# EFFICIENT INFERENCE ALGORITHMS FOR NETWORK ACTIVITIES

A Thesis
Presented to
The Academic Faculty

by

Long Quoc Tran

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
May 2015

# EFFICIENT INFERENCE ALGORITHMS FOR NETWORK ACTIVITIES

Approved by:


Professor Polo Chau,
Committee Chair
School of Computational Science and Engineering
*Georgia Institute of Technology*

Professor Hongyuan Zha, Advisor
School of Computational Science and Engineering
*Georgia Institute of Technology*

Professor Alexander G. Gray
School of Computational Science and Engineering
*Skytree Inc.*

Professor Le Song
School of Computational Science and Engineering
*Georgia Institute of Technology*

Professor Jimeng Sun
School of Computational Science and Engineering
*Georgia Institute of Technology*

Professor Haomin Zhou
School of Mathematics
*Georgia Institute of Technology*



Date Approved: 6$^{\text{th}}$ January 2015

*To the memory of my father,*

*Trần Trọng Cảnh,*

*and to my wife,*

*Hoàng Nam Phương.*

# ACKNOWLEDGEMENTS

I would like to thank all the people who helped and supported me with carrying out this research project.

Firstly, I would like to thank the committee members for their critical advices throughout the preparation of this dissertation.

Secondly, I would like to thank Professor Hongyuan Zha and Professor Alexander Gray for their excellent guidance from the beginning of my PhD program to its ending.

Thirdly, I would like to thank the members of my family for their endless support and encouragement.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

The real social network and associated communities are often hidden under the declared friend or group lists in social networks. We usually observe the manifestation of these hidden networks and communities in the form of recurrent and time-stamped individuals' activities in the social network. The inference of relationship between users/nodes or groups of users/nodes could be further complicated when activities are interval-censored, that is, when one only observed the number of activities that occurred in certain time windows. The same phenomenon happens in the online advertisement world where the advertisers often offer a set of advertisement impressions and observe a set of conversions (i.e. product/service adoption). In this case, the advertisers desire to know which advertisements best appeal to the customers and most importantly, their rate of conversions.

Inspired by these challenges, we investigated inference algorithms that efficiently recover user relationships in both cases: time-stamped data and interval-censored data. In case of time-stamped data, we proposed a novel algorithm called NetCodec, which relies on a Hawkes process that models the intertwine relationship between group participation and between-user influence. Using Bayesian variational principle and optimization techniques, NetCodec could infer both group participation and user influence simultaneously with iteration complexity being O((N+I)G), where N is the number of events, I is the number of users, and G is the number of groups. In case of interval-censored data, we proposed a Monte-Carlo EM inference algorithm where we iteratively impute the time-stamped events using a Poisson process that has intensity function approximates the underlying intensity function. We show that that proposed simulated approach delivers better inference performance than baseline methods.

In the advertisement problem, we propose a Click-to-Conversion delay model that uses Hawkes processes to model the advertisement impressions and thinned Poisson processes to model the Click-to-Conversion mechanism. We then derive an efficient Maximum Likelihood Estimator which utilizes the Minorization-Maximization framework. We verify the model against real life online advertisement logs in comparison with recent conversion rate estimation methods.

To facilitate reproducible research, we also developed an open-source software package that focuses on various Hawkes processes proposed in the above mentioned works and prior works. We provided efficient parallel (multi-core) implementations of the inference algorithms using the Bayesian variational inference framework. To further speed up these inference algorithms, we also explored distributed optimization techniques for convex optimization under the distributed data situation. We formulate this problem as a consensus-constrained optimization problem and solve it with the alternating direction method for multipliers (ADMM). It turns out that using bipartite graph as communication topology exhibits the fastest convergence.

