Implementações paralelas para o algoritmo Online Sequential Extreme Learning Machine aplicadas a Previsão de concentração de Material Particulado no ar

Luís Fernando L. Grim[1], **Andrés Bueno**[1], **André Leon S. Gradvohl**[1]

[1]Faculdade de Tecnologia (FT) – Universidade Estadual de Campinas (UNICAMP)
R. Paschoal Marmo, 1888 – CEP 13484-332 – Limeira – SP

lfgrim@hotmail.com, andresbueno@gmail.com, gradvohl@ft.unicamp.br

Abstract. *In this work, we proposed two implementations for the Online Sequential Extreme Learning Machine algorithm in C programming language, one with OpenBLAS and another with MAGMA, both open source libraries. The goal is to compare the performance – forecasting error and real execution time – between the implementations when forecasting concentrations of particulate matter in the air. The results showed that the update block size of the algorithm influences the execution time of each implementation differently.*

Resumo. *Neste trabalho propomos duas implementações para o algoritmo Online Sequential Extreme Learning Machine em Linguagem C, uma com OpenBLAS e outra com MAGMA, ambas bibliotecas de código aberto. O objetivo é comparar o desempenho – erro de previsão e tempo real de execução – entre as implementações, ao prever concentrações de material particulado no ar. Os resultados mostraram que o tamanho do bloco de atualização do algoritmo influencia o tempo de execução de cada implementação de maneira distinta.*

1. Introdução

A previsão da concentração de poluentes atmosféricos têm chamado a atenção de autoridades públicas e de agências ambientais devido à ocorrência de vários episódios de poluição em áreas urbanas no mundo todo. As altas concentrações de poluentes causam efeitos nocivos para a saúde e até mortes prematuras entre grupos sensíveis, como crianças, idosos e pessoas com doenças respiratórias e cardíacas [CETESB 2016].

Nós consideramos os dados das concentrações de poluentes atmosféricos como Séries Temporais porque correspondem a uma coleção de valores obtidos periodicamente. Além disso, também os consideramos como Fluxos de Dados, uma vez que são uma sequência ordenada de amostras que chegam em alta velocidade. Considerando essas características, os sistemas que operam em fluxos de dados não podem lidar com muitas amostras na memória principal por um longo período, pois sofrer uma sobrecarga. Portanto, precisamos de processamento em tempo real para lidar com fluxos de dados.

Os fluxos de dados são dinâmicos por natureza e podem apresentar Desvios de Conceito, que consistem na alteração da relação entre os dados de entrada e a variável alvo a ser prevista ao longo do tempo [Gama et al. 2014]. Essa natureza dinâmica pode comprometer o desempenho de um modelo ao prever novas amostras recebidas. As abordagens de aprendizagem *online*, como o algoritmo "Online Sequential Extreme Learning Machine" (OS-ELM) [Liang et al. 2006], podem lidar com cenários de desvios de

conceito, devido à capacidade de aprender novos padrões de dados ao longo do tempo, tornando-se adequadas para o processamento de fluxos de dados.

Neste trabalho propomos implementações paralelas do OS-ELM, em Linguagem C, para executar em CPUs *multicore* e em arquiteturas híbridas – CPU + Unidades de Processamento Gráfico (GPUs). Portanto, nosso objetivo é comparar o desempenho dessas implementações ao prever concentrações de Material Particulado (MP) no ar.

2. Trabalhos relacionados

Em [Akusok et al. 2015] os autores apresentaram uma ferramenta de aprendizado *batch* chamada "High-Performance Extreme Learning Machines" (HP-ELM). O HP-ELM possui uma versão que acelera o seu treinamento através da biblioteca "Matrix Algebra on GPU e Multicore Architectures" (MAGMA) [Tomov et al. 2010]. Os resultados mostraram que a aceleração na GPU beneficia casos com mais de 1.000 neurônios ocultos.

Seguindo o mesmo raciocínio, [Krawczyk 2016] propôs uma abordagem que acelera o algoritmo OS-ELM através de GPUs, utilizando o ambiente R com RMOA, a biblioteca cuBLAS [NVIDIA 2012] e os seguintes parâmetros de entrada: de 10 a 100 neurônios ocultos com função de ativação sigmoide e blocos de atualização de 2.500 amostras. Os resultados mostraram que a simples delegação das operações matriciais para a GPU reduziu o tempo de processamento em quase 10 vezes.

