Accelerating Parameter Estimation for Multivariate Self-Exciting Point Processes

Ce Guo
Department of Computing
Imperial College London
United Kingdom
ce.guo10@imperial.ac.uk

Wayne Luk
Department of Computing
Imperial College London
United Kingdom
w.luk@imperial.ac.uk

ABSTRACT

Self-exciting point processes are stochastic processes capturing occurrence patterns of random events. They offer powerful tools to describe and predict temporal distributions of random events like stock trading and neurone spiking. A critical calculation in self-exciting point process models is parameter estimation, which fits a model to a data set. This calculation is computationally demanding when the number of data points is large and when the data dimension is high. This paper proposes the first reconfigurable computing solution to accelerate this calculation. We derive an acceleration strategy in a mathematical specification by eliminating complex data dependency, by cutting hardware resource requirement, and by parallelising arithmetic operations. In our experimental evaluation, an FPGA-based implementation of the proposed solution is up to 79 times faster than one CPU core, and 13 times faster than the same CPU with eight cores.

1. INTRODUCTION

The study of random processes is attracting attention from researchers and practitioners in various areas. These processes serve as powerful tools to analyse stochastic mechanisms. A self-exciting point process is a random process that describes the occurrences of repeatable momentary random events. This point process is particularly useful in modelling random events whose occurrence patterns do not follow obvious temporal distributions. Applications of self-exciting point process models include crime detection [6], high-frequency trading [1], earthquake analysis [4] and neurone spiking analysis [2].

We focus on self-exciting point process models for multivariate data. To build such a model, one needs to estimate an appropriate parameter set from data. This calculation is computationally expensive for a data set with a long occurrence sequence or high dimensionality. On the other hand, to adapt a model to a rapidly changing environment, one needs to repeat the estimation frequently to incorporate newly arrived data. As a result, an acceleration solution for this parameter estimation problem is in great demand.

FPGA'14, February 26–28, 2014, Monterey, CA, USA.
Copyright 2014 ACM 978-1-4503-2671-1/14/02 ...$15.00.
http://dx.doi.org/10.1145/2554688.2554765.

Reconfigurable computing is a promising acceleration technology for this problem, but the best acceleration solution we know in existing work [3] can only handle univariate data. It is challenging to design a solution for the multivariate case. The most time-consuming calculation in parameter estimation, log-likelihood estimation, involves complex data dependency. It is necessary and challenging to develop novel and hardware-specific mathematical methods to eliminate this data dependency.

In this paper, we address this challenge and propose the first acceleration solution to the parameter estimation problem. The major contributions of this paper are as follows.

- An acceleration strategy for log-likelihood evaluation which eliminates complex data dependency.
- A pipelined hardware accelerator for maximum likelihood estimation based on the proposed strategy.
- An implementation of the proposed architecture on a commercial FPGA acceleration card.

2. SELF-EXCITING POINT PROCESSES

In this section we provide a short introduction to multivariate self-exciting point processes and the parameter estimation problem.

A sequence of random variables, $\{t_i\} = (t_1, t_2, t_3, \ldots)$, is a *univariate point process* if and only if $t_i > 0$ and $t_i < t_{i+1}$ for all $i \in \mathbb{N}^+$. The *intensity* of a point process $\{t_i\}$ at a time point t is mathematically defined by

$$\lambda(t) = \lim_{h \to 0^-} E\Big(\frac{C(t+h) - C(t)}{h}\Big) \tag{1}$$

where

$$C(t) = \sum_{t_i \le t} 1_{i \in \mathbb{N}^+} \tag{2}$$

A *multivariate point process* with M dimensions is a collection of M univariant point processes. In this research we focus on the *multivariate self-exciting point process* whose intensity functions satisfy

$$\lambda_m(t) = \lambda_m^{\perp} + \sum_{m'=1}^{M} \int_0^t \alpha_{m,m'} e^{-\beta_{m,m'}(t-w)} \mathrm{d}C_{m'}(w) \tag{3}$$

The property of such a point process is uniquely given by a parameter set $\theta = \langle \Lambda^{\perp}, A, B \rangle$ where $\Lambda^{\perp} = \{\lambda_m^{\perp} : m \in [1..M]\}$, $A = \{\alpha_{m,m'} : (m, m') \in [1..M] \times [1..M]\}$ and $B = \{\beta_{m,m'} : (m, m') \in [1..M] \times [1..M]\}$.

