CUDA-ACCELERATED FEATURE SELECTION

Sterling Ramroach[1*], Jonathan Herbert[2] and Ajay Joshi[3]

[1,3]Department of Electrical and Computer Engineering,
[2]Department of Computing and Information Technology,
The University of the West Indies at Saint Augustine.
[1]Email: sterling.ramroach@sta.uwi.edu *(Corresponding author)
[2]Email: jonathan.herbert@my.uwi.edu
[3]Email: ajay.joshi@sta.uwi.edu

Abstract*:* Identifying important features from high dimensional data is usually done using one-dimensional filtering techniques. These techniques discard noisy attributes and those that are constant throughout the data. This is a time-consuming task that has scope for acceleration via high performance computing techniques involving the graphics processing unit (GPU). The proposed algorithm involves acceleration via the Compute Unified Device Architecture (CUDA) framework developed by Nvidia. This framework facilitates the seamless scaling of computation on any CUDA-enabled GPUs. Thus, the Pearson Correlation Coefficient can be applied in parallel on each feature with respect to the response variable. The ranks obtained for each feature can be used to determine the most relevant features to select. Using data from the UCI Machine Learning Repository, our results show an increase in efficiency for multi-dimensional analysis with a more reliable feature importance ranking. When tested on a high-dimensional dataset of 1000 samples and 10,000 features, we achieved a 1,230-time speedup using CUDA. This acceleration grows exponentially, as with any embarrassingly parallel task.

Keywords: *CUDA, Feature Selection, High Performance Computing, Pearson Correlation.*

https://doi.org/10.47412/JUQG5057

1. Introduction

Processing high dimensional data is often computationally intense and timely. In most cases, high dimensional data is noisy and contains random features that may not be relevant or useful. Feature selection is the process by which relevant or important features are extracted from the total set of features. Noisy, irrelevant and redundant features are removed to reduce the size of the data, which directly reduces processing time [1]. Feature selection algorithms can be sorted into three categories: supervised, semi-supervised, and unsupervised [1].

The Compute Unified Device Architecture (CUDA) is a parallel computing platform and programming model developed by NVIDIA for general computing on Graphical Processing Units (GPUs) [2]. This allows for applications which take advantage of the Central Processing Unit (CPU) for sequential tasks and the GPU for parallelisable tasks. For a CUDA program, it executes data-parallel functions called kernels on a set of threads referred to as a grid [3]. Each grid consists of blocks and each block consists of threads. These threads are identified by their block id and thread id based on the gridDim and blockDim variables that are populated by CUDA [3].

In recent times, high-throughput experiments in various fields have led to a significant increase in the volume and dimensionality of data produced. In Biology, genome sequencing and micro-array techniques now produce datasets with more than 20,000 columns. This rate of increase of data production is also seen across many other fields such as graph data generated by social networks or finance data generated by the stock markets [5]. There is therefore a need for more reliable, cost-effective, and faster data mining techniques and machine learning models which use the data [6–9]. Useful features can be extracted from the data to reduce dimensionality by removing redundant or noisy features. Reducing the number of features lessens the amount of data to be analysed and in some cases improves the classification accuracy of machine learning models [10]. The inclusion of noisy and irrelevant features into any dataset corrupts the interpretation of that dataset. Thus, feature reduction also contributes to illustrating the core signals in the data. Thus, feature reduction also contributes to illustrating the core signals in the data. When feature reduction is performed on classification datasets, the fluctuation of each attribute is compared against the target. Using Pearson Correlation [11-13] as the feature selector, all attributes can be treated independently. This technique is widely used for feature selection [11, 12]. Thus, this task is perfectly suited to GPU computing [14].

2. Method

The feature selector of choice is the Pearson Correlation Coefficient [13, 15]. Similar to a Naive Bayes classifier, the Pearson Correlation Coefficient treats all attributes as independent. A coefficient is generated for each feature and via a user-defined threshold, the features with high coefficients are considered to be more useful than those with lower coefficients. The population (Pearson) correlation is defined by the following equations.

$$\rho_{x,y} = \frac{cov(x,y)}{\sqrt{var(x)}\sqrt{var(y)}} \tag{1}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2}$$

$$var(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{3}$$

$$cov(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{4}$$

Given that we are using equation 1 on a sample, we will have to use the uncorrected sample estimators for the covariance and variances to determine the correlation. They are defined as follows.

$$r_{x,y} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{5}$$

$$= \frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{\sqrt{n\sum_{i=1}^{n}x_i^2-(\sum_{i=1}^{n}x_i)^2}\sqrt{n\sum_{i=1}^{n}y_i^2-(\sum_{i=1}^{n}y_i)^2}} \tag{6}$$

