Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

Paralelização do Algoritmo das Projeções Sucessivas em GPU usando uma Implementação das Regressões Sequenciais para Seleção de Variáveis em Problemas de Calibração Multivariada

Lauro C. M. de Paula, **Anderson S. Soares**, **Telma W. Soares**,
UFG - Instituto de Informática
74001-970, Goiânia, GO

Arlindo R. G. Filho
ITA - Departamento de Sistemas e Controle
12228-900, São José dos Campos, SP
E-mail: argfilho@gmail.com

Clarimar J. Coelho
PUC Goiás - Departamento de Computação
74605-010, Goiânia, GO
E-mail: clarimarc@gmail.com,

Resumo: *Este artigo apresenta uma implementação paralela do Algoritmo das Projeções Sucessivas (APS) denominada APS-SR-CUDA. Tal implementação explora uma unidade de processamento gráfico (Graphics Processing Unit, GPU), por meio de uma plataforma de computação paralela (Compute Unified Device Architecture, CUDA), e o conceito de regressões sequenciais (Sequential Regression, SR). O APS é utilizado tradicionalmente para a seleção de variáveis no contexto da calibração multivariada. É um procedimento iterativo composto por três fases que cooperam para minimizar problemas de multicolinearidade de dados multivariados. A estratégia SR é empregada na Fase 2 para eliminar o cálculo da matriz de projeção do APS original. Os resultados obtidos com a aplicação do APS-SR-CUDA têm desempenho quatro vezes melhor quando comparado ao APS.*

Palavras-chave: *calibração multivariada, seleção de variáveis, GPU, APS, APS-SR-CUDA.*

1 Introdução

A presença de correlação linear entre duas ou mais variáveis de um conjunto de dados é considerada multicolinear [6]. O cálculo da matriz inversa em modelos de calibração pode apresentar problemas de estabilidade numérica quando ocorre a multicolineridade entre as variáveis de uma matriz [4]. Em problemas de predição multivariada, grande parte das variáveis não contribuem para a capacidade preditiva do modelo [14]. Nesse caso, a seleção de variáveis pode ser uma boa alternativa para eliminar a multicolinearidade e melhorar a capacidade preditiva dos modelos [7].

O APS é utilizado com sucesso para a seleção de variáveis no contexto de diferentes aplicações como, por exemplo, calibração multivariada [12]; classificação e detecção de falhas quando o objetivo é minimizar a multicolinearidade [2]; e melhorar a capacidade predtiva dos modelos de calibração [13]. Este trabalho apresenta os resultados obtidos com uma implementação paralela do APS, denominada APS-SR-CUDA, para reduzir o tempo computacional do APS original. Tal implementação reduz o tempo computacional por meio da utilização da estratégia de regressões

sequenciais, que evita o cálculo de inversão matricial. Os resultados mostram que o tempo computacional do APS-SR-CUDA, em aplicações típicas, pode ser reduzido significativamente.

2 Algoritmo das Projeções Sucessivas

Em modelos de calibração, dada a matriz $\mathbf{X}$ e o vetor $\mathbf{y}$, a modelagem divide os dados em três conjuntos: calibração ($\mathbf{X}_{cal}$ e $\mathbf{y}_{cal}$), validação ($\mathbf{X}_{val}$ e $\mathbf{y}_{val}$) e predição ($\mathbf{X}_{pred}$ e $\mathbf{y}_{pred}$). O APS é um procedimento iterativo bastante utilizado para a seleção de variáveis em modelos de calibração. Dado uma variável inicial, uma nova variável é inserida no subconjunto de dados caso tenha uma maior projeção ortogonal em relação à variável anterior [13]. Esse procedimento é realizado até que um número máximo m seja atingido [12].

O APS é composto por três fases:

1. Fase 1 - consiste em operações de projeção realizadas na matriz $\mathbf{X}_{cal}$. Cada elemento de uma cadeia é selecionado de modo a obter a maior projeção ortogonal

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{x}^i(\mathbf{x}^i)^T}{(\mathbf{x}_i)^T\mathbf{x}_i}, \tag{1}$$

   onde $\mathbf{I}$ é uma matriz identidade de dimensões $N_{cal} \times N_{cal}$, $\mathbf{x}^i$ é a i-ésima coluna da matriz $\mathbf{X}_{cal}$ e $\mathbf{P}$ a matriz de projeções.