# CHAPTER I

# INTRODUCTION

## *1.1 Point Processes*

Modern theory of point processes originated from important strands of studies [18], for example, survival analysis, theory of self-renewal processes; particle physics; communication engineering. For example, survival analysis and the theory of self-renewal processes are the studies of intervals between events. In modeling the *time to failure* of a component, denoted by a random variable $T$, survival analysis's primary interest is the *survival function*

$$S(t) = \mathbb{P}\{T > t\}, \tag{1.1.1}$$

the probability that the component survives pass time $t$. Equivalent entities of the survival functions are

$$F(t) = 1 - S(t) \text{ (lifetime distribution function)}, \tag{1.1.2}$$

$$f(t) = \frac{d}{dt}F(t) = -\frac{d}{dt}S(t) \text{ (lifetime density function)}, \tag{1.1.3}$$

$$\lambda(t) = \frac{f(t)}{S(t)} = \frac{d}{dt}\left[-\ln S(t)\right] \text{ (hazard function)}. \tag{1.1.4}$$

Different models in survival studies differ in the definitions of these functions. Choices for the density function include the exponential, Weibull, gamma, and log-normal functions. Among them, the exponential density function is the most popular choice.

Informally, a *point process* is a random collection of points in some space. These points could be the specific time and/or locations of the events that one is studying. In this thesis, we primarily concern with *temporal point process* where the points are the positions on the real half-line $\mathbb{R}_+$. In this case, a point process is just a random

countable subset of $\mathbb{R}_+$, i.e. a realization of the point process is an ordered set of positive real numbers

$$0 < t_1 < t_2 < \ldots < t_n.$$

Let us start with the equivalent definitions of temporal point processes.

**Definition 1.1.1** (Probability space definition)**.** A temporal point process is a measurable map

$$\Pi : \Omega \mapsto \mathfrak{N}$$

from the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ to the set of finite counting measures $\mathfrak{N}$ on $\mathbb{R}_+$.

In other words, the number of points in $\Pi$ falling in any test set $A \subset \mathbb{R}_+$ is a *random positive integer*. Because of the properties of open intervals, one could have an equivalent defintion based on the count of points falling in the intervals $[0, t)$.

**Definition 1.1.2** (Counting process)**.** A counting process is a stochastic process $\{N(t)|t \geq 0\}$ such that

- $N(t) \geq 0, \forall t \in \mathbb{R}_+$ (positive).
- $N(t) \in \mathbb{Z}_+$ (integer).
- $s \leq t \Rightarrow N(s) \leq N(t)$ (increasing).

This definition allows one to model the random number $N(t)$ directly. As we could see in later sections, in the case of Poisson point processes, $N(t)$ follows Poisson distribution. In the case of *simple processes* (i.e. with probability one, all points are distinct), one could equivalently define the process using the *conditional intensity function*

$$\lambda(t|\mathcal{H}_t) = \lim_{\Delta t \to 0^+} \frac{\mathbb{E}[N(t + \Delta t) - N(t)|\mathcal{H}_t]}{\Delta t}, \tag{1.1.5}$$

where $\mathcal{H}_t$ is the history of the point process just before time $t$. One could view the intensity $\lambda(t)$ as the rate that new event occurs in a infinitesimal interval just after time $t$. There is a connection between Eq. (1.1.5) and the hazard function in Eq. (1.1.4). As one could see, the hazard function is just the intensity of new event given that the object has survived until time $t$ (i.e. given history). Therefore, popular choices for hazard function in survival analysis are also popular choices for intensity function in temporal point process.

## *1.2 Applications*

Applications of point processes are both theoretical and practical. Below, we list some of the most well known practical applications of this theory:

- Expected number of events in a specific intervals: The most obvious application of point processes is to compute the quantity

$$\mathbb{E}\left[N(b) - N(a)\right] = \int_a^b \lambda(t)dt. \tag{1.2.1}$$

  The earliest example is Erlang's pioneering work on the Poisson distribution of number of calls [30]. Recent applications involves manipulating $\lambda(t)$ such that a desired expected number of events will occur in the future [32].

- Population size and population growth rate: In a community (e.g. a country or a city), the population is simultaneously dying and reproducing. From these changes arise the differential and integral equations concerning population growth. One could then related the survival function of each individual with the population size of the community. This then leads to the problem of manipulating key characteristics of the population so that the community could achieve the desired population growth rate or population size.

