 describes the SPA. The proposed algorithm and parallelization are detailed in Section 3. Section 4 describes the materials and methods used in the experiment. The results are discussed in Sect. 5. Finally, Sect. 6 shows the conclusions.

2 SPA and MLR review

2.1 MLR

The multivariate calibration refers to obtaining a mathematical model that allows to provide the value of a quantity y based on values measured from a set of explanatory variables $x_1, x_2, \ldots, x_k$ [9]. In other words, it is possible to obtain a suitable model

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon, \tag{1}$$

where $\beta_0, \beta_1, \ldots, \beta_k$, $k = 1, 2, \ldots, K$, are the coefficients to be determined, and ε is a portion of random error.

Given a matrix $\mathbf{X}$ and a vector $\mathbf{y}$, data set may be divided into three sets: calibration ($\mathbf{X}_{cal}$ and $\mathbf{y}_{cal}$), validation ($\mathbf{X}_{val}$ and $\mathbf{y}_{val}$) and prediction ($\mathbf{X}_{pred}$ e $\mathbf{y}_{pred}$). Equation (2) shows how the regression coefficients can be calculated. Once obtained the regression coefficients, matrix $\mathbf{X}_{pred}$ and vector $\mathbf{y}_{pred}$ can be used to analyze the accuracy of the coefficients.

$$\beta = (\mathbf{X}_{cal}^T \mathbf{X}_{cal})^{-1} \mathbf{X}_{cal}^T \mathbf{y}_{cal}, \tag{2}$$

where $\mathbf{X}_{cal}$ is the matrix of variables and samples collected by instruments (used for the construction of multivariate calibration models), $\mathbf{y}_{cal}$ is the vector of dependent variables or a property of interest obtained in laboratory (reference parameter for model calibration), and β is the vector of regression coefficients.

The calculation for estimating the response variable involves the measures $\mathbf{X}_{val}$ and β:

$$\hat{y} = \mathbf{X}_{val}\beta. \tag{3}$$

The predictive ability of the calibration model is calculated by the Root Mean Squared Error of Prediction (RMSEP), which is a measure of absolute error:

$$\text{RMSEP} = \sqrt{\frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{N}}, \tag{4}$$

where $\hat{y}$ is the estimated value, y is the actual value of the property of interest and $N = N_{cal} + N_{val} + N_{pred}$ is the total number of samples.

Another criteria that may be used to determine the predictive ability of MLR models is the Mean Absolute Percentage Error (MAPE) [20]. MAPE is a relative measure to express errors as a percentage of the actual data defined as:

$$MAPE = \frac{\sum \mid \frac{\mathbf{y}_i - \hat{\mathbf{y}}_i}{\mathbf{y}_i} \mid}{N}(100) = \frac{\sum \mid \frac{e_i}{\mathbf{y}_i} \mid}{N}(100), \tag{5}$$

where $\mathbf{y}_i$ is the actual data at variable i, $\hat{\mathbf{y}}_i$ is the forecast at variable i, e_i is the forecast error at variable i, and N is the number of samples.

2.2 SPA

For standard MLR method, the design matrix $\mathbf{X}$ must have full column rank in order to calculates the Eq. 2, however, often we have a condition known as multicollinearity in the predictor variables. SPA is an iterative procedure widely used for variable selection in calibration models [21]. Given an initial variable, a new selected variable is inserted into the data subset with greater orthogonal projection in relation to the previous variable until a maximum number m is reached [14]. The three phases of SPA are described as follows:

1. Phase 1: it consists of projection operations in matrix $\mathbf{X}_{cal}$. The projections are used to generate variable chains. Each element of a chain is included so as to obtain the largest orthogonal projection according the procedure described by [14].

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{x}_i(\mathbf{x}_i)^T}{(\mathbf{x}_i)^T\mathbf{x}_i}, \tag{6}$$

where $\mathbf{I}$ is the identity matrix with dimension $N_{cal} \times N_{cal}$, $\mathbf{x}_i$ is the i-th column of the matrix $\mathbf{X}_{cal}$, and $\mathbf{P}$ is the projection matrix;

2. Phase 2: subsets of candidate variables are evaluated according to the error of the model.
3. Phase 3: it reduces the number of variables selected in Phase 2 discarding those that do not contribute to the predictive ability of the model.

In Phase 2 (Algorithm 1), SPA uses the validation set to assess subsets of variables extracted from the chains generated in Phase 1. The best subset of variables is the smallest one that provide the lowest error value between the tested subsets. The Phase 2 is considered the computational bottleneck of the SPA compared to other phases [19]. In addition, the inverse matrix calculation of the regression (line 8 of Algorithm 1) may require a large computational effort and contribute to a low performance of the SPA [14].