2. Fase 2 - subconjuntos de variáveis candidatas são avaliados de acordo com o erro do modelo

3. Fase 3 - consiste na redução do número de variáveis selecionadas na Fase 2, descartando aquelas que não contribuem para a capacidade preditiva do modelo.

Na Fase 2 (Algoritmo 1), o APS utiliza o conjunto de validação para avaliar subconjuntos de variáveis extraídas a partir das cadeias geradas na Fase 1. O melhor subconjunto de variáveis fornece o menor valor de erro entre os subconjuntos testados. A Fase 2 é considerada o gargalo computacional do APS quando comparada com as outras fases. Além disso, o cálculo da matriz inversa da regressão (linha 7, Algoritmo 1) pode exigir grande esforço computacional e contribuir para um baixo desempenho do algoritmo, principalmente quando a matriz for grande [1, 9].

Algoritmo 1: Fase 2 do APS.

1. **faça** $k = 1$
2. **enquanto** $k < K$
3. **faça** $m = 1$
4. **enquanto** $m < M$
5. Seja $\mathbf{X}_{k\times m}$ um subconjunto de variáveis formado pelos m primeiros elementos da cadeia k gerada na Fase 1
6. Seja $\mathbf{S}^{-1}_{k\times m}$, a matriz inversa da regressão $(\mathbf{X}^T_{cal}\mathbf{X}_{cal})^{-1}\mathbf{X}^T_{cal}\mathbf{y}_{cal}$
7. Utilizando as variáveis contidas em $\mathbf{X}_{k\times m}$, calcule a inversa $\mathbf{S}^{-1}_{k\times m}$ e o restante da regressão $(\mathbf{X}^T_{cal}\mathbf{X}_{cal})^{-1}\mathbf{X}^T_{cal}\mathbf{y}_{cal}$
8. Calcule o erro da cadeia k com m variáveis
9. **faça** $m = m + 1$
10. **fim enquanto** m
11. **faça** $k = k + 1$
12. **fim enquanto** k

DOI: 10.5540/03.2015.003.01.0094

Soares [1] sugere uma implementação em Matlab que reduz o tempo computacional do APS que evita a inversão de matrizes. Tal implementação é baseada na estratégia de regressões sequenciais. Seja $\{x_1, x_2, \ldots, x_M\}$ uma cadeia de variáveis obtidas na Fase 1. Na Fase 2, as variáveis são utilizadas para obter M modelos de calibração iniciando a partir de um modelo com uma variável (x_1), seguindo com $(x_1,\ x_2)$ até $(x_1, x_2, \ldots, x_M)$ variáveis. Cada um dos modelos de calibração podem ser obtidos por um procedimento de mínimos quadrados que requer a inversão de matrizes maiores a medida que novas variáveis são adicionadas [5]. Entretanto, a formulação das regressões sequenciais reduz o tempo computacional evitando o cálculo de matrizes inversas [1].

A formulação das regressões sequenciais inicia-se a partir de uma única variável da seguinte maneira:

$$y = \beta_1^{(1)} x_1 + \epsilon^{y|x_1}, \tag{2}$$

onde $\beta_1^{(1)}$ é o coeficiente de regressão e $\epsilon^{y|x_1}$ o resíduo do modelo. Os superescritos (1) e $y|x_1$ denotam que uma variável independente é empregada no modelo e que y é regredido em x_1, respectivamente [1]. A estimativa dos mínimos quadrados de $\beta_1^{(1)}$ é dada por:

$$\hat{\beta}_1^{(1)} = \frac{\sum_{i=1}^{N} y_i x_{i,1}}{\sum_{i=1}^{N} (x_{i,1})^2}, \tag{3}$$

em que y_i e $x_{i,1}$ representam os valores de y e x_1 para o i-ésimo objeto de calibração (amostra), respectivamente, e N representa o número de amostras (observações). Usando notação similar, o modelo com duas variáveis pode ser escrito como:

$$y = \beta_1^{(2)} x_1 + \beta_2^{(2)} x_2 + \epsilon^{y|x_1,x_2}. \tag{4}$$

Para obter $\beta_1^{(2)}$ e $\beta_2^{(2)}$, x_2 é inicialmente regredido em x_1 de acordo com o modelo:

$$x_2 = \delta_1^{x_2|x_1} x_1 + \epsilon^{x_2|x_1}. \tag{5}$$

A estimativa do coeficiente $\delta_1^{x_2|x_1}$ pode ser calculada por uma regressão univariada como:

$$\hat{\delta}_1^{x_2|x_1} = \frac{\sum_{i=1}^{N} x_{i,2} x_{i,1}}{\sum_{i=1}^{N} (x_{i,1})^2}. \tag{6}$$

Então, $\hat{\beta}_1^{(2)}$ e $\hat{\beta}_2^{(2)}$ podem ser obtidos como:

$$\hat{\beta}_2^{(2)} = \frac{\sum_{i=1}^{N} e_i^{y|x_1} x_{i,2}}{\sum_{i=1}^{N} e_i^{x_2|x_1} x_{i,2}}, \hat{\beta}_1^{(2)} = \hat{\beta}_1^{(1)} - \hat{\delta}_1^{x_2|x_1} \hat{\beta}_2^{(2)}, \tag{7}$$

onde

$$e_i^{y|x_1} = y_i - \hat{\beta}_1^{(1)} x_{i,1}, \tag{8}$$

e

$$e_i^{x_2|x_1} = x_{i,2} - \hat{\delta}_1^{x_2|x_1} x_{i,1}. \tag{9}$$

O mesmo procedimento pode ser generalizado para obter um modelo com m variáveis a partir de um modelo com $m-1$ variáveis, onde m varia entre 2 e M. Para tal, a nova variável independente x_m é inicialmente regredida em $\{x_1,\ x_2,\ ...,\ x_{m-1}\}$ de acordo com

$$x_m = \delta_1^{x_m|x_1,\ldots,x_{m-1}} x_1 + \delta_2^{x_m|x_1,\ldots,x_{m-1}} x_2 + \ldots + \delta_{m-1}^{x_m|x_1,\ldots,x_{m-1}} x_{m-1} + \epsilon^{x_m|x_1,\ldots,x_{m-1}}. \tag{10}$$

DOI: 10.5540/03.2015.003.01.0094

Os coeficientes $\hat{\beta}$ do modelo com m variáveis são calculados como:

$$\hat{\beta}_m^{(m)} = \frac{\sum_{i=1}^{N} e_i^{y|x_1,\ldots,x_{m-1}} x_{i,m}}{\sum_{i=1}^{N} e_i^{x_m|x_1,\ldots,x_{m-1}} x_{i,m}}, \tag{11}$$

$$\hat{\beta}_{m-j}^{(m)} = \hat{\beta}_{m-j}^{(m-1)} - \hat{\delta}_{m-j}^{x_m|x_1,\ldots,x_{m-1}} \hat{\beta}_m^{(m)}, j = 1, \ldots, m-1, \tag{12}$$

onde

$$e_i^{x_m|x_1,\ldots,x_{m-1}} = x_{i,m} - (\hat{\delta}_1^{x_m|x_1,\ldots,x_{m-1}} x_{i,1} + \hat{\delta}_2^{x_m|x_1,\ldots,x_{m-1}} x_{i,2} + \ldots + \hat{\delta}_{m-1}^{x_m|x_1,\ldots,x_{m-1}} x_{i,m-1}), \tag{13}$$

e

$$e_i^{y|x_1,\ldots,x_{m-1}} = y_i - (\hat{\beta}_1^{(m-1)} x_{i,1} + \hat{\beta}_2^{(m-1)} x_{i,2} + \ldots + \hat{\beta}_{m-1}^{(m-1)} x_{i,m-1}). \tag{14}$$

Soares [1] apresenta o exemplo da determinação de proteína em amostras de trigo. As previsões do modelo apresentam baixos valores de erros e ganhos no tempo computacional em relação à implementação tradicional do APS. No entanto, não explora avanços recentes das arquiteturas computacionais nem a paralelização de tarefas [9].