Parameter estimation for self-exciting point process models is the computation of a reasonable parameter set that fits a model to a data set. This calculation is the critical part in various applications related to point processes.

Parameter estimation can be achieved by maximum likelihood estimation [7] where we compute at least one member of the following set:

$$\Theta_{\mathrm{MLE}} = \left\{\theta : (\forall\theta')[L(\theta, D(T)) \geq L(\theta', D(T))]\right\} \tag{4}$$

where $L(\theta, D(T))$ is the log-likelihood value calculated given a parameter set θ and a data set $D(T)$. The log-likelihood of a multivariate self-exciting point process can be computed by

$$L(\theta, D(t)) = \sum_{m=1}^{M} L_m = Q + R + S \tag{5}$$

where

$$Q = \sum_{m=1}^{M} \sum_{k=1}^{C_m(T)} \log \lambda_m(t_{m,k}) \tag{6}$$

$$R = T \sum_{m=1}^{M} \lambda_m^{\perp} \tag{7}$$

$$S = \sum_{m=1}^{M} h_m(T) \tag{8}$$

The search of parameters can be conducted by any general-purpose optimisation algorithm. However, no matter what algorithm is used, it needs to evaluate log-likelihood frequently. This evaluation process can be computationally expensive if the number of data points is large or the data dimension is high [3]. Hardware acceleration solution to the parameter estimation problem for multivariate self-exciting point processes has not been developed. There are only partial solutions [3, 5] based on simplified problem settings.

3. STRATEGY OF ACCELERATION

Log-likelihood evaluation is the most computationally expensive subroutine in parameter estimation. In this section, we describe an acceleration strategy for log-likelihood evaluation.

3.1 Log-Likelihood Evaluation

Each observation in the data set is recorded in the form $o_i = (t_i, m_i)$ where t_i is the occurrence time and m_i is the corresponding dimension label.

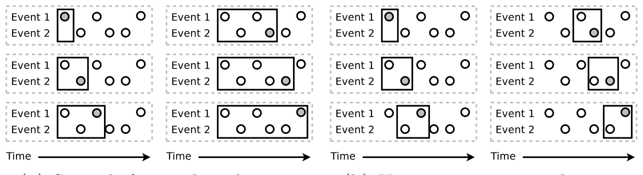
For ease of discussion, we denote the number of data points in the data set as N. If o_i corresponds to $t_{m,k}$, then we let $\lambda(o_i) = \lambda_m(t_{m,k})$. As we have discussed, a log-likelihood value can be computed by adding up three components by Equation (5). By Equation (7), we know that it is easy to compute R directly in O(M) time. $(S+Q)$ can be computed by

$$S + Q = \sum_{i=1}^{N} (s_i + q_i) \tag{9}$$

where

$$s_i = \sum_{m=1}^{M} \frac{\alpha_{m,m_i}}{\beta_{m,m_i}} (e^{-\beta_{m,m_i}(T-t_i)} - 1) \tag{10}$$

$$q_i = \log \lambda(o_k) \tag{11}$$



(a) Straightforward evaluation (b) Hyper-recursive evaluation

Figure 1: Data dependency pattern in different intensity evaluation methods

In other words, one may obtain $(S + Q)$ by adding up $(s_i + q_i)$ for all $i \in [1..N]$. The effort to compute $(S + Q)$ depends on the efficiency of intensity evaluation. We need to evaluate $\lambda(o_i)$ for all observations o_i for all $i \in [1..N]$. Moreover, to evaluate $\lambda(o_i)$, one needs to collect a piece of statistical information from o_1 to o_i. The data dependency pattern of this method is shown in Figure 1(a).

3.2 Hyper-Recursive Intensity Evaluation

We propose *hyper-recursive intensity evaluation* which enables the log-likelihood to be computed without complex data dependency. The key insight of this method is that we introduce a group of intermediate variables that (i) contain all necessary statistical information of all past points for intensity computation and (ii) can be computed with a hardware resource requirement that does not scale up with data size N.