Algorithm 1: Phase 2 of SPA.

```
1.  Let be $K$ the number of variables available and $M$ the maximum number of variable that can be
    selected.
2.  do $k = 1$
3.  while $k < K$
4.      do $m = 1$
5.      while $m < M$
6.          Let $\mathbf{X}_{k\times m}$ be a subset of variables composed by the first $m$ elements of the $k$-th chain
    generated in step 1
7.          Let $\mathbf{S}^{-1}_{k\times m}$ be the inverse matrix of regression $(\mathbf{X}^T_{cal}\mathbf{X}_{cal})^{-1}\mathbf{X}^T_{cal}\mathbf{y}_{cal}$
8.          Using the variables contained in $\mathbf{X}_{k\times m}$, compute the inverse matrix $\mathbf{S}^{-1}_{k\times m}$ and the rest
    of the regression $(\mathbf{X}^T_{cal}\mathbf{X}_{cal})^{-1}\mathbf{X}^T_{cal}\mathbf{y}_{cal}$
9.          Compute the error of the $k$-th chain with $m$ variables
10.         do $m = m + 1$
11.     end while $m$
12.     do $k = k + 1$
13. end while $k$
```

3 Proposal

Let us consider $\{x_1, x_2, \ldots, x_M\}$ as a chain of variables obtained in Phase 1 of SPA. Phase 2 may use these variables to obtain m progressively larger MLR models, starting from a single-variable (x_1), followed by models with two (x_1, x_2), up to M ($x_1, x_2, \ldots, x_M$) variables. Each model can be obtained by a least-squares regression procedure. However, this procedure requires the calculation of the inverse of progressively larger matrices. On the other hand, the proposed formulation of Parallel Regressions (PR) can reduce the computational time of Phase 2 by avoiding the need of matrix inversions [19,22].

The formulation of PR starts from a single-variable model as follows:

$$y = \beta_1^{(1)} x_1^t + \epsilon^{y|x_1}, \tag{7}$$

where β_1 is the regression coefficient and ϵ is the residue. The superscripts (1), t and $y|x_1$ denote, respectively, one independent variable that is employed in the model, each thread t access one element of the vector, and y is regressed on x_1.

The least-squares estimate of $\beta_1^{(1)}$ can be given by

$$\beta_1^{(1)} = \frac{\sum_{i=1}^{N} y_i x_{i,1}^t}{\sum_{i=1}^{N} (x_{i,1}^t)^2}, \tag{8}$$

where y_i and $x_{i,1}$ represent the values of y and x_1 for the i-th calibration object (line i of matrix $\mathbf{X}$), respectively.

By using a similar notation, the two-variable model may be written as

$$y = \beta_1^{(2)} x_1^t + \beta_2^{(2)} x_2^t + \epsilon^{y|x_1,x_2}. \tag{9}$$

In attempt to obtain $\beta_1^{(2)}$ and $\beta_2^{(2)}$, x_2 is initially regressed on x_1 according to a model of the form

$$x_2 = \hat{\delta}_1^{x_2|x_1} x_1^t + \epsilon^{x_2|x_1}. \tag{10}$$

where the estimate coefficient $\hat{\delta}_1^{x_2|x_1}$ is calculated by univariate regression as

$$\hat{\delta}_1^{x_2|x_1} = \frac{\sum_{i=1}^{N} x_{i,2}^t x_{i,1}^t}{\sum_{i=1}^{N} (x_{i,1}^t)^2}. \tag{11}$$

Then, $\beta_1^{(2)}$ and $\beta_2^{(2)}$ may be obtained as

$$\beta_2^{(2)} = \frac{\sum_{i=1}^{N} e_i^{y|x_1} x_{i,2}^t}{\sum_{i=1}^{N} e_i^{x_2|x_1} x_{i,2}^t}, \beta_1^{(2)} = \beta_1^{(1)} - \delta_1^{x_2|x_1} \beta_2^{(2)}, \tag{12}$$

where

$$e_i^{y|x_1} = y_i - \beta_1^{(1)} x_{i,1}^t, \tag{13}$$

and

$$e_i^{x_2|x_1} = x_{i,2} - \hat{\delta}_1^{x_2|x_1} x_{i,1}^t. \tag{14}$$

Such procedure may be generalized to obtain a model with m variables from a model with $m - 1$ variables, where m ranges from 2 to M [19]. Thus, the new independent variable x_m is regressed on $\{x_1, x_2, \ldots, x_{m-1}\}$ according to a model of the following form

$$x_m = \hat{\delta}_1^{x_m|x_1,\ldots,x_{m-1}} x_1^t + \hat{\delta}_2^{x_m|x_1,\ldots,x_{m-1}} x_2^t + \cdots + \hat{\delta}_{m-1}^{x_m|x_1,\ldots,x_{m-1}} x_{m-1}^t + \epsilon^{x_m|x_1,\ldots,x_{m-1}}. \tag{15}$$

The coefficients $\beta_1^{(m)}, \beta_2^{(m)}, \ldots, \beta_m^{(m)}$ of the m-variable model are calculated as

$$\beta_m^{(m)} = \frac{\sum_{i=1}^{N} e_i^{y|x_1,\ldots,x_{m-1}} x_{i,m}^t}{\sum_{i=1}^{N} e_i^{x_m|x_1,\ldots,x_{m-1}} x_{i,m}^t}, \tag{16}$$

$$\beta_{m-j}^{(m)} = \beta_{m-j}^{(m-1)} - \hat{\delta}_{m-j}^{x_m|x_1,\ldots,x_{m-1}} \beta_m^{(m)},\ j = 1, \ldots, m-1, \tag{17}$$

where

$$\begin{aligned} e_i^{x_m|x_1,\ldots,x_{m-1}} &= x_{i,m}^t - (\hat{\delta}_1^{x_m|x_1,\ldots,x_{m-1}} x_{i,1}^t + \hat{\delta}_2^{x_m|x_1,\ldots,x_{m-1}} x_{i,2}^t \\ &\quad + \cdots + \hat{\delta}_{m-1}^{x_m|x_1,\ldots,x_{m-1}} x_{i,m-1}^t), \end{aligned} \tag{18}$$

and

$$e_i^{y|x_1,\ldots,x_{m-1}} = y_i - (\beta_1^{(m-1)} x_{i,1}^t + \beta_2^{(m-1)} x_{i,2}^t + \cdots + \beta_{m-1}^{(m-1)} x_{i,m-1}^t). \tag{19}$$