- Life time distributions of system of elements: In a sophisticated system that

consists of many components, the success or failure of that system when performing some task in a observing interval $[0, T]$ depends greatly on the structure of that system; whether the components are connected series or in parallel or some hybrid structure. From the point process viewpoint, one could model a system of components as a set of related point processes where each process represents the failure events of one component. One such process is the Hawkes process that will be discussed in more details in subsequent sections. Application of this viewpoint could be future failure prediction, optimized maintenance, etc.

## *1.3 Inference problems*

The most basic inference problem for temporal point processes is the estimation of the intensity function $\lambda(t)$. The difficulty of the estimation problem depends on various theoretical and practical situations. Let us discuss a few situations that are related to this thesis

- *Deterministic* or *Random*: The Poisson process has a deterministic intensity function (i.e. $\lambda(t)$ depends only on $t$), other processes may have random intensity function (e.g. Cox processes, Hawkes processes).

- *Memoryless* or *History dependent*: The Poisson process also has the memoryless property where the intensity $\lambda(t)$ is independent of events before $t$. On the other hand, the Hawkes processes has an history-dependent intensity function.

- *Non-parametric* or *Parametric*: The estimation could be non-parametric [22] or via a parametric model of $\lambda(t)$.

- *Full data* or *Histogram data*: In certain situations, one may have the full data (i.e. the time of the occurred events) while on other situations, the data may

be *interval-censored* and only counts of events in each pre-defined intervals are available.

- *One dimensional* or *Multi-dimensional*: The events could be the same type (1-dimensional) or consist of different types of related events (multi-dimensional).

- *Unmarked* or *Marked*: Sometimes, the data are annotated with additional information of the events. For example, a comment consists of the comment timestamps and the comment text.

## *1.4 Motivation*

This thesis focuses on the inference problem of multi-dimensional point processes. Multi-dimensional point processes recently have many new applications with respect to the analysis of network generated activity. The crux of multi-dimensional point processes is that they could model the interaction among different individual point processes. For example, friends or foes activities often trigger a burst of subsequent activities either to support or to counteract the starting activity. Therefore, it is intuitive to estimate or infer the "closeness" of individuals in a network by observing their activities pattern.

If, however, the network that generates the recorded events/activities forms certain community or clustering pattern, then the task of parameter estimation would be made more accurate if one could leverage this *prior information* about the network. In fact, a general framework for these "low-rank" or "small hidden dimension" prior information would be very useful. The advantage is two folds

- The prior information allows one to reduce the number of model parameters. This is desired because given limited training data, the more the number of parameters is, the less confidence the estimation becomes.

- The models become more tailored to the problem at hand by conforming to the prior knowledge of the network.

This is a great advantage compared to general approaches to enforce prior information such as regularization techniques. On one hand, while regularization techniques could generate sparse solutions, in the optimization phase, they still have to work with a significant amount of model parameters. On the other hand, all regularization terms are only surrogate functions of the desired quantities. For example, the $\ell_1$ norm is a substitution for the number of non-zero elements, a combinatorial quantity. The nuclear norm is a substitution for the rank of a matrix, which is also a combinatorial quantity. Therefore, minimizing these regularization terms does not necessarily minimize the desired quantities. Therefore, in the author's opinion, to create efficient inference algorithm for network activities in the case of available prior knowledge, it is better to incorporate the prior information into the models themselves.

In another interesting situation where only interval-censored data (i.e. histogram data) are available, many algorithms for both parametric and non-parametric models have been proposed for the Poisson processes [71]. However, the application of Poisson processes is limited in the case of social networks where an event is often a consequence of previous events (history dependent). The Hawkes processes is a useful models that recently gains interest in network generated activity analysis. However, to the author best knowledge, no parameter estimation algorithm for Hawkes processes has been proposed in the case of interval-censored data. Therefore, it would be a useful contribution to the research community to propose an algorithm that could work with interval-censored data.

## *1.5 Thesis statement*

This thesis claims that in the case that the activities generating network forms communities or in the case that only interval censored activity data are available, there

are efficient inference algorithms that estimate the intensity function via parametric models.

