A MACHINE LEARNING MODEL FOR AN EARTHQUAKE FORECASTING USING PARALLEL PROCESSING

Manoj Kollam[1*], Dr. Ajay Joshi[2]

[1,2]Faculty of Engineering, Department of Electrical and computer Engineering,
[1,2]The University of the West Indies, Trinidad
[1]Email: mkollam@gmail.com (corresponding author)*,
[2]Email: Ajay.Joshi@sta.uwi.edu

Abstract: Earthquake is a devastating natural hazard which has a capability to wipe out thousands of lives and cause economic loss to the geographical location. Seismic stations continuously gather data without the necessity of the occurrence of an event. The gathered data is processed by the model to forecast the occurrence of earthquakes. This paper presents a model to forecast earthquakes using Parallel processing. Machine Learning is rapidly taking over a variety of aspects in our daily lives. Even though Machine Learning methods can be used for analyzing data, in the scenario of event forecasts like earthquakes, performance of Machine Learning is limited as the data grows day by day. Using ML alone is not a perfect solution for the model. To increase the model performance and accuracy, a new ML model is designed using parallel processing. The drawbacks of ML using central processing unit (CPU) can be overcome by Graphic Processing unit (GPU) implementation, since the parallelism is naturally provided using framework for developing GPU utilizing computational algorithms, known as the Compute Unified Device Architecture (CUDA). The implementation of hybrid state vector machine (H-SVM) algorithm using parallel processing through CUDA is used to forecast earthquakes. Our experiments show that the GPU based implementation achieved typical speedup values in the range of 3-70 times compared to conventional central processing unit (CPU). Results of different experiments are discussed along with their consequences.

Keywords : *GPU, CUDA, Parallel Processing, Machine Learning, H-SVM*

https://doi.org/10.47412/DHHV5862

1. Introduction

Earthquakes (EQ) are natural hazards that don't happen frequently, and their unpredictability causes significant loss to human life and property. EQ forecasting/prediction is one of the most critical problem which attracted the consideration of numerous researchers. Earthquake forecast remained an unachieved objective due to some reasons.

One of the reasons is the lack of technology in accurately monitoring the stress changes, pressure and temperature variations deep beneath the crust through scientific instruments, which eventually results in unavailability of comprehensive data about seismic features. The second probable cause is the gap between seismologists and computer scientist for exploring the various venues of technology to hunt this challenging task. With the advent of modern computer science based intelligent algorithms, significant results have been achieved in different fields of research, such as weather forecasting [1], churn prediction [2] and disease diagnosis [3]. Therefore, by bridging gap between computer science and seismology, substantial outcomes may be achieved. The bright aspect of recent achievements is the encouraging results for earthquake prediction achieved through Computational Intelligence (CI) and Artificial Neural Networks (ANN) in combination with seismic parameters [7–12], thus initiating new lines of research and ideas to explore for earthquake prediction. There is a slight difference between earthquake prediction and earthquake forecasting.

Earthquake prediction problem is initially considered as a time series prediction [4] Later, seismic parameters are mathematically calculated on the basis of well-known seismic laws and facts, corresponding to every target earthquake (*Et*). The calculation of seismic parameters corresponding to every *Et* provides a feature vector related to *Et*. Thus, earthquake prediction is carried out on the basis of computed features in place of time series of earthquakes, thereby converting a time series prediction problem into a classification problem. The mathematically calculated seismic parameters are basically meant to represent the internal geological state of the ground before earthquake occurrence. This research work employs the known mathematical methods and computes all the seismic features in a bid to retain maximum information, which leads to sixty seismic features corresponding to every earthquake occurrence (*Et*). After acquiring maximum available seismic features, Maximum Relevancy and Minimum Redundancy (mRMR) based feature selection is applied to select most relevant feature having maximum information. Support Vector Machine (SVM). Some changes had been made to the SVM by adding parallelization to the SVM algorithm is applied to model relationship between feature vectors and their corresponding *Et*, thereby generating robust earthquake forecasting model (H-SVM).

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) [5]. 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 [4]. 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 [4]. Figure 1 shows an example of this organisation of threads and blocks in a grid.

High Performance Computing aims to solve the issues of Big Data by leveraging parallel computing methodologies. Parallel computing is used extensively in the field of engineering as an emerging powerhouse for data processing. This makes use of multiple compute units to solve a problem by breaking the problem up into discrete parts to be solved concurrently and in turn further splitting the problem into a series of instructions whereby each instruction is executed simultaneously on a different processor. There are two parallel programming topologies, namely multi-core CPU's and GPU's.