It is worthy to note that the superscript t indicates that all basic mathematical operations between vectors and matrices are performed in parallel by threads in order to increase the computing performance of the algorithm. The Sect. 3.1 presents a numerical example to facilitate the understanding of the formulation.

The estimation of regression coefficients using a least-squares regression sequential procedure has computational complexity $O(n^3)$ due to the matrix inversion calculation [22]. On the other hand, using our proposed strategy the largest complexity consists on the calculation of Eqs. (8), (11), (12) and (16), which have computational complexity $O(n^2)$ and it can be reduced by parallelization. The Sect. 3.2 provides an example about the exploitation of parallelism.

3.1 Numerical example for the parallel regressions strategy

Let $\mathbf{X}_{3\times3}$ and $\mathbf{y}_{3\times1}$ be the matrices randomly obtained bellow:

$$\mathbf{X} = \begin{bmatrix} 0.9528 & 0.5982 & 0.8368 \\ 0.7041 & 0.8407 & 0.5187 \\ 0.9539 & 0.4428 & 0.0222 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 0.3759 \\ 0.8986 \\ 0.4290 \end{bmatrix}. \tag{20}$$

Initially, we assume a scenario in which it is used only the first column of $\mathbf{X}$: $\mathbf{x}_1 = [0.9528 \quad 0.7041 \quad 0.9539]^T$. The regression coefficient β_{x_1}, that is, the regression with just the variable x_1 is equal to $\beta_1^{(1)}$, which is then obtained by substituting the respective values in Eq. (8), which leads to

$$\begin{aligned} \beta_1^{(1)} &= \frac{(0.3759 \times 0.9528) + (0.8986 \times 0.7041) + (0.4290 \times 0.9539)}{(0.9528)^2 + (0.7041)^2 + (0.9539)^2} \\ &= 0.6052. \end{aligned} \tag{21}$$

Now we add the second column of $\mathbf{X}$: $\mathbf{x}_2 = [0.5982 \quad 0.8407 \quad 0.4428]^T$. In order to obtain the regression coefficients $\beta_1^{(2)}$ and $\beta_2^{(2)}$, that is, the coefficients with two variables, we must calculate $\beta_2^{(2)}$ according to Eq. 12. However, first we need to regress $\mathbf{x}_2$ on $\mathbf{x}_1$ in order to obtain $\hat{\delta}^{x_2|x_1}$ (Eq. 11), $e^{y|x_1}$ (Eq. 13) and $e^{x_2|x_1}$ (Eq. 14).

$$\begin{aligned}\hat{\delta}^{x_2|x_1} &= \frac{(0.5982 \times 0.9528) + (0.8407 \times 0.7041) + (0.4428 \times 0.9539)}{(0.9528)^2 + (0.7041)^2 + (0.9539)^2} \\ &= 0.6848, \end{aligned} \tag{22}$$

$$e^{y|x_1} = \begin{bmatrix} 0.3759 \\ 0.8986 \\ 0.4290 \end{bmatrix} - 0.6052 \begin{bmatrix} 0.9528 \\ 0.7041 \\ 0.9539 \end{bmatrix} = \begin{bmatrix} -0.2007 \\ 0.4725 \\ -0.1483 \end{bmatrix} \tag{23}$$

and

$$e^{x_2|x_1} = \begin{bmatrix} 0.5982 \\ 0.8407 \\ 0.4428 \end{bmatrix} - 0.6848 \begin{bmatrix} 0.9528 \\ 0.7041 \\ 0.9539 \end{bmatrix} = \begin{bmatrix} -0.0543 \\ 0.3586 \\ -0.2104 \end{bmatrix}. \tag{24}$$

Substituting the values found in Eq. (12) we have

$$\begin{aligned}\beta_2^{(2)} &= \frac{(-0.2007 \times 0.5982) + (0.4725 \times 0.8407) + (-0.1483 \times 0.4428)}{(-0.0534 \times 0.5982) + (0.3586 \times 0.8407) + (-0.2104 \times 0.4428)} \\ &= 1.2033, \end{aligned} \tag{25}$$

$$e^{y|x_2,x_1} = \begin{bmatrix} -0.2007 \\ 0.4725 \\ -0.1483 \end{bmatrix} - 1,2033 \begin{bmatrix} -0.0543 \\ 0.3585 \\ -0.2104 \end{bmatrix} = \begin{bmatrix} -0.1354 \\ 0.0411 \\ 0.1049 \end{bmatrix}. \tag{26}$$