Evaluation of these claims is accomplished by creating novel inference algorithms that estimate the parameter of the Hawkes processes in the case that the network forming communities and in the case that only interval censored data are available (see Chapter 3 and Chapter 4). To further speed up these inference algorithms in distributed data settings, we create a novel distributed optimization algorithm that could leverage network communication for a concerted optimization effort (see Chapter 5).

## *1.6 Thesis overview*

In Chapter 2, I would like to review two interesting *point processes*, namely the Poisson processes and the Hawkes processes. I will discuss their definitions, their properties and the important problems of sampling and inference from these point processes.

In Chapter 3, I will discuss a novel Hawkes model that takes into account the community structure of the network. The proposed model reduces the number of parameters significantly and explicitly models the low-rank assumption of the influence pattern among individuals in a network. An efficient inference algorithm, NetCodec, is proposed based on the *mean-field variational inference* framework. We tested NetCodec against the vanilla Hawkes inference algorithm using synthetic and real-life datasets.

In Chapter 4, I will discuss the parameter estimation problem of the Hawkes model in the case that only *histogram data*, or *interval-censored* data, are available. The proposed inference algorithm works under Monte-Carlo EM framework and uses carefully designed sampling algorithm in order to sample the hidden variables. Experiments with synthetic and real-life network shows that the proposed algorithm using interval-censored data is comparable to the inference algorithm using fully observed

data.

In Chapter 5, I will discuss the usage of Hawkes process and thinned process to model the Click-to-Conversion mechanism in online advertisement campaigns. We propose that one could model the clicks as a Hawkes process and the conversions as a thinned process of the click process. Using the Minorization-Maximization framework, we derive an efficicient Maximum Likelihood Estimator for the proposed model. Experiments with real life advertisement log shows that our model could predict future conversion volume reasonably.

In chapter 6, I will discuss a distributed optimization algorithm that could further speed up learning algorithms in distributed data settings. We formulate the distributed learning problem as a concensus constrained optimization problem and solve it using the general methodology of Alternating Direction Method of Multipliers (ADMM). We then investigate the effects of the communication network topology on the convergence rate of the optimization.

In chapter 7, I will briefly introduce our open-source software package that implements the proposed models and inference algorithms described in this dissertation. We will show the design and the programming techniques that we used achieve efficient implementations that perform and scale well both with data size and computing power (i.e. parallelism).

In chapter 8, I will conclude the dissertation with some remarks on the current work and possible future directions.

# CHAPTER II

# TEMPORAL POINT PROCESSES

## *2.1 Poisson Processes*

### 2.1.1 Poisson distribution

**Definition 2.1.1.** An *non-negative integer* random variable $X$ has the Poisson distribution, $\mathcal{P}(\mu)$ if

$$\mathbb{P}\{X = n\} = \pi_n(\mu) = \frac{\mu^n e^{-\mu}}{n!}, n = 0, 1, 2, \ldots \tag{2.1.1}$$

The parameter $\mu > 0$ is the *mean* of the distribution $\mathcal{P}(\mu)$ and $\pi_n(\mu)$ is the *probability mass function* of $\mathcal{P}(\mu)$.

From the equality

$$\mathbb{E}(z^X) = e^{-\mu(1-z)}, \forall z : |z| \leq 1, \tag{2.1.2}$$

one could differentiate and set $z = 1$ to get

$$\begin{aligned} \mathbb{E}[X] &= \mu, \\ \mathbb{E}[X(X-1)] &= \mu^2, \\ \mathbb{E}[X(X-1)(X-2)] &= \mu^3, \ldots \end{aligned}$$

and compute the important statistics of $X$ such as

$$\mathbb{E}[X] = \mu \tag{2.1.3}$$

$$\mathbb{V}[X] = \mu. \tag{2.1.4}$$

The following *countable additivity theorem* shows the distribution of the sum of independent Poisson random variables.