Os algoritmos SR propostos neste trabalho, denominados APS-SR-MATLAB e APS-SR-CUDA, utilizam a estratégia de regressões sequenciais proposta por Soares [1]. O APS-SR-MATLAB e APS-SR-CUDA foram implementados, respectivamente, em Matlab e CUDA-C [3]. O APS-SR-CUDA é parcialmente paralelizado e executado como uma sub-rotina no ambiente Matlab por meio de executáveis *MEX* [11].

Como mostra o Algoritmo 2, o código do APS-SR-MATLAB inicia na Fase 1 do APS. Os parâmetros de entrada são: a matriz do conjunto de calibração (respostas instrumentais), validação e predição; vetor das variáveis dependentes do conjunto de calibração (concentrações), validação e predição; e número mínimo e máximo (normalmente igual a 1 e N_{cal}, respectivamente) de variáveis a serem selecionadas.

Algoritmo 2: Implementação APS-SR-MATLAB.

1. **Parâmetros**: $\mathbf{X}cal_{N_{cal}\times K}$, $\mathbf{X}val_{N_{val}\times K}$, $\mathbf{X}pred_{N_{pred}\times K}$, $\mathbf{y}cal_{N_{cal}\times 1}$, $\mathbf{y}val_{N_{cal}\times 1}$, $\mathbf{y}pred_{N_{pred}\times 1}$, $N1$, $N2$.
2. Executa a Fase 1:

   Centralização na média e auto-escalonamento das colunas de $\mathbf{X}cal$

   Geração das cadeias de variáveis, contendo no mínimo $N1$ e no máximo $N2$ variáveis, a partir de cada coluna de $\mathbf{X}cal$
3. Executa a Fase 2 utilizando o próprio código em Matlab ou o Algoritmo 3
4. Executa a Fase 3
5. Gera o gráfico das variáveis selecionadas
6. Calcula o erro de predição

Depois da execução da Fase 1, a Fase 2 pode ser executada por qualquer uma das duas versões: APS-SR-MATLAB ou APS-SR-CUDA. Na execução do APS-SR-CUDA, antes de iniciar a Fase 2, os dados são transferidos para a memória da GPU (*device*), coprocessador da *Central Processing Unit* (CPU ou *host*), onde é realizada a divisão de processos entre várias tarefas (*threads*) para serem executadas concorrentemente [3].

2.1 Paralelização para a Formulação das Regressões Sequenciais

O trecho de código entre as linhas 4 e 8 do Algoritmo 3 é implementado em paralelo empregando funções CUDA *kernel*. A cada execução de um *kernel*, o número de blocos e o número de *threads* por bloco são iguais a $\sqrt{N}$. O inteiro N é o número de linhas da matriz ou vetor de entrada para a função *kernel*. Quando $\sqrt{N}$ for um número decimal, o número de blocos e o número de *threads* por bloco são obtidos pela função teto ($\lceil\sqrt{N}\rceil$). Essa estratégia de implementação explora mais eficientemente os núcleos de processamento da GPU, pois evita a utilização de um único bloco com várias *threads* ou vários blocos com uma única *thread* [8, 9, 10].

Na solução do problema proposto, são utilizadas cinco funções *kernel*, não sendo necessário sincronizar as *threads* para obter um desempenho computacional melhor do que o apresentado por implementações tradicionais. O paralelismo é explorado somente para implementar as operações de adição, subtração e multiplicação matricial. Isto é, adição e subtração de vetores, multiplicação de matriz por vetor e multiplicação de vetor por escalar. Explora-se também o paralelismo para a cópia de elementos entre matrizes e vetores. A ideia da implementação paralela destas operações pode ser ilustrada por meio da operação vetorial $\mathbf{z} = \mathbf{v}_N + \mathbf{w}_N$. A soma dos vetores utiliza N *threads*, onde cada uma executa $\mathbf{z}_i = \mathbf{v}_i + \mathbf{w}_i$ da operação vetorial. O Algoritmo 3 mostra um pseudocódigo para a implementação APS-SR-CUDA.

Algoritmo 3: Implementação APS-SR-CUDA.