Updating $\beta_1^{(1)}$ to $\beta_1^{(2)}$, we have

$$\begin{aligned}\beta_1^{(2)} &= 0.6052 - (0.6848 \times 1.2032) \\ &= -0.2188. \end{aligned} \tag{27}$$

Thus, the regression coefficients with variables x_1 and x_2 is

$$[\beta_1^{(2)} \beta_2^{(2)}]^T = [-0.2188 \quad 1.2033]^T. \tag{28}$$

For comparison purposes, using the classical method with matrix inversion, it is easy to see that the result showed in (29) is equal to the result in (28). For this, where $\mathbf{X}_{1,2}$ is a matrix containing the first and second column of $\mathbf{X}$, we have

$$
\begin{aligned}
(\mathbf{X}_{1,2}^T\mathbf{X}_{1,2})^{-1} &= \left(\begin{bmatrix} 0.9528 & 0.7041 & 0.9539 \\ 0.5982 & 0.8407 & 0.4428 \end{bmatrix} \times \begin{bmatrix} 0.9528 & 0.5982 \\ 0.7041 & 0.8407 \\ 0.9539 & 0.4428 \end{bmatrix}\right)^{-1} \\
&= \begin{bmatrix} 3.0998 & -3.8952 \\ -3.8952 & 5.6879 \end{bmatrix}. \\
\beta_{1,2} &= (\mathbf{X}_{1,2}^T\mathbf{X}_{1,2})^{-1}\mathbf{X}_{1,2}^T\mathbf{y}, \\
\beta_{1,2} &= \begin{bmatrix} 3.0998 & -3.8952 \\ -3.8952 & 5.6879 \end{bmatrix} \times \begin{bmatrix} 0.9528 & 0.7041 & 0.9539 \\ 0.5982 & 0.8407 & 0.4428 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 0.3759 \\ 0.8986 \\ 0.4290 \end{bmatrix}, \\
\beta_{1,2} &= \begin{bmatrix} -0.2188 \\ 1.2033 \end{bmatrix} = \begin{bmatrix} \beta_1^{(2)} \\ \beta_2^{(2)} \end{bmatrix}. \qquad (29)
\end{aligned}
$$

Finally, we add the third column of $\mathbf{X}$: $\mathbf{x}_3 = [0.8368 \quad 0.5187 \quad 0.0222]^T$. In a similar manner to the univariate model, we first regress $\mathbf{x}_3$ on $\mathbf{x}_2$ and $\mathbf{x}_1$.

$$
\begin{aligned}
\hat{\delta}^{x_3|x_1} &= \frac{(0.8368 \times 0.9528) + (0.5187 \times 0.7041) + (0.0222 \times 0.9539)}{(0.9528)^2 + (0.7041)^2 + (0.9539)^2} \\
&= 0.5116, \qquad (30)
\end{aligned}
$$

and

$$
\begin{aligned}
e^{x_3|x_1} &= \begin{bmatrix} 0.8368 \\ 0.5187 \\ 0.0222 \end{bmatrix} - 0.5116 \begin{bmatrix} 0.9528 \\ 0.7041 \\ 0.9539 \end{bmatrix} = \begin{bmatrix} 0.3493 \\ 0.1585 \\ -0.4659 \end{bmatrix}. \qquad (31) \\
\hat{\delta}^{x_3|x_1,x_2} &= \frac{(0.3493 \times 0.5982) + (0.1585 \times 0.8407) + (-0.4659 \times 0.4428)}{(0.5982 \times -0.0543) + (0.8407 \times 0.3585) + (0.4428 \times -0.2104)} \\
&= 0.7731, \qquad (32)
\end{aligned}
$$

and

$$
e^{x_3|x_1,x_2} = \begin{bmatrix} 0.3493 \\ 0.1585 \\ -0.4659 \end{bmatrix} - 0.7731 \times \begin{bmatrix} -0.0543 \\ 0.3585 \\ -0.2104 \end{bmatrix} = \begin{bmatrix} 0.3913 \\ -0.1187 \\ -0.3032 \end{bmatrix}. \qquad (33)
$$

Updating $\hat{\delta}^{x_3|x_1}$,

$$
\begin{aligned}
\hat{\delta}^{x_3|x_1} &= \hat{\delta}^{x_3|x_1} - (\hat{\delta}^{x_2|x_1} \times \hat{\delta}^{x_3|x_1,x_2}), \\
&= 0.5116 - (0.6848 \times 0.731), \qquad (34) \\
&= -0.0177. \qquad (35)
\end{aligned}
$$

Thus, we can obtain $\beta_3^{(3)}$

$$\beta_3^{(3)} = \frac{(-0.1354 \times 0.8368) + (0.0410 \times 0.5187) + (-0.1049 \times 0.0222)}{(0.3913 \times 0.8368) + (-0.1187 \times 0.5187) + (-0.3033 \times 0.0222)} \\ = -0.3461. \tag{36}$$