Com base nos trabalhos apresentados, vemos que a aceleração em GPUs do HP-ELM e do OS-ELM mostrou resultados promissores para os parâmetros adotados. Em nosso trabalho decidimos explorar implementações distintas do OS-ELM com paralelismo em diferentes arquiteturas. Usando diferentes parâmetros de entrada, pretendemos verificar para cada caso, qual das implementações apresenta melhor desempenho.

3. Implementações propostas

Nós desenvolvemos duas implementações em Linguagem C: OS-ELM com OpenBLAS (OS-ELMob) e OS-ELM com MAGMA (OS-ELMmgm). Essas bibliotecas nos permitem realizar as operações matriciais usando as funções "Basic Linear Algebra Subprograms" (BLAS) e "Linear Algebra PACKage" (LAPACK). O OpenBLAS fornece funções que exploram os recursos *multithread* em processadores *multicore*, o que também permite o paralelismo em arquiteturas de multiprocessadores. O MAGMA fornece funções paralelas em arquiteturas heterogêneas/híbridas para sistemas *Multicore* + GPU.

A diferença entre as implementações reside na nomenclatura e nos parâmetros das respectivas funções, e também pela necessidade de explicitar transferências entre as memórias de CPU e GPU no MAGMA. Neste trabalho, restringimos as implementações propostas aos casos de regressão com a função de ativação sigmoide ($g(x) = \frac{1}{(1+e^{(-x)})}$).

3.1. O conjunto de dados e os passos de pré-processamento

O conjunto de dados ambientais utilizado nos experimentos contém amostras horárias de concentração de MP_{10} coletadas sequencialmente de 01 de Janeiro de 1998 a 23 de Novembro de 2017, da estação automática Cubatão – Vila Parisi. Essas concentrações são medidas em micrômetros por metro cúbico ($\mu m/m^3$) e foram obtidos através do sistema QUALAR, da CETESB. O conjunto de dados compreende 174.397 amostras, apresenta

zero como o valor mínimo, 1.470 $\mu\mathrm{m}/\mathrm{m}^3$ como o valor máximo e 8.011 amostras faltantes. Para mitigar os efeitos dessas amostras faltantes, preenchemos com o valor mais provável, usando o "Amelia II" [Honaker et al. 2011] e a interpolação linear.

Nós fizemos uma análise de *outliers* considerando os valores máximos de MP_{10} reportados pela CETESB para a estação de monitoramento Cubatão – Vila Parisi, normalizando cerca de 964 amostras acima de 350 $\mu\mathrm{m}/\mathrm{m}^3$. Em seguida, utilizamos o filtro Hampel para a detecção e substituição dos *outliers* remanescentes no conjunto de dados. Posteriormente, nós normalizamos os dados entre $[0, 1]$, como sugerido por [Huang 2013], pela normalização min-max. Ao final, organizamos amostras horárias de tal forma que a amostra atual e os últimos cinco instantes de tempo, $x_n = [s_{n-5}, \ldots, s_{n-1}, s_n]$, são usados para prever o próximo instante $y = [s_{n+1}]$. Em outras palavras, a atual e as últimas cinco concentrações medidas por hora são usadas para prever a concentração da próxima hora.

3.2. Configurações das implementações OS-ELM

Nós testamos casos com 100 neurônios ocultos e com blocos de atualização de 10, 100, 1.000, 2.500 e 5.000 instâncias. Para o treinamento inicial, usamos as 5.000 instâncias mais antigas e um fluxo de dados foi simulado com as 169.397 instâncias restantes.

Realizamos todos os testes no *cluster* GPU do HPI Future SOC Lab. Utilizamos um ambiente com 32 *cores* Intel (R) Xeon (R) CPU E5-2620 v4 @ 2,10GHz; 128 GB de RAM; 6 Tesla K80 @ 824 MHz e 11.440 MB de memória; e S.O. Ubuntu 16.04.2 64 bits. Instalamos a biblioteca OpenBLAS 0.2.20 com a configuração padrão, o que limitou a execução de nossas implementações em 16 *threads*. Também instalamos a biblioteca MAGMA 2.2.0, compilada para CUDA 8.0 e OpenBLAS 0.2.20. Ressaltamos que, por enquanto, o OS-ELMmgm está restrito à utilização de apenas uma GPU.