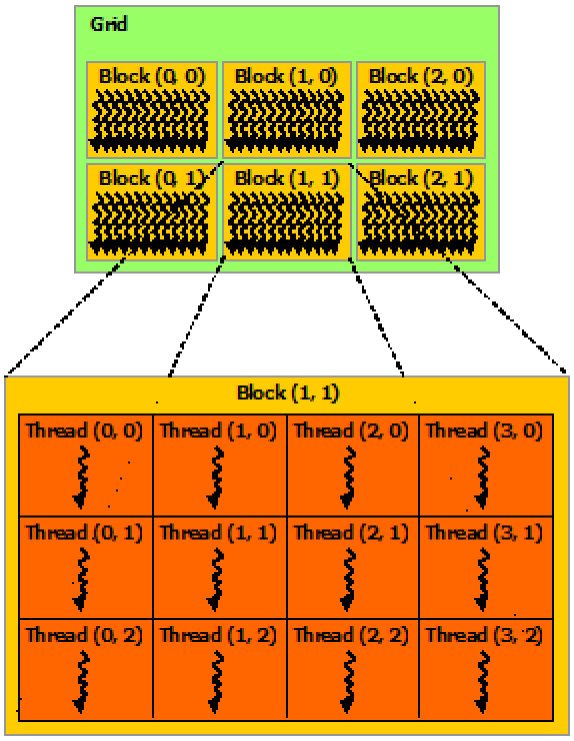

Figure 1: Grid of Thread Blocks [4]

Using equation 6, we can determine the correlation between the feature variable and the response (target). The top-K features are ranked, based on which has the highest correlation to the response. From this list of features, we can select the best feature subset to use. Table 1 shows how the Pearson correlation coefficient can be implemented for GPU computing via the CUDA framework. The GPU implementation requires the host to pass the data to the GPU via CUDA kernels which are executed in parallel. The majority of the speedup occurs at this stage. The __*global*__ identifier used tells the CUDA C++ compiler that the function runs on the GPU and can be called from CPU code. Functions identified by __*global*__ are known as kernels. These kernels along with any code that runs on the GPU is known as device code. Whereas code that runs on the CPU is called host code. The code presented in Table 1 also uses a variable *i*, which tells the function which thread to use to execute the computation. In our experiments, we let each feature's coefficient be calculated on its own thread. This is done simultaneously for all features using the architecture illustrated in Fig. 1.

In our experiments, we set out to show the extent to which CUDA can accelerate the Pearson Correlation algorithm. All experiments were performed on a 64-bit Windows 10 operating system with an Intel(R) Core (TM) i7-8750H CPU and an Nvidia GeForce GTX 1060 with 1,280 CUDA cores. All experiments were performed on randomly generated datasets which simulates real world data in terms of size and dimensionality.

Table 1: GPU Implementation

CUDA Feature Selection Function

```
__global__ void correlationCoefficient(const float *X, const float *Y, const int n, float* output){

    int sumX = 0, sumY = 0, sumXY = 0, sqSumX = 0, sqSumY = 0;

    int i = blockIdx.x * blockDim.x + threadIdx.x;

    if (i < n){
        sumX = sumX + X[i];
        sumY = sumY + Y[i];
        sumXY = sumXY + X[i] * Y[i];
        sqSumX = sqSumX + X[i] * X[i];
        sqSumY = sqSumY + Y[i] * Y[i];
    }

   output[i] = (float)(n * sumXY - sumX * sumY) / sqrt((float) ((n * sqSumX - sumX *
                  sumX)* (n * sqSumY - sumY * sumY)));
}
```

3. Results and Discussion

When using the method described in section 2, the results in Table 2 were obtained when selecting variables from a dataset of 1000 samples. The results were generated by running the experiments 10 times to calculate the average runtime +/- the standard deviation.

Table 2: CPU vs GPU average runtime and standard deviation

Number of Features K	CPU Runtime (s)	GPU Runtime (s)
1	0.8 +/- 0.75	0.16 +/- 0.12
10	1.2 +/- 0.40	0.16 +/- 0.12
100	2.8 +/- 0.98	0.16 +/- 0.12
1000	29 +/- 1.55	0.16 +/- 0.12
10000	246 +/- 10.83	0.20 +/- 0.12

We note that for a dataset of size 2 x 1000 (1 feature column and 1 target column), the CPU calculates the Pearson correlation coefficient in half of the time taken by the GPU. This is a direct result of the overhead of the memory accesses and transfers incurred by using the GPU. However, it is not feasible to use a feature selector on a dataset with only 1 dimension. It was only included to provide a benchmark for higher dimensional datasets. As the number of features grow by a factor of 10, so to does the time taken by the CPU to calculate all coefficients. When compared to the GPU implementation, the time is constant until the number of features approach 10,000. The initial overhead incurred by the setting up of the GPU to compute these coefficients is justified when the number of features is greater than 1.