Updating $\beta_2^{(2)}$ to $\beta_2^{(3)}$,

$$\beta_2^{(3)} = 1.2033 - 0.7728 \times (-0.3469), \\ = 1.4708, \tag{37}$$

and $\beta_1^{(2)}$ to $\beta_1^{(3)}$,

$$\beta_1^{(3)} = -0.2188 - (-0.0176) \times (-0.3459), \\ = -0.2250. \tag{38}$$

For comparison purposes, the classical least-squares regression is calculated for the example in question as shown below:

$$\begin{aligned}
(\mathbf{X}^T\mathbf{X})^{-1} &= \left(\begin{bmatrix} 0.9528 & 0.7041 & 0.9539 \\ 0.5982 & 0.8407 & 0.4428 \\ 0.8368 & 0.5187 & 0.0222 \end{bmatrix} \times \begin{bmatrix} 0.9528 & 0.5982 & 0.8368 \\ 0.7041 & 0.8407 & 0.5187 \\ 0.9539 & 0.4428 & 0.0222 \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} 3.1010 & -3.9476 & 0.0678 \\ -3.9476 & 7.9925 & -2.9821 \\ 0.0678 & -2.9821 & 3.8587 \end{bmatrix}. \\
\beta &= \begin{bmatrix} 3.1010 & -3.9476 & 0.0678 \\ -3.9476 & 7.9925 & -2.9821 \\ 0.0678 & -2.9821 & 3.8587 \end{bmatrix} \times \begin{bmatrix} 0.9528 & 0.7041 & 0.9539 \\ 0.5982 & 0.8407 & 0.4428 \\ 0.8368 & 0.5187 & 0.0222 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 0.3759 \\ 0.8986 \\ 0.4290 \end{bmatrix} \\
&= \begin{bmatrix} -0.2250 \\ 1.4708 \\ -0.3461 \end{bmatrix} = \begin{bmatrix} \beta_1^{(3)} \\ \beta_2^{(3)} \\ \beta_3^{(3)} \end{bmatrix}.
\end{aligned} \tag{39}$$

As can be seen, the result showed in (39) is the same which had been reached in Eqs. (36), (37) and (38). Therefore, PR strategy is able to provide the same results without performing matrix inverse calculations.

3.2 Parallel regression encoding in SPA using GPU

Programming models such as Compute Unified Device Architecture (CUDA) [23] allows that applications can be run more easily on the GPU. CUDA was the first architecture and Application Programming Interface (API), created by $NVIDIA$©in 2006 to allow the GPU to be used for a wide variety of applications [23]. However, as any technology the GPU has its limitations. Depending on the data volume, GPU's computational performance may shown itself inferior when compared to Central Processing Unit (CPU) performance [11]. In this case, the data amount to be transferred to the GPU memory must be taken into account, because there is an overhead associated with the parallelization of tasks on the GPU [11,12]. Factors regarding the access time to memory may also influence the computational performance. In other words, access to GPU global memory usually has a high latency and it is subject to a coalesced access to data in memory [23].

Before starting Phase 2, the data is transferred to the GPU (device) memory, coprocessor of the CPU (host) where the division process is done in several tasks (threads) to run concurrently. The Phase 2 of SPA is parallelized by the kernel[2] showed in Algorithm 2. The function has four input parameters and one output variable. Each thread calculates the coefficients of the k-th chain generated in Phase 1. In classical implementation, all K chains are evaluates $N2$ using least squares with matrix inversion sequentially. Our approach calculate the coefficients for each chain in parallel avoiding the matrix inversion.

Using the regression formulation proposed in Sect. 3, the kernel function uses the device function called PartReg (Algorithm 3) to calculate Eqs. (8), (11), (12) and (16). The only difference is that one must change the entries for each equation. PartReg uses other two functions showed in Algorithms 4 and 5 to calculates the summation of dot product and element-wise division, respectively.

4 Experimental

All data used in this paper consist of whole wheat samples obtained from plant material of Canadian producers. The data were determined in the Grain Research Laboratory by 1090 reflectance spectra in the near infrared (NIR) of whole wheat samples, which were used as reference data in the 2008 International Conference of Diffuse reflectance (http://www.idrc-chambersburg.org/shootout.html).

The protein content was chosen as the property of interest. The spectra were acquired by a spectrophotometer in the range 400–2500 nanometers (nm) with a resolution of 2 nm. In this work, the NIR was used in the range of 1100–2500 nm. The reference values of protein concentration in wheat samples were determined in the laboratory by the Kjeldahl method [24]. This method uses the destruction of organic substances in the presence of concentrated sulfuric acid and a catalyst through the action of heat, with subsequent distillation of nitrogen from the sample. However, the use of indirect

[2] Kernel is a function that is performed on the device by each thread. Threads are organized into blocks [23].