Finding such a group of intermediate variables is a very difficult task from the perspective of statistics, information theory and algorithm design, because this variable set needs to be compact, informative and easy to compute. In this study, we manage to find a set of variables that meet all these requirements. We propose to use the following equations to calculate intensities in a hardware-friendly manner:

$$\lambda(o_i) = \lambda_{m_i}^{\perp} + \sum_{m'=1}^{M} \alpha_{m_i,m'} r_{i,m'} \tag{12}$$

where

$$r_{i,m'} = u_{i,m'} + v_{i,m'} \tag{13}$$

$$u_{i,m'} = e^{-\beta_{m^*,m'}(x_i - x_{\iota(i,m^*)})} \omega_{i,m',m_i} \tag{14}$$

$$v_{i,m'} = e^{-\beta_{m_i,m'}(x_i - x_{\iota(i,m_i)})} \psi_{i,m',m_i} \tag{15}$$

$$\iota(i, m^*) = \sup\{i' : m_{i'} = m^* \text{ and } i' < i\} \tag{16}$$

$$\omega_{i,m',m^*} = \begin{cases} r_{i-1,m'} & \text{if } m_{i-1} = m^* \\ \omega_{i-1,m',m^*} & \text{if } m_{i-1} \neq m^* \end{cases} \tag{17}$$

$$\psi_{i,m',m^*} = \begin{cases} 0 & \text{if } m_{i-1} = m^* \\ \psi_{i-1,m',m^*} + \tilde{\phi}_{i-1,m',m^*} & \text{if } m_{i-1} \neq m^* \end{cases} \tag{18}$$

$$\tilde{\phi}_{i-1,m',m^*} = \begin{cases} e^{-\beta_{m^*,m'}(x_{\iota(i,m^*)} - x_{i-1})} & \text{if } m_{i-1} = m' \\ 0 & \text{if } m_{i-1} \neq m' \end{cases} \tag{19}$$

Equation (14) to (19) suggest that the variables ω_{i,m',m^*} and ϕ_{i,m',m^*} for all $(m', m^*) \in [1..M] \times [1..M]$ provide sufficient information for intensity computation without involving complex data dependency.

We name this method *hyper-recursive intensity evaluation* because the intensity is obtained from a two-layer recursive computation. The mathematical derivation of this method will be provided in a future publication.

This method is an excellent candidate for hardware implementation. One reason is that the requirement for arithmetic resources scales linearly with the dimensionality of the problem M. Another reason is that the data dependency, shown in Figure 1(b), is not related to the size of the data set N. Moreover, updating operations for ω_{i,m',m^*} and ψ_{i,m',m^*} can easily be parallelised.

Hyper-recursive intensity evaluation is arguably the most general hardware-oriented algorithmic optimisation for parameter estimation, as there is no theoretical constraint in the number of data dimensions. An existing work about hardware-oriented adoption of univariate log-likelihood evaluation for self-exciting point process described in [3] is merely a special case of hyper-recursive intensity evaluation when $M = 1$. A detailed discussion of how our proposed method relates to existing work will be provided in a future publication.

Moreover, hyper-recursive intensity evaluation can be used with any likelihood-based parameter searching algorithm, because it produces the same results as the straightforward intensity evaluation.

4. PIPELINED ACCELERATOR FOR LOG-LIKELIHOOD EVALUATION

In this section, we develop a pipelined architecture for the hyper-recursive log-likelihood evaluation strategy described in the previous section.

4.1 Hardware Design

We propose a basic hardware module called the *intensity evaluation cell* to compute $r_{i,m'}$, ψ_{i,m',m^*} and ω_{i,m',m^*} for a data point o_i and a dimension m' following the hyper-recursive intensity evaluation strategy discussed in the previous section. In other words, an intensity evaluation cell does not provide the intensity value directly, but it collects statistical information from data so that the intensity value can easily be computed.

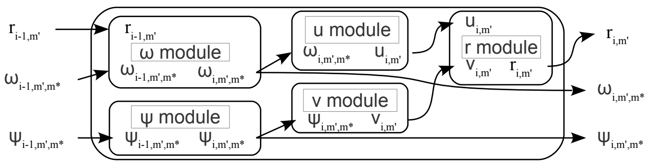


Figure 2: Intensity evaluation cell

The structure of the intensity evaluation cell is shown in Figure 2. The data input and the parameter set are omitted for brevity. As shown in the figure, an intensity evaluation cell contains five modules. The ω module and ψ module compute Equation (17) and (18) respectively for $\omega_{i,m',m*}$ and $\psi_{i,m',m*}$. The u module and v module take the outputs of the $\omega_{i,m',m*}$ and $\psi_{i,m',m*}$ respectively, and then evaluate Equation (14) and (15) to produce $u_{i,m'}$ and $v_{i,m'}$. The r module calculates $r_{i,m'}$ by adding up $u_{i,m'}$ and $v_{i,m'}$.