GPU's have multiple cores which when used together can perform a massive amount of computation. A GPU utilizes compute rather than control meaning that GPU's use simple control complexity and more compute power allowing for an embarrassingly amount of parallelization. The fundamental architecture of a GPU is much simpler than a CPU, as CPU's are complex processors and are widely used, GPU's are much simpler in design and as a result much more highly specialized. GPU's excel at heavy arithmetic, processing vast amounts of data, highly parallel data can be processed much faster.

The execution of a CUDA program can be decomposed by –

1. The input data to be acted upon is copied from the CPU's memory to the GPU's memory.
2. The data which is now copied to the GPU's memory is loaded and the GPU program is executed.

3. The results of the data processing from the GPU is copied from the GPU's memory to the CPU's memory.

The CUDA programming model is composed of three layers of abstraction for code execution on a GPU which are –

1. A hierarchy of group threads
2. Shared memories
3. Synchronization

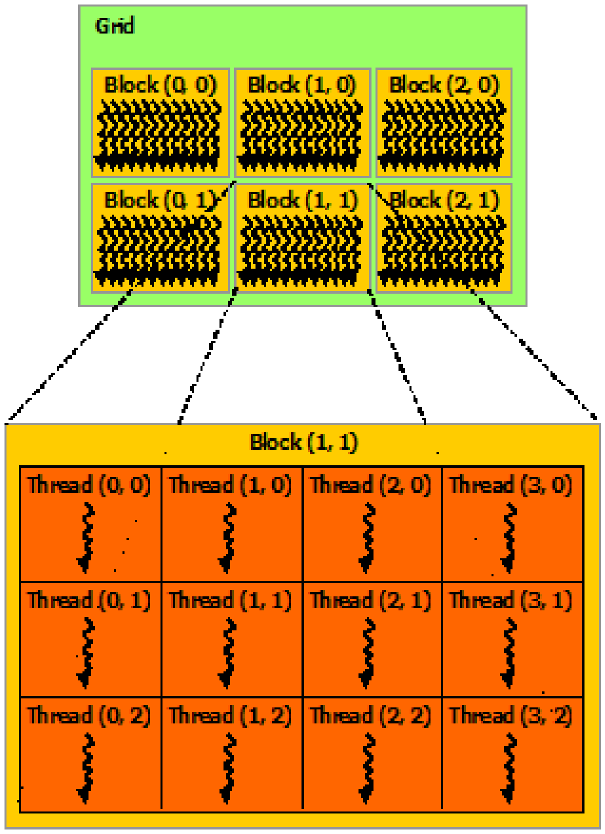

Figure 1: Grid of Thread Blocks [4].

2. Method

The earthquake catalog is the starting point of this process therefore, quality of catalog directly affects the prediction results. (Fig. 2) Further processes involved are feature calculation, selection, training of model, and finally predictions are obtained on unseen part of dataset. In the end performance of prediction model is evaluated and comparison is drawn.

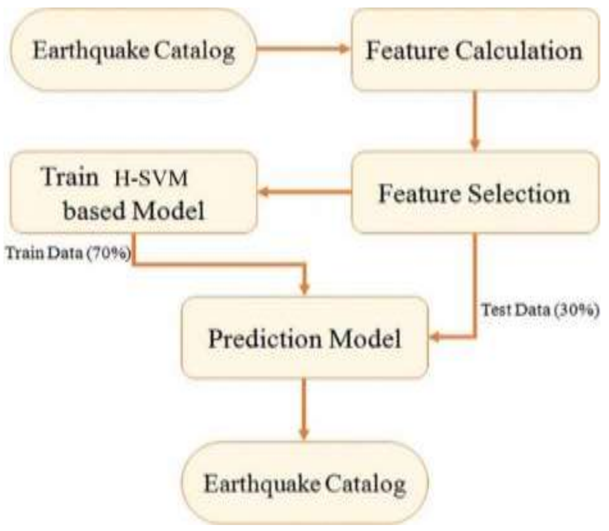

Figure 2: Flow chart of research methodology.

2.1 The Support Vector Machine

The analysis of SVM is a powerful technique that offers cutting-edge output for both classification and regression tasks (Vapnik 1995). SVM analysis is a powerful machine-learning tool. The performance of this approach involves the prevention of a real number through training parameters in the case of regression. The data collection has an overall impact on developing the model (Insom et al . 2015). We can solve the overfitting problem with SVR. In this analysis, vector regression (β-SVR), which is one of the prediction methods widely used in some recent papers, is considered.