Algorithm 2: Kernel: PR-CUDA.

```
1.  **Input Parameters**:
2.      **Xcal**: Matrix of samples and independents variables;
3.      **Ycal**: Response variables;
4.      $N2$: Maximum number of variables in each chain;
5.      **Lambdas**: Matrix ($K \times N2$), i.e., $K$ chains with $N2$ selected variables in Phase 1.
6.  **Output Parameters**:
7.      Data structure $\beta$ with $K$ elements. Each element has $N2$ regression coefficients;
8.  $k \leftarrow$ thread identification
9.  $\mathbf{chain} \leftarrow \mathbf{lambdas}(k, :)$
    $\mathbf{error}(:, 1), \mathbf{betas}(1, 1) \leftarrow$ PartReg($\mathbf{y}, \mathbf{Xcal}_{\mathbf{chain}(1)}, \mathbf{Xcal}_{\mathbf{chain}(1)}$)
10. $\mathbf{coefficients}(1) \leftarrow \mathbf{betas}(1, 1)$
11. **for** $p = 2$ **to** $N2$
        $\mathbf{error}(:, p), \mathbf{betas}(p + 1, 1)$ = PartReg($\mathbf{Xcal}_{\mathbf{chain}(p)}, \mathbf{Xcal}_{\mathbf{chain}(1)}, \mathbf{Xcal}_{\mathbf{chain}(1)}$)
12. **end for**
13. **for** $m$ = 2 **to** $N2$
14.     $\mathbf{error}(:, m + 1)$, $\mathbf{betas}$(m+1,1) = PartReg($\mathbf{e2}$, $\mathbf{error}(:, m)$, $\mathbf{Xcal}_{\mathbf{chain}(m)}$)
15.     **for** $d$ = m+1 **to** $N2$
16.         $\mathbf{error}(:, d), \mathbf{betas}(d + 1, m)$ = PartReg($\mathbf{error}(:, d), \mathbf{error}(:, m), \mathbf{Xcal}_{\mathbf{chain}(m)}$)
17.         **for** $j = 1$ **to** $m - 1$
18.             $\mathbf{betas}(d + 1, j) = \mathbf{betas}(d + 1, j) - (\mathbf{betas}(m + 1, j) \times \mathbf{betas}(d + 1, m))$
19.         **end for**
20.     **end for**
21.     $\mathbf{coefficients}(m) \leftarrow \mathbf{betas}(1, 1 : m)$
22. **end for** $\beta(k) \leftarrow \mathbf{coefficients}$
```

Algorithm 3: Device Function PartReg: Calculates partial regressions.

```
1. **Input Parameters**: **y**, **xx**, **xc**, $N$.
2. **Output Parameters**: **betas**, **r**.
3. $\mathbf{betas}_1 \leftarrow DotK(\mathbf{y}, \mathbf{xc})$
4. $\mathbf{betas}_2 \leftarrow DotK(\mathbf{xc}, \mathbf{xx})$
5. $\mathbf{betas} \leftarrow ElementWiseDiv(\mathbf{betas}_1, \mathbf{betas}_2)$
6. $\mathbf{r} = \mathbf{y} - VecProduct(\mathbf{betas}, \mathbf{xx})$
```

Algorithm 4: Device Function ElementWiseSumMult: function to calculates summation of dot product.

```
1. **Input Parameters**: **v1**, **v2**, $N$.
2. **Output Parameters**: **v**.
3. $\mathbf{v} \leftarrow 0$
4. While $j < N$
5.         $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{v1}(j) * \mathbf{v2}(j)$
```

Algorithm 5: Device Function ElementWiseDiv: function to calculates vector division.

```
1. **Input Parameters**: **v1**, **v2**, $N$.
2. **Output Parameters**: **v**.
3. While $j < N$
4.         $\mathbf{v}(j) \leftarrow \mathbf{v1}(j) / \mathbf{v2}(j)$
```

instrumental methods such as NIR and mathematical models as MLR enables the protein level to be determined without destroying the sample [12,25].

All calculations were carried out by using a desktop computer with an Intel Core i7 2600 (3.40 GHz), 8 GB of RAM memory and a $NVIDIA$©GeForce GTX 780Ti graphics card with 2880 CUDA cores and 3 GB of memory config. The Matlab 8.1.0.604 (R2013a) software platform was employed throughout. Furthermore, all outcomes were obtained by averaging twenty executions.

5 Results and discussion

The variables selected using the proposed implementations can be visualized by Fig. 1. The maximum number of variables selected $N2$ is defined by user. Such outcome showed in the chart indicates that these regions are the most promising in the spectrophotometer. In practice, this result implies a small number of wavelengths in a spectrophotometer to quantify the property of protein concentration in real samples of wheat [12].

Figure 2 presents the real values versus predictions using SPA-PR-CUDA and the implementation proposed by Soares et al. [19]. Differences between predictions and current concentrations result in points on the straight line. Predicted concentrations are close to the actual concentrations for both methods. However, the model using the variables selected by SPA-PR-CUDA is nearest the line than Soares et al. [19] predictions. It indicates that the variables selected by our proposed implementation is able to provide a model with a more appropriate prediction capacity.