**Theorem 2.1.2** (Countable Additivity)**.** *Let $X_j, j = 1, 2, \ldots$ be independent random variables, and assume that $X_j$ has the distribution $\mathcal{P}(\mu_j)$ for all $j$. If the series*

$$\sigma = \sum_{j=1}^{\infty} \mu_j \tag{2.1.5}$$

*converges, then*

$$S = \sum_{j=1}^{\infty} X_j \tag{2.1.6}$$

*converges with probability 1, and $S$ has distribution $\mathcal{P}(\sigma)$. If on the other hand* (2.1.5) *diverges, then $S$ diverges with probability 1.*

*Proof.* See [50], chapter 1. □

### 2.1.2 Probability spaces

The state space for a point process in a real line is the real line, $\mathbb{R}$, itself. A Poisson process on the real line, defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, is a function $\Pi$ from the set of "outcomes" $\Omega$ to the set $\mathbb{R}^\infty$ of all countable subsets of $\mathbb{R}$. That is, for every possible outcome $\omega \in \Omega$, $\Pi(\omega)$ is a countable subset of $\mathbb{R}$. To make the definition of the probability space concrete, one needs the construction of the sets of events $\mathcal{F}$ and the probability measure $\mathbb{P} : \mathcal{F} \to [0, 1]$.

Let us consider a *test set* $A \subseteq \mathbb{R}$, and let us denote

$$N(A) = \#\{\Pi(\omega) \cap A\} \tag{2.1.7}$$

the number of points in $\Pi(\omega)$ that are also in $A$. Therefore, for each test set $A$, the function $N(A)$ is a *non-negative integer* random variable if the following condition holds

$$\{\omega \in \Omega : N(A) = n\} \in \mathcal{F}, \forall n = 0, 1, 2, \ldots. \tag{2.1.8}$$

For Poisson process on the real line, it is sufficient that the condition (2.1.8) holds for all open intervals $A = (a, b)$. This is true because

- Any open set $G$ on $\mathbb{R}$ is a union of disjoint open intervals $A_j$, therefore

$$N(G) = \sum_j N(A_j)$$

  is a random variable.

- Any closed set $F$ on $\mathbb{R}$ is a limit of sequence of open sets $G_i$ such that $G_{i+1} \subset G_i, i = 1, 2, \ldots$, and

$$N(F) = \lim_{i \to \infty} N(G_i)$$

  is also a random variable.

Therefore, for any Borel set $A \subset \mathbb{R}$, one could define the random variable $N(A)$ if (2.1.8) holds for any open intervals on the real line. The construction of the "events" set $\mathcal{F}$ is the same for all point processes. The difference among point processes is the construction of the probability measure $\mathbb{P}$.

**Definition 2.1.3** (Probability space construction)**.** The probability space $(\Omega, \mathcal{F}, \mathbb{P})$ of a Poisson process on the real line satisfies the following conditions

- For all open intevals $A \subset \mathbb{R}$, the event $\{\omega \in \Omega : N(A) = n\}$ is an event in $\mathcal{F}$, for all $n = 0, 1, 2, \ldots$.

- For any disjoint measurable sets $A_1, A_2, \ldots, A_n \subset \mathbb{R}$, the random variables $N(A_1), N(A_2), \ldots, N(A_n)$ are independent, and

- $N(A)$ has Poisson distribution $\mathcal{P}(\mu(A))$, where $\mu(A)$ is called the *mean measure* of $A$.

In the interesting case that $\mu(A)$ is the integral of another function

$$\mu(A) = \int_A \lambda(t) dt, \tag{2.1.9}$$

we call $\lambda(t)$ the *rate* or *intensity* function of the Poisson process. This is because for continuous $\lambda(t)$ and small set $A$, the mean measure $\mu(A) \approx \lambda(t)|A|$ is the expected number of points in $\Pi$ that fall into $A$ (see Eq. (2.1.3)).

### 2.1.3 Properties of Poisson processes

In this section, some important properties of Posson processes are listed. The proofs of these properties could be found in [50].