4. Resultados Experimentais

Os experimentos foram repetidos por 50 vezes e para cada caso testamos o desempenho considerando as médias e os desvios-padrão. Avaliamos as Raízes do Erro Quadrático Médio (RMSE) entre as saídas reais e previstas do fluxo simulado e também os Tempos Reais (TR) da execução de todo o algoritmo. Os resultados são exibidos na Tabela 1, conforme variamos o Tamanho do Bloco (TB) de atualização.

Tabela 1. Comparação dos resultados para cada caso testado

TB	OS-ELMob		OS-ELMmgm	
	RMSE	TR	RMSE	TR
1	0,1080 ± 0,00003	9,90 ± 0,47	0,1080 ± 0,00004	134,95 ± 0,70
10	0,1080 ± 0,00003	2,48 ± 0,02	0,1080 ± 0,00005	17,13 ± 0,07
100	0,1080 ± 0,00004	2,29 ± 0,02	0,1080 ± 0,00005	4,27 ± 0,02
1.000	0,1081 ± 0,00005	10,31 ± 0,07	0,1081 ± 0,00005	5,09 ± 0,03
2.500	0,1081 ± 0,00003	34,92 ± 0,15	0,1081 ± 0,00004	12,08 ± 0,08
5.000	0,1081 ± 0,00005	105,33 ± 1,43	0,1081 ± 0,00005	33,08 ± 0,17

Os resultados da Tabela 1 indicam que o RMSE é praticamente o mesmo em todos os casos. A Tabela 1 também mostra que o OS-ELMob apresentou melhores TRs para os casos com TBs de até 100 instâncias, executando cerca de 13,5 vezes mais rápido do que

OS-ELMmgm para o caso com TB=1. No entanto, o OS-ELMmgm demonstra ser superior, relacionado ao TR, para os casos com TBs a partir de 1.000 instâncias, executando cerca de 3,2 vezes mais rápido que o OS-ELMob quando TB=5.000. Cabe mencionar que ainda há espaço melhora na versão em CPU (OS-ELMob) utilizando bibliotecas e compiladores do fabricante da CPU, no nosso caso "Intel MKL" e "Intel C++ Compiler".

5. Conclusões

Os resultados mostraram que o TB é determinante para os TRs do OS-ELMob e do OS-ELMmgm. Casos com TBs de até 100 instâncias, foram melhores com o OS-ELMob. Esse comportamento está relacionado ao núcleo *online* do OS-ELM, pois a complexidade de algumas operações matriciais no seu núcleo está relacionada diretamente ao TB.

Quanto menor o TB, menor a complexidade dessas operações. Por outro lado, quanto menor o TB, maior a quantidade de iterações no núcleo *online*, o que aumenta o número de chamadas de função (muitas chamadas para operações de pouca complexidade). Isso faz com que a sobrecarga de comunicação entre as memórias de CPU e GPU nas funções híbridas do MAGMA seja considerável, tornando o seu uso desvantajoso.

Mesmo em cenários com maiores taxas de dados recebidos, e. g. 1.000 instâncias por segundo, as implementações OS-ELMob e OS-ELMmgm, com os parâmetros abordados aqui, são adequadas para o processamento do fluxo de dados, já que o pior caso (OS-ELMmgm com TB=1) levou apenas 134,95 segundos para processar 174.937 instâncias, ou seja, 0,0007 segundos por instância. Devido a isso, também podemos considerar como trabalhos futuros a execução de ambas as implementações em *ensembles* distribuídos.

Referências

Akusok, A., Bjork, K. M., Miche, Y., and Lendasse, A. (2015). High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications. *IEEE Access*, 3:1011–1025.

CETESB (2016). Qualidade do ar no estado de Sao Paulo 2015. Technical report, CETESB, São Paulo.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *Computing Surveys*, 46(4):1–37.

Honaker, J., King, G., and Blackwell, M. (2011). Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47.

Huang, G.-B. (2013). MATLAB Codes of ELM Algorithm.

Krawczyk, B. (2016). GPU-accelerated extreme learning machines for imbalanced data streams with Concept Drift. *Procedia Computer Science*, 80:1692–1701.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423.

NVIDIA (2012). CUDA Toolkit 4.2 CUBLAS Library.

Tomov, S., Dongarra, J., and Baboulin, M. (2010). Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Computing*, 36(5-6):232–240.