We design a pipelined accelerator to finalise the log-likelihood evaluation by combining multiple intensity evaluation cells. The structure of the accelerator is shown in Figure 3.

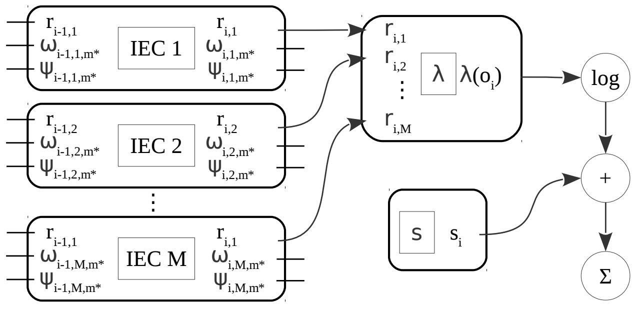


Figure 3: Log-likelihood evaluation accelerator

Each block marked with 'IEC' represents an intensity evaluation cell. In this architecture, we deploy M intensity evaluation cells to compute $r_{i,1}$, $r_{i,2}$, ..., $r_{i,M}$ for each data point o_i. These results are passed to a λ module which computes the intensity $\lambda(o_i)$ by Equation (12). In addition, we design the s module which computes s_i according to Equation (10). By accumulating the sum of $\log(o_i)$ and s_i in an accumulator, we can finally harvest the value of $(S + Q)$ by Equation (9).

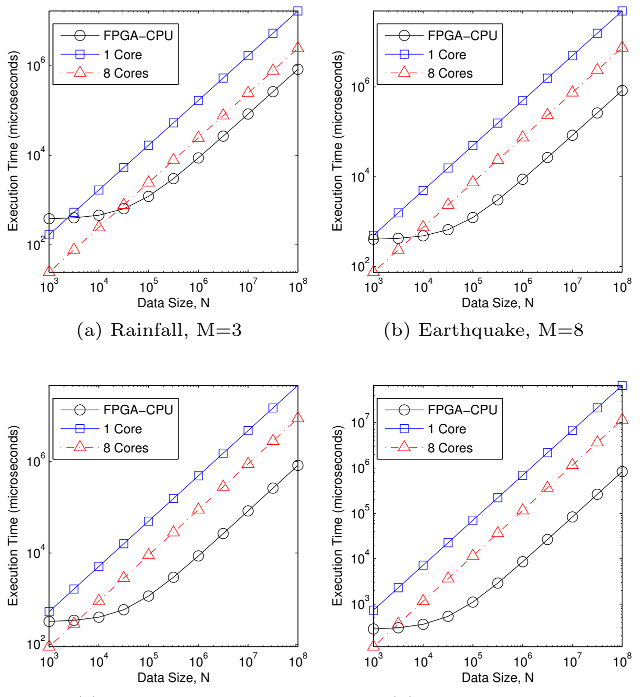
4.2 An FPGA Implementation

We build an FPGA implementation of the proposed accelerator. The hardware platform we use is a Maxeler MAX3 acceleration card. This card is equipped with a Xilinx Virtex-6 V6-SX475T FPGA. The card is installed in a host computer with eight Intel i7-870 CPU cores running at 2.93GHz and 16GB DDR3 memory. The acceleration card communicates with the host computer via an 8-lane PCI Express 2.0 interface. The hardware design is described in the MaxJ language and compiled to VHDL with Maxeler MaxCompiler.

The occurrence times of data points are represented in IEEE single precision floating point numbers. The dimension labels are represented in 8-bit unsigned integers. We validate the correctness and precision of the system by comparing its results with a MATLAB implementation using simulated data sets with known parameters. The maximum relevant error is less than 1.1 percent.

We deploy 14 intensity evaluation cells in the FPGA to maximise resource usage. This implementation takes around 71 percent fine-grained logic, 38 percent of DSPs and 45 percent of block memory on the FPGA. Although the fine-grained logic is the resource usage bottleneck in this particular implementation, we believe that the block memory consumption will become the bottleneck when the number of intensity evaluation cells increases. This is because such resource consumption theoretically grows quadratically with the number of intensity evaluation cells while the consumption of other resources grows linearly.