**Theorem 2.1.4** (Superposition)**.** *Let $\Pi_1, \Pi_2, \ldots$ be a countable collection of independent Poisson processes on $\mathbb{R}$ and let $\Pi_n$ has mean measure $\mu_n$, for all $n$. Then their superposition*

$$\Pi = \bigcup_{n=1}^{\infty} \Pi_n$$

*is a Poisson process with mean measure*

$$\mu = \sum_{n=1}^{\infty} \mu_n. \tag{2.1.10}$$

**Theorem 2.1.5** (Restriction)**.** *Let $\Pi$ be a Poisson process on $\mathbb{R}$ with mean measure $\mu$, and $S \subset \mathbb{R}$ be a measurable set. Then the random countable set*

$$\Pi_S = \Pi \cap S$$

*is a Poisson process on $\mathbb{R}$ with mean measure*

$$\mu_S(A) = \mu(A \cap S), \forall \text{ measurable } A. \tag{2.1.11}$$

**Theorem 2.1.6** (Mapping)**.** *Let $\Pi$ be a Poisson process on $\mathbb{R}$ with mean measure $\mu$, and a function $f : \mathbb{R} \to \mathbb{R}$ such that the measure*

$$\mu^* = \mu^*(A) = \mu(f^{-1}(A)) \tag{2.1.12}$$

*is non-atomic (where $f^{-1}$ is the inverse mapping of $f$). Then $f(\Pi)$ is a Poisson process with mean measure $\mu^*$.*

**Collorary 2.1.7.** *Let $\lambda(t)$ be the intensity function of the Poisson process $\Pi$ and $M(t) = \mu([0,t)) = \int_0^t \lambda(\tau)d\tau$. Then $M(\Pi)$ is a Poisson process with constant unit intensity.*

*Proof.* Let $M(t) = x \in \mathbb{R}$, because $M(t)$ is monotone, we have

$$\mu^*([0, x)) = \mu^*([0, M(t))) = \mu([M^{-1}(0), M^{-1}(M(t)))) = \mu([0, t)) = M(t) = x.$$

Therefore, the intensity of the process $M(\Pi)$ is $\lambda^*(x) = \frac{d}{dx}\mu^*([0, x)) = 1$. □

**Theorem 2.1.8** (Existence)**.** *Let $\mu$ be a non-atomic measure on $\mathbb{R}$ which can be expressed in the form*

$$\mu = \sum_{n=1}^{\infty} \mu_n, \mu_n(\mathbb{R}) < \infty. \tag{2.1.13}$$

*Then there exists a Poisson process on $\mathbb{R}$ having $\mu$ as its mean measure.*

**Theorem 2.1.9** (Waiting time)**.** *Let the points in the point processes $\Pi$ are sorted ascendingly $x_1 < x_2 < \ldots < x_i < \ldots$ then $\Pi$ is a Poisson process with constant intensity function $\lambda(t) = \lambda$ if and only if the* waiting times *$d_i = x_{i+1} - x_i, i = 1, 2, \ldots$ are independently and identically distributed from the exponential distribution* $\text{Exp}(\lambda)$.

### 2.1.4 Sampling from Poisson process

In this section, we consider the problem of simulating a Poisson process on a closed interval $A = [0, T]$ where $T$ is the right limit of the observation window. By definition, if $\Pi$ is a Poisson process with mean measure $\mu$ then the number of points of $\Pi$ that fall into $A$, $N(A)$, has Poisson distribution $\mathcal{P}(\mu(A))$. In case that there is an intensity function $\lambda(t)$, we have

$$\mathbb{P}(N(A) = n) = \frac{\left(\int_A \lambda(t)dt\right)^n}{n!} \exp\left\{-\int_A \lambda(t)dt\right\}. \tag{2.1.14}$$

Now, suppose that we know $N(A) = n$, let us consider the distribution of $n$ points $\{x_1, x_2, \ldots, x_n\} = \Pi \cap A$. To that end, let $A_1, A_2, \ldots, A_k$ be an arbitrary partition of