Figure 3 shows the computational time of Phase 2 using four different SPA implementations. Multi-core implementation of SPA-PR in CPU uses the par-for command of Matlab parallel toolbox. The best result was obtained using eight threads defined experimentally. For instance, when the maximum number of variables is equal to 250,

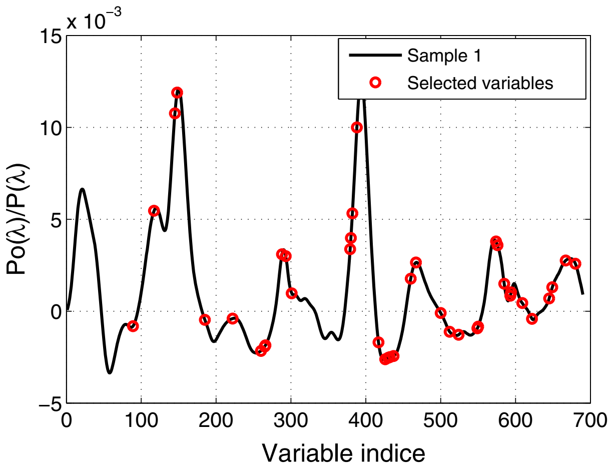

Fig. 1 Visualization of selected variables

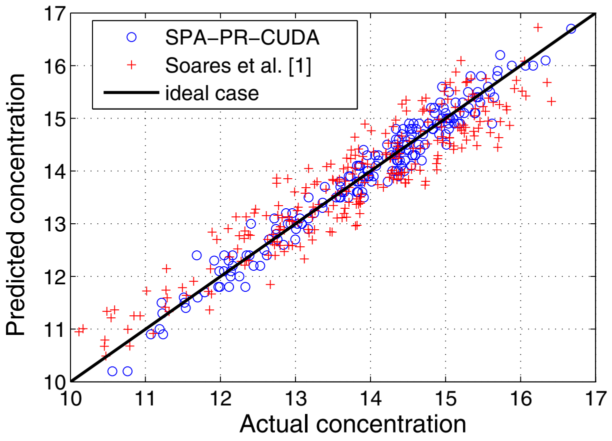

Fig. 2 Comparison between predicted and actual protein concentration using SPA-PR-CUDA and Soares et al. [19] implementation

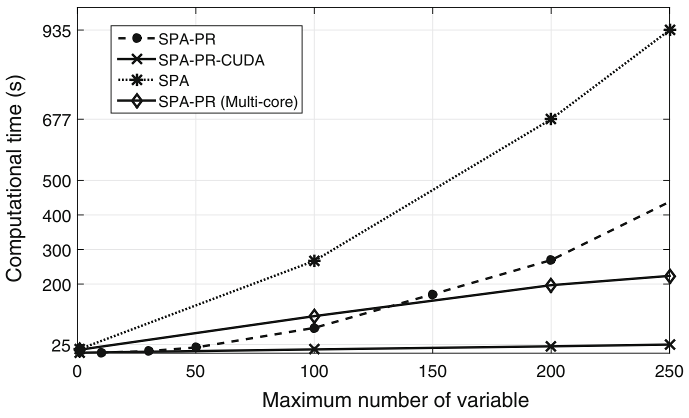

Fig. 3 Comparison of computational performance between classical implementation of SPA, SPA-PR (single and multi-core) and SPA-PR-CUDA

regressions with 1 up to 250 variables are performed. One can observe that the time spent increases according to the number of variables selected for all implementations. In SPA-PR-CUDA the number of threads per block equal to 512 was capable of providing the most reduced computational time. Nevertheless, the increase is significantly less pronounced for SPA-PR-CUDA.

Table 1 presents a comparison of computational time between the proposed implementations. It is possible to note that SPA-PR-CUDA is approximately 9× faster than SPA-PR implementation performing on a multi-core CPU. Regarding the classical SPA and SPA-PR, SPA-PR-CUDA is 37× and 12× faster, respectively.

Finally, a comparison between SPA-PR-CUDA and two traditional algorithms (GA-MLR [10] and PLS [4]) used for variable selection is showed in Table 2. It demonstrates

Table 1 Computational time (seconds) for SPA (classical implementation), SPA-PR (single and multi-core) and SPA-PR-CUDA

Algorithm	Maximum number of variables		
	100	200	250
SPA	267.05	677.00	935.05
SPA-PR	73.13	269.33	468.42
SPA-PR (multi-core)	107.13	196.74	223.63
SPA-PR-CUDA	10.88	19.80	24.87

Table 2 Results for SPA-PR-CUDA, GA-MLR and PLS

	Number of variables	RMSEP	MAPE (%)
GA-MLR	146	0.21	1.50
PLS	15	0.21	1.50
SPA-PR-CUDA	13	0.20	1.43

that SPA-PR-CUDA was able to yield the best results in terms of number of variables selected as well as in terms of prediction error.

6 Conclusion and future works

SPA is an iterative method that can be used for variable selection in multivariate calibration problems. Its major limitation consists in the calculation of an inverse matrix every time a new variable is inserted into the calibration model. In order to deal with this restriction, this paper proposes a new strategy called Parallel Regressions (PR). PR strategy uses in Phase 2 of SPA a mathematical formulation that eliminates the need of matrix inverse calculation. Furthermore, PR strategy can exploit a GPU through CUDA to increase the computational performance of the algorithm.

In comparison with computational time obtained by SPA-PR implementations, one can observe that SPA-PR-CUDA is more efficient. For instance, for 250 variables selected SPA-PR and SPA-PR-CUDA perform around 468 and 24 s, respectively. Therefore, it is possible to conclude that SPA-PR-CUDA is about twelve times faster than the sequential implementation , specially when the number of variables available is relatively large for the selection of variables in multivariate calibration problems. Moreover, when compared with traditional algorithms SPA-PR-CUDA may be a better alternative for obtaining a model with a lower value of the prediction error as well as a significant reduction of the number of variables.