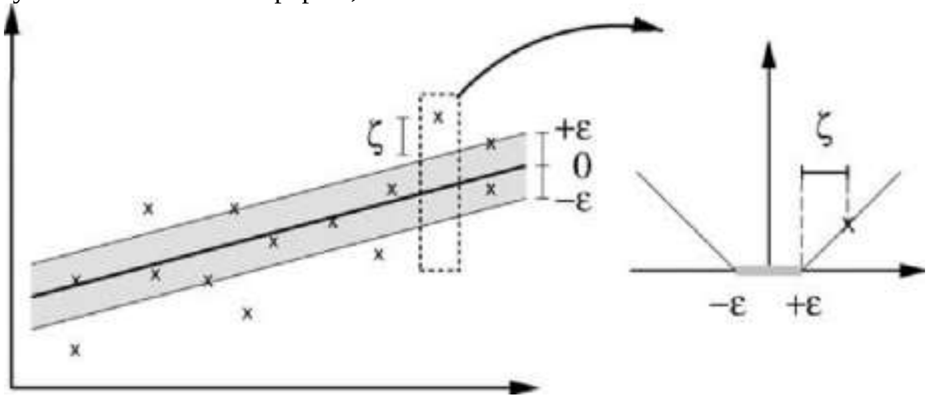

Fig 3: Specified deviation and slack variable in ε-SV regression [Adapted from (Smola and Schölkopf 2004)]

Given a set of training data vectors xi ∈ R n, i = 1, ..., m, of two classes and a label vector y such that yi ∈ {1,-1}, i = 1, ..., m, training a SVM for use in classification (C-SVC) is equivalent to solving the following primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, m. \end{aligned}$$

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}} & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ \text{subject to} & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, m \end{array}$$

where 1 is a vector of ones; Q is the m by m positive semidefinite kernel matrix, $Q_{ij} \equiv y_i y_j K(x_i , x_j)$; and $K(x_i , x_j) \equiv \varphi(x_i)$ T $\varphi(x_j)$ is the kernel function, the most popular of which is the Gaussian:

$$e^{-\lambda}||x_i - x_j||^2$$

Once the SVM has been trained, the following function is used to classify a new datapoint:

$$\mathrm{Sign}(\Sigma y_i \alpha_i K(X_i,X)+b)$$

3. Implementation

We describe an efficient SVM training algorithm on GPUs in this section. The following three steps of our algorithm are implemented on GPUs in a highly parallel way.

Select a working set: The process of calculating kernel values is the performance bottleneck in SVM training. We select a large working set using first-order heuristic for better GPU utilization [6] (Cotter, Srebro, and Keshet 2011). Naively using a large working set result in low training efficiency. Only a few are updated after the subproblem is solved. In other words, many rows of kernel matrix are wasted. To address this problem, we propose to use a working set buffer, like [7] (Vanek, Michalek, and Psutka 2017), but much simpler. Except for filling up whole working set in the first iteration, we select |W| 2 instances in the following iterations and replace the oldest instances in the working set. In this way, only |W| 2 rows of kernel matrix need to be calculated in each iteration. The use of working set buffer also leads to faster convergence, because the instances already in working set are more likely to be selected in next iterations.

Precomputation of kernel matrix: Instead of using cache, we precompute the rows of kernel matrix needed to solve the subproblem and store them in GPU memory. Note that only |W| 2 ×n kernel values need to be calculated because of working set buffer. With large working set, the precomputation is very efficient on GPUs.

Solving a subproblem: Given a working set and precomputed kernel matrix, the subproblem can be solved efficiently on GPUs. We implement the improved SMO[8] (Fan, Chen, and Lin 2005) to solve the subproblem. Each GPU thread is assigned to optimize one instance. After the subproblem is solved, an indicator variable will be transfer to CPU to check if the optimization problem is optimal

4. Experiments

It is important to assign three parameters prior to the PSVR training: a penalty parameter, C and β, and α, RBF. Different combinations of C, p and β are checked. The hunt for error-based parameters determines the best possible settings. A number of intervals are divided from each set of parameters. For each combination of these parameters, the workout is performed on each grid with multiple threads on the kernel. In order to increase SIMD (Single Instruction Multiple Data) performance, kernel function needs large samples. The CUDA-framework GPU enhances the evaluation of better parameters in order to boost model performance. We compare our implementation with LIBSVM with OpenMP, gtSVM (Cotter, Srebro, and Keshet 2011) and OHD-SVM (Vanek, Michalek, and Psutka 2017). GPUSVM{Catanzaro, 2008 #66} is excluded because gtSVM reports better results. The datasets are from USGS website(https://earthquake.usgs.gov/earthquakes/search/). We choose commonly used Gaussian kernel. The regularization parameter C and Gaussian kernel parameter γ are chosen by grid-search. The experiments were conducted in a workstation with GTX 1050Ti, 4 GB memory.