**Algorithm 2.1** Simulating a Poisson process on $A = [0, T]$

1: Draw $n$ from Poisson distribution $\mathcal{P}\left(\int_A \lambda(t)dt\right)$.
2: **for** $i = 1, 2, \ldots, n$ **do**
3: Draw $x_i$ independently and identically from the density

$$p_X(t) = \frac{\lambda(t)}{\int_A \lambda(t)dt}$$

4: **end for**

$A$, i.e. $A_i \cap A_j = \emptyset, i \neq j$ and $\cup_{i=1}^k A_i = A$. By definition of Poisson process, we have

$$\begin{aligned}
&\mathbb{P}(N(A_i) = n_i, i = 1, 2, \ldots, k | N(A) = n) = \prod_{i=1}^k \mathbb{P}(N(A_i) = n_i) \left[P(N(A) = n)\right]^{-1} \\
&= \prod_{i=1}^k \frac{\left(\int_{A_i} \lambda(t)dt\right)^{n_i}}{n_i!} \exp\left\{-\int_{A_i} \lambda(t)dt\right\} \times \frac{n!}{\left(\int_A \lambda(t)dt\right)^n} \exp\left\{\int_A \lambda(t)dt\right\} \\
&= \frac{n!}{n_1! n_2! \ldots n_k!} \prod_{i=1}^k \left(\frac{\int_{A_i} \lambda(t)dt}{\int_A \lambda(t)dt}\right)^{n_i}. \qquad (2.1.15)
\end{aligned}$$

Now let us consider $n$ independent random points $Y = (y_1, y_2, \ldots, y_n)$ in $A$ that follow the density function

$$p_Y(t) = \frac{\lambda(t)}{\int_A \lambda(t)dt}. \qquad (2.1.16)$$

It easily follows that the number of points in $Y$ that fall into the sets $A_i, i = 1, 2, \ldots, k$ has multinomial distribution given by Eq. (2.1.15). Thus, we have shown that, given the number of points fall into $A$, the points $\Pi \cap A = \{x_1, x_2, \ldots, x_n\}$ are independent random points follow the density function in Eq. (2.1.16). Eq. (2.1.14) and Eq. (2.1.16) suggest that one could sample from a Poisson process on a observation time frame $[0, T]$ using Algorithm 2.1.

### 2.1.5 Likelihood function

*Full data.* In order to make inference about the Poisson process, one often needs to maximize the likelihood of the observed data [71]. For a Poisson process $\Pi$ on the real line and a test set $A$, a realization of $\Pi$ is a pair $\xi = (n, X)$ where $n = N(A)$ is

the number of points in $\Pi \cap A$, and $X = \{x_1, x_2, \ldots, x_n\}$ is the realized points. From the previous section, one could write the likelihood of the realized data as

$$\begin{aligned} p(\xi) &= \mathbb{P}(N(A) = n) p(\{x_1, x_2, \ldots, x_n\} \,|N(A) = n) \\ &= \frac{\left(\int_A \lambda(t)dt\right)^n}{n!} \exp\left\{-\int_A \lambda(t)dt\right\} n! \prod_{i=1}^n \frac{\lambda(x_i)}{\int_A \lambda(t)dt} \\ &= \exp\left\{-\int_A \lambda(t)dt\right\} \prod_{i=1}^n \lambda(x_i). \end{aligned} \tag{2.1.17}$$

In Eq. (2.1.17), we have used the fact that given the number of points $n$, the points $x_1, x_2, \ldots, x_n$ are independently and identically distributed with density $\frac{\lambda(t)}{\int_A \lambda(t)dt}$. The factor $n!$ arises from the fact that there are equally likely $n!$ ordered sets corresponding to the unordered set $\{x_1, x_2, \ldots, x_n\}$. In some cases, it is more convenient to work with the log-likelihood

$$\ln p(\xi) = \sum_{i=1}^n \ln \lambda(x_i) - \int_A \lambda(t)dt. \tag{2.1.18}$$