1. $L \leftarrow$ matriz onde cada coluna contém os índices das variáveis selecionadas na Fase 1
2. Transferência dos dados da memória do *host* para a memória do *device*
3. **para** $i = 1$ **até** K
4. $\quad lambdas \leftarrow L_i$ {cada *thread* copia um elemento da coluna i da matriz L}
5. $\quad x \leftarrow Xcal_{lambdas}$ {cada *thread* copia um elemento de cada coluna de $Xcal$}
6. $\quad$ Calcula $\beta_1^{(1)}$ utilizando a Equação (3)
7. $\quad$ Calcula $\beta_1^{(2)}$ e $\beta_2^{(2)}$ utilizando as Equações (5), (6), (7), (8) e (9)
8. $\quad$ Calcula $\beta_1^{(m)}$, ..., $\beta_m^{(m)}$, $m = 1, 2, \ldots, K$, utilizando as Equações (11), (12), (13) e (14)
9. **fim para**
10. Transferência dos dados da memória do *device* para a memória do *host*
11. Seta os parâmetros de saída

3 Dados e Equipamentos

Os dados utilizados neste trabalho consistem de amostras integrais de trigo obtidas a partir de material vegetal de produtores canadenses. Os dados foram determinados no laboratório de pesquisa de grãos (*Grain Research Laboratory*) por 1090 espectros de reflectância no infravermelho próximo (NIR) de amostras inteiras de grãos de trigo, os quais foram utilizados como dados referenciais na conferência internacional de reflectância difusa, em 2008 (http://www.idrc-chambersburg.org/shootout.html).

Os algoritmos foram executados em um *desktop* com processador *Intel core* i7 2600 (3,4 GHz), 8 GB de memória RAM e uma placa gráfica $NVIDIA^{®}$ *GeForce* GTX 550Ti com 192 CUDA *cores* e 2 GB de memória configurada.

4 Resultados e Discussão

A Figura 1 mostra o tempo de execução para a Fase 2 do APS utilizando as implementações APS-SR-MATLAB e APS-SR-CUDA. A variável $N2$ é o número máximo de variáveis selecionadas para compor o modelo. Para $N2 = 100$, são realizadas regressões envolvendo de 1 até 100 variáveis. O tempo gasto aumenta de acordo com $N2$, porém o aumento é menos acentuado para o APS-SR-CUDA. Vale ressaltar que o número (N) de linhas da matriz $\mathbf{X}$, que é parâmetro de entrada para as funções *kernel*, é fixo para toda execução. A variação ocorre apenas com o número ($N2$) de variáveis selecionadas para fazer parte do modelo de calibração.

A partir do gráfico da Figura 1, observa-se uma tendência de crescimento para o desempenho do algoritmo proposto com o crescimento do número de variáveis. Espera-se que essa tendência seja mantida para conjunto de dados cada vez maiores. A Tabela 1 apresenta a comparação de tempo computacional entre as implementações.

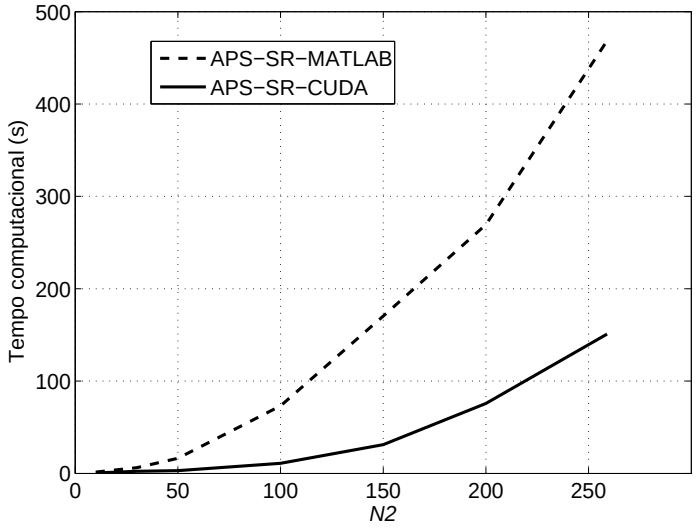

Figure 1: Comparação de desempenho computacional entre APS-SR-MATLAB e APS-SR-CUDA.

Table 1: Tempo computacional (em segundos) para APS-SR-MATLAB, APS-SR-CUDA e Soares [1].