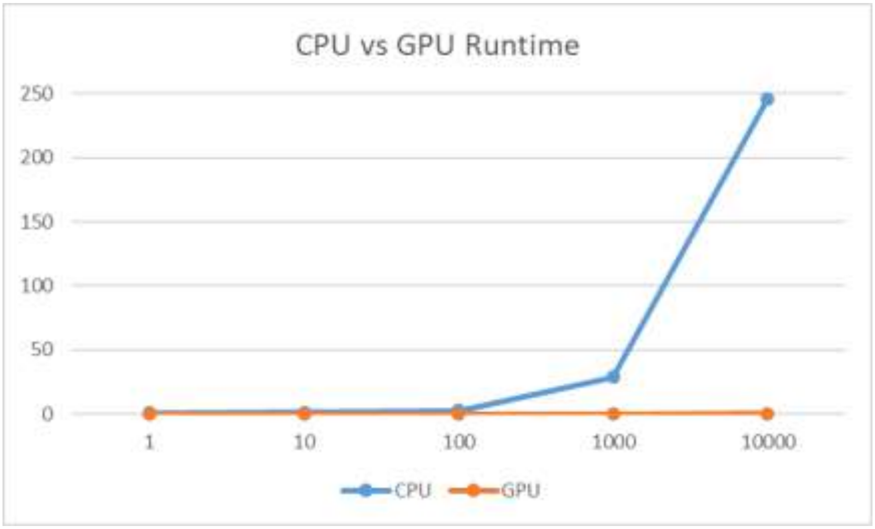

Figure 2: CPU vs GPU Runtime

Our parallel coefficient computing solution achieved a 1,230-time speedup from the conventional CPU implementation for a dataset with 10,000 features. For all experiments, the average speedup achieved by the GPU is 286.25. Figure 2 shows that as the number of features increases, the runtime of the program scales exponentially. For higher dimensional datasets, there can be significant performance improvements if a CUDA enabled GPU is used to accelerate feature selection. Feature selection via the Pearson correlation coefficient is an example of an embarrassingly parallel problem. Using parallel processing, significant acceleration can remedy this problem.

4. Conclusion

Feature selection removes noisy or irrelevant variables in high dimensional data. This reduces the processing time needed to build machine learning models. We use the Pearson Correlation Coefficient as a method for determining which features are highly correlated with a target and are thus most important in the data. Since we use the Pearson Correlation Coefficient, we can treat our features as independent of each other and use GPUs to process them individually. This accelerates the feature selection process for datasets with a large number of features. CUDA provides a framework to effectively use the compute power of a GPU for parallelisable general-purpose computing. A CUDA kernel can be defined to do the correlation calculations between features and the target. This effective use of a GPU gives a 286.25 times speedup on average across all of the experiments compared to the CPU execution and a max speedup of 1230 times compared to CPU execution.

References

[1] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.
[2] J. Nickolls, I. Buck, and M. Garland, "Scalable parallel programming," in *2008 IEEE Hot Chips 20 Symposium (HCS)*, pp. 40–53, IEEE, 2008.
[3] D. B. Kirk and W. H. Wen-Mei, Programming massively parallel processors: a hands-on approach. Morgan kaufmann, 2016.
[4] D. Guide, "Cuda c programming guide," NVIDIA, July, 2013.
[5] O. Soufan, D. Kleftogiannis, P. Kalnis, and V. B. Bajic, "Dwfs: a wrapper feature selection tool based on a parallel genetic algorithm," *PloS one*, vol. 10, no. 2,p. e0117988, 2015.
[6] M.-S. Yang and K. P. Sinaga, "A feature-reduction multi-view k-means clustering algorithm," *IEEE Access*, vol. 7, pp. 114472–114486, 2019.
[7] F. Li, Y. Yin, J. Shi, X. Mao, and R. Shi, "Method of feature reduction in short text classification based on feature clustering," *Applied Sciences*, vol. 9, no. 8, p. 1578,2019.
[8] S. Xu, X. Yang, H. Yu, D.-J. Yu, J. Yang, and E. C. Tsang, "Multi-label learning with label-specific feature reduction," *Knowledge-Based Systems*, vol. 104, pp. 52–61, 2016.
[9] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "Feature reduction and selection for emg signal classification," *Expert systems with applications*, vol. 39,no. 8, pp. 7420–7431, 2012.
[10] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.
[11] N. Nekuri, and M. Sookshma. "Adaptive Feature Selection and Classification Using Optimization Technique." In Frontiers in Intelligent Computing: Theory and Applications, pp. 146-155. Springer, Singapore, 2020.
[12] S. Eulalia, J. Kacprzyk, and P. Bujnowski. "Attribute Selection via Hellwig's Algorithm for Atanassov's Intuitionistic Fuzzy Sets." In Computational Intelligence and Mathematics for Tackling Complex Problems, pp. 81-90. Springer, Cham, 2020.
[13] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, pp. 1–4, Springer, 2009.
[14] J. Gonzalez-Domınguez, R. R. Exposito, and V. BolonCanedo, "Cuda-jmi: Acceleration of feature selection on heterogeneous systems," *Future Generation Computer Systems*, vol. 102, pp. 426–436, 2020.
[15] T. D. V. Swinscow, M. J. Campbell,et al.,Statistics at square one. *Bmj London*,2002.