Future works may present new comparisons between SPA-PR-CUDA and other variable selection techniques such as bio-inspired metaheuristics (e.g. Firefly Algorithm. Additionally, improvements in the parallel strategies may be implemented in order to make an even better usage of the GPU architecture and its memory hierarchy.

Acknowledgements Authors thank to the research agencies CAPES, CNPq, FAPESP and FAPEG for the support provided to this study. This is also a contribution of the National Institute of Advanced Analytical Science and Technology (INCTAA) (CNPq-proc. no. 573894/2008-6 e FAPESP proc. no. 2008/57808-1).

References

1. Lavine BK, Workman J (2005) Chemometrics: past, present, and future. ACS symposium series, vol 894. Oxford University Press, Vandœuvre-lès-Nancy, pp 1–13. doi:10.1021/bk-2005-0894.ch001
2. Chau F-T, Liang Y-Z, Gao J, Shao X-G (2004) Chemometrics from basic to wavelet transform. Willey, Hoboken
3. Tang Y, Liang Y, Fang K-T (2003) Data mining in chemometrics: sub-structures learning via peak combinations searching in mass spectra. J Data Sci 1:481–496
4. Lawson CL, Hanson RJ (1974) Solving least squares problems. SIAM, Philadelphia
5. Cortina JM (1994) Interaction, nonlinearity, and multicollinearity: implications for multiple regression. J Manag 19:915–922
6. Montgomery DC, Peck EA, Vining GG (2012) Introduction to Linear Regression Analysis, Wiley Series in Probability and Statistics
7. Estienne F (2003) New trends in multivariate analysis and calibration, Laboratorium voor Farmaceutische en Biomedische Analyse
8. Skoog DA, Leary JJ (2002) Princpios de Anlise Instrumental. Artmed Editora S.A, Porto Alegre
9. Martens H (1991) Multivariate calibration. Wiley, New York
10. Soares AS, Lima TW, Soares FA, Coelho CJ, Delbem AC (2014) Mutation-based compact genetic algorithm for spectroscopy variable selection in determining protein concentration in wheat grain. Electron Lett 50:932–934
11. Paula LCM, Soares AS, Soares TW, Martins WS, Filho ARG, Coelho CJ (2013) Partial parallelization of the successive projections algorithm using compute unified device architecture. In: International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, p 737–741
12. Paula LCM, Soares AS, Soares TW, Delbem ACB, Coelho CJ, Filho ARG (2014) Parallelization of a modified firefly algorithm using GPU for variable selection in a multivariate calibration problem. Int J Natl Comput Res 4:31–42
13. Soares AS, Galvão Filho AR, Galvão RKH, Araújo MCU (2010) Multi-core computation in chemometrics: case studies of voltammetric and NIR spectrometric analyses. J Braz Chem Soc 21:1626–1634
14. Soares SFC, Gomes AA, Araújo MC, Galvão RK, Filho ARG (2013) The successive projections algorithm. TrAC Trends Anal Chem 42:84–98
15. Marreto PD, Zimer AM, Faria RC, Mascaro LH, Pereira EC, Fragoso WD, Lemos SG (2014) Multivariate linear regression with variable selection by a successive projections algorithm applied to the analysis of anodic stripping voltammetry data. Electrochim Acta 127:6878
16. Moreira ED, Pontes MJ, Galvão RK, Araújo MC (2009) Near infrared reflectance spectrometry classification of cigarettes using the successive projections algorithm for variable selection. Talanta 79:5. doi:10.1016/j.talanta.2009.05.031
17. Tang G, Huang Y, Tian K, Song X, Yan H, Hu J, Xionga Y, Min S (2014) A new spectral variable selection pattern using competitive adaptive reweighted sampling combined with successive projections algorithm. Analyst. doi:10.1039/C4AN00837E
18. Pontes MJC, Galvo RKH, Araújo MCU, Moreira PNT, Neto ODP, José GE, Saldanha TCB (2005) The successive projections algorithm for spectral variable selection in classification problems. Chemom Intell Lab Syst 78(12):1118
19. Soares AS, Galvão Filho AR, Galvão RKH, Araújo MCU (2010) Improving the computational efficiency of the successive projections algorithm by using a sequential regression implementation: a case study involving NIR spectrometric analysis of wheat samples. J Braz Chem Soc 21:760–763
20. Makridakis SG, Hibon M (1995) Evaluating accuracy (or error) measures. INSEAD working paper. INSEAD, Fontainebleau
21. Araújo MCU, Saldanha TC, Galvão RK, Yoneyama T (2001) The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. Chemom Intell Lab Syst 57:65–73
22. Gusnanto A, Pawitan Y, Huang J (2003) Variable selection in random calibration of near-infrared instruments: ridge regression and partial least squares regression settings. J Chemom 17:174–185
23. $CUDA^{TM}$ (2013) NVIDIA CUDA C Programming Guide, NVIDIA Corporation, 5.0
24. Bradstreet RB (1965) The Kjeldahl method for organic nitrogen. Academic Press Inc., New York
25. Paula LCM, Soares AS, Delbem ACB, Lima TW, Coelho CJ, Filho ARG (2014) A GPU-based Implementation of the firefly algorithm for variable selection in multivariate calibration problems. Plos One 9:e114145