The clock frequency of the FPGA is set to 120MHz. The memory bandwidth of our acceleration platform is around 4GB/s. The actual consumption of our accelerator is only around 0.56GB/s. This bandwidth consumption only depends on the clock frequency and the number representation scheme. If these two factors do not change, the memory consumption will not change even if we deploy more intensity evaluation cells in the system.



(a) Rainfall, M=3

(b) Earthquake, M=8

(c) Neuron, M=10

(d) HighFreq, M=14

Figure 4: Experimental Results

5. EXPERIMENTAL EVALUATION

We use four data sets in the experiments, namely 'Rainfall', 'Earthquake' , 'Neuron' and 'HighFreq'. These data sets are sampled from four different real-world problems: 'Rainfall' is collected from three rainfall monitors; 'Earthquake' is based on the log of a micro-earthquake detector; 'Neuron' is extracted from a neuron spiking train data; and 'HighFreq' is adapted from a high-frequency currency trading log. The numbers of dimensions of these four data set are 3, 8, 10, and 14 respectively.

We build a CPU-only implementation that runs exclusively on the host computer of the acceleration card. This implementation is programmed with the OpenMP library. To make a fair comparison, we apply a series of optimisations to this implementation, including avoiding redundant memory access and selecting appropriate parallelisation scheme. Both the host code for the FPGA accelerator and the CPU-only implementation are written in the C programming language and compiled with the Intel C compiler with the highest compiling optimisation.

The performance measure of the experiments is the execution time of log-likelihood evaluation. This performance measure is independent of the selection of searching algorithm. Experimental results for log-likelihood evaluation are shown in Figure 4. We record the data size in log scale with base 10 in these figures. To reflect the trend of the increment of the execution time, we also record the execution time in log scale. Note that a small difference along the vertical axis in each figure means a huge difference in execution time due to the properties of the logarithm function.

The FPGA-CPU system demonstrates higher speedup for large data in general. The maximum speedup is achieved when $M = 14$ and the data size reaches 10^7. In this case, the FPGA implementation is respectively 79 times and 13 times faster than a single-core CPU and an eight-core CPU for log-likelihood evaluation. Moreover, we find that the FPGA-CPU implementation consumes 31 times less energy than the CPU-only implementation during the computation. A detailed analysis of the experimental results will be provided in a future publication.

6. CONCLUSION AND FUTURE WORK

This paper presents the first hardware acceleration solution to the parameter estimation problem of multivariate self-exciting point processes. We first develop an acceleration strategy for log-likelihood evaluation. The core method of the strategy is the hyper-recursive intensity evaluation which eliminates complex data dependency. This strategy enables log-likelihood values to be computed in a pipelined manner. By mapping the proposed strategy into reconfigurable hardware, we design an accelerator for log-likelihood evaluation which works for all likelihood-based parameter searching algorithms. This accelerator evaluates log-likelihood values efficiently, and its resource requirement does not grow with the data size. One possible direction of future work is to further improve the performance by applying application-specified optimisations. Another possible direction is to develop point process models which natively support hardware acceleration.

7. ACKNOWLEDGEMENT

This work is supported in part by the China Scholarship Council, by Maxeler University Programme, by the European Union Seventh Framework Programme under grant agreement number 257906, 287804 and 318521, by UK EPSRC, and by Xilinx.

8. REFERENCES

[1] C. G. Bowsher. Modelling security market events in continuous time: Intensity based, multivariate point process models. *Journal of Econometrics*, 141(2):876–912, 2007.

[2] R. Dahlhaus, M. Eichler, and J. Sandkühler. Identification of synaptic connections in neural ensembles by graphical models. *Journal of neuroscience methods*, 77(1):93–107, 1997.

[3] C. Guo and W. Luk. Accelerating maximum likelihood estimation for Hawkes point processes. In *International Conference on Field Programmable Logic and Applications*, 2013.

[4] Y. Y. Kagan. Statistical distributions of earthquake numbers: consequence of branching process. *Geophysical Journal International*, 180(3):1313–1328, 2010.

[5] S. W. Linderman and R. P. Adams. Discovering structure in spiking data. In *New England Machine Learning Day*, Cambridge, MA USA, 2013.

[6] G. O. Mohler, M. B. Short, P. J. Brantingham, F. P. Schoenberg, and G. E. Tita. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493):100–108, 2011.

[7] Y. Ogata. On Lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, 27(1):23–31, 1981.