Algoritmo	Número de variáveis		
	100	200	250
APS-SR-MATLAB	73,13	269,33	468,42
APS-SR-CUDA	19,88	89,80	170,87
Soares [1]	110,00	400,00	700,00

5 Conclusões

Este trabalho propôs as implementações APS-SR-MATLAB e APS-SR-CUDA. Ambas implementações são baseadas na proposta de Soares [1], que utiliza a estratégia de regressões sequenciais para a Fase 2 do APS. O APS-SR-CUDA é parcialmente paralelizado utilizando uma GPU com a tecnologia CUDA, por meio da linguagem computacional CUDA-C, e é capaz de reduzir o tempo computacional do algoritmo proposto por Soares [1]. Comparando com o tempo dos algoritmos, observa-se que as implementações SR são mais eficientes que as implementações tradicionais. Por exemplo, para $N2 = 250$, a implementação de Soares [1] e o APS-SR-CUDA

DOI: 10.5540/03.2015.003.01.0094

executam em torno de 700 e 171 segundos, respectivamente. Foi possível observar que o APS-SR-CUDA é, em média, 4x mais rápido.

Agradecimentos

Os autores agradecem à CAPES, FAPESP e FAPEG pelo apoio fornecido. Esta é uma contribuição do Instituto Nacional de Ciências e Tecnologias Analíticas Avançadas (INCTAA) (CNPq - proc. no. 573894/2008-6 e FAPESP proc. no. 2008/57808- 1).

References

[1] A. S. Soares, A. R. Galvão Filho, R. K. H. Galvão, M. C. U. Araújo, 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, *Journal of the Brazilian Chemical Society*, 21 (2010) 760-763.

[2] A. S. Soares, Detecção e diagnóstico de falhasempregando técnicas de classificação de padrões com seleção de atributos, *Tese de doutorado*, Institito Tecnlogico de Aeronáutica, São José dos Campos, 2010. (2010) 760-763.

[3] $CUDA^{TM}$, "NVIDIA CUDA C Programming Guide", NVIDIA Corporation, 5.0, 2013.

[4] D. C. Montgomery, E. A. Peck, G. G. Vining, "Introduction to Linear Regression Analysis", Wiley Series in Probability and Statistics, 2012.

[5] H. Martens, "Multivariate Calibration", John Wiley & Sons, 1991.

[6] J. M. Cortina, Interaction, nonlinearity, and multicollinearity: Implications for multiple regression, *Journal of Management*, 19 (1994) 915-922.

[7] K. R. Beebe, "Chemometrics: a practical guide", Wiley New York, 1998.

[8] L. C. M. Paula, Implementação paralela do método bicgstab(2) em gpu usando cuda e matlab para solução de sistemas lineares, *Revista de Sistemas e Computação*, 3 (2013) 125-131.

[9] L. C. M. Paula, A. S. Soares, T. W. Soares, W. S. Martins, A. R. G. Filho, C. J. Coelho, Partial parallelization of the successive projections algorithm using compute unified device architecture, em International Conference on "Parallel and Distributed Processing Techniques and Applications" pp. 737-741, Las Vegas, USA, 2013.

[10] L. C. M. Paula, A. S. Soares, T. W. Soares, A. C. B. Delbem, C. J. Coelho, A. R. G. Filho, Parallelization of a Modified Firefly Algorithm using GPU for Variable Selection in a Multivariate Calibration Problem, *International Journal of Natural Computing Research*, 4 (2014) 31-42.

[11] Mathworks, Introducing mex-files, http://www.mathworks.com/help/matlab/, 2011.

[12] M. C. U. Araújo, T. C. Saldanha, R. K. Galvão, T. Yoneyama, The successive projections algorithm for variable selection in spectroscopic multicomponent analysis, *Chemometrics and Intelligent Laboratory Systems*, 57 (2001) 65-73.

[13] S. F. C. Soares, A. A. Gomes, M. C. Araújo, R. K. Galvão, A. R. G. Filho, The successive projections algorithm, *TrAC Trends in Analytical Chemistry*, 42 (2013) 84-98.

[14] S. Patra, "Variable Selection In Categorical Regression Models: Theory And Applications", Lambert, 2013.