Table 1: CPU vs GPU average runtime and standard deviation. Training time (s) among different implementations of algorithm

Algorithm	**Number of Features**	**CPU Runtime (s)**	**GPU Runtime (s)**
Libsvm-omp	45	15	13
gstsvm	45	11	10
H-SVM	45	8	2

Within this paper, the efficiency and calculation speed are improved through the application of the GPU model and the comparison of model results with LIBSVM and Scikit Learn (Chang and Lin 2011).

The data set includes 15 rows with 67937 columns of data, used to compare the output between PSVR, Scikit Learn and Libsvm, for models training and testing, and show clearly that the length of PSVR is much smaller than that shown in Fig. 4.

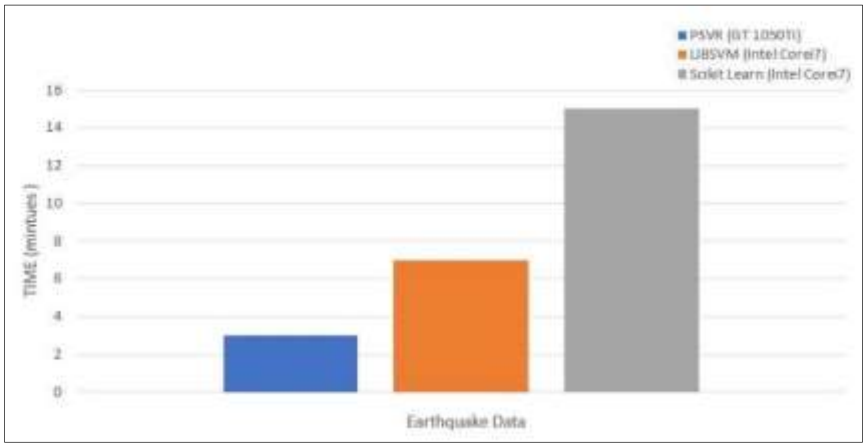

Fig 4: Time taken for training and testing of the models with Earthquake catalog

5. Conclusion

The results of this study indicate that the GPU with the CUDA system is measured with a good Parallel support vector regression. When the model is a training phase, device values are transmitted to the GPU from the CPU. The total acceleration of PSVR results in this. The model is well educated and validated with good results and it shows that the computational speed and accuracy are significantly improved and thus the overall performance of the model is enhanced. This paper presents the parallel SWR algorithm built for GPU and the CUDA system, and also enables us to use this realistic ML algorithm for the processing of real-world datasets, and interdisciplinary research is conducted for earthquake prediction by HPC.

References

[1] Nayak DR, Mahapatra A, Mishra P. A survey on rainfall prediction using artificial neural network. International Journal of Computer Applications. 2013;72(16).

[2] Idris A, Rizwan M, Khan A. Churn prediction in telecom using Random Forest and PSO based data balancing in combination with various feature selection strategies. Computers & Electrical Engineering. 2012;38(6):1808–19. 10.1016/j.compeleceng.2012.09.001.
[3] Cosma G, Brown D, Archer M, Khan M, Pockley AG. A survey on computational intelligence approaches for predictive modeling in prostate cancer. Expert systems with applications. 2017;70:1–19.
[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] Cosma G, Brown D, Archer M, Khan M, Pockley AG. A survey on computational intelligence approaches for predictive modeling in prostate cancer. Expert systems with applications. 2017;70:1–19..
[7] Chang, C.-C., and Lin, C.-J. 2011. Libsvm: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) 2(3):27.
[8] Cotter, A.; Srebro, N.; and Keshet, J. 2011. A gpu-tailored approach for training kernelized svms. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 805–813. ACM.
[9] Fan, R.-E.; Chen, P.-H.; and Lin, C.-J. 2005. Working set selection using second order information for training support vector machines. Journal of machine learning research 6(Dec):1889–1918.
[10] Joachims, T. 1998. Making large-scale svm learning practical. Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund..
[11] Platt, J. C. 1999. Fast training of SVMs using Sequential Minimal Optimization. In Advances in kernel methods. MIT Press. 185–208.
[12] Vanek, J.; Michalek, J.; and Psutka, J. 2017. A gpuarchitecture optimized hierarchical decomposition algorithm for support vector machine training. IEEE Transactions on Parallel and Distributed Systems.