*Histogram data.* Another interesting inference problem is the problem of estimating $\lambda(t)$ given the observed count $n_1, n_2, \ldots, n_k$ of points falling into disjoint sets $A_1, A_2, \ldots, A_k$. To this end, one needs the likelihood function of these observed counts. Because of the definition of Poisson processes, we have

$$\begin{aligned} \mathbb{P}(N(A_i) = n_i, i = 1, 2, \ldots, k) &= \prod_{i=1}^k \mathbb{P}(N(A_i) = n_i) \\ &= \prod_{i=1}^k \exp\left\{-\int_{A_i} \lambda(t)dt\right\} \frac{\left(\int_{A_i} \lambda(t)dt\right)^{n_i}}{n_i!} \\ &= \exp\left\{-\int_A \lambda(t)dt\right\} \prod_{i=1}^k \frac{\left(\int_{A_i} \lambda(t)dt\right)^{n_i}}{n_i!}, \end{aligned} \tag{2.1.19}$$

where $A = \cup_{i=1}^k A_i$ is the union the interested sets $A_i$'s. The log-likelihood of the observed counts is

$$\ln \mathbb{P}(N(A_i) = n_i, \forall i) = \text{const} + \sum_{i=1}^k n_i \ln\left(\int_{A_i} \lambda(t)dt\right) - \int_A \lambda(t)dt. \tag{2.1.20}$$

Poisson processes provides a useful framework for deterministic point processes where the intensity function are fixed (i.e. depends only on $t$). In the next section, I will discuss the Hawkes processes where the intensity function is random and dependent on the history of events before time $t$.

## *2.2 Hawkes Processes*

### 2.2.1 One-dimensional Hawkes processes

It maybe most intuitive to define Hawkes processes using the intensity function. We first start with the definition of self-exciting processes.

**Definition 2.2.1** (Self-exciting process)**.** Let $N_s = N((-\infty, s))$ be the number of points that occur before point $s \in \mathbb{R}$. The intensity function $\lambda(t)$ of a self-exciting process is defined as

$$\lambda(t) = \lambda_0(t) + \int_{-\infty}^{t} \nu(t-s) dN_s = \lambda_0(t) + \sum_{i:t_i<t} \nu(t-t_i), \tag{2.2.1}$$

where $\lambda_0(t)$ is the deterministic *base intensity*, $\nu : \mathbb{R}_+ \mapsto \mathbb{R}_+$ describes the influence of past points $t_i < t$ on the current value of $\lambda(t)$. In case one would like to point out the influence of past points, one could write $\lambda(t) = \lambda(t|\mathcal{H}_t)$ where $\mathcal{H}_t$ is the history of events before time $t$.

Hawkes [39] proposes the influence function be a weight sum of of exponential function

$$\nu(t) = \sum_{k=1}^{K} \alpha_k e^{-\beta_k t} \mathbb{I}(t > 0). \tag{2.2.2}$$

In this work, we only consider the case where $\lambda_0(t) = \lambda_0$ is a constant function, and $K = 1$.

**Definition 2.2.2** (Hawkes processs)**.** The Hawkes proces is a self exciting process with intensity function $\lambda(t)$

$$\lambda(t) = \lambda_0 + \alpha \sum_{i:t_i<t} \omega e^{-\omega(t-t_i)}, \tag{2.2.3}$$

where $\lambda_0$ is the constant base intensity, $\alpha$ describes the *influence* of past points $t_i < t$ on the current value of $\lambda(t)$ and $\omega$ is the rate parameter of the influence function. The scale $\omega$ normalizes the influence function such that $\int_0^\infty \nu(t) = 1$.

Properties of Hawkes process could be found in many monographs. Below, we list the most important results that are relevant to this thesis. The proofs of these results could be found in [18].

**Theorem 2.2.3** (Likelihood function)**.** *Given an observing time frame* $[0, T)$*, the joint probability density of a realization* $\xi = (N_T = n, t_1 < t_2 < \ldots < t_n < T)$ *is*

$$p(\xi) = \prod_{i=1}^{n} \lambda(t_i|\mathcal{H}_{t_i}) \exp\left\{-\int_0^T \lambda(t|\mathcal{H}_t)dt\right\}, \text{ or} \tag{2.2.4}$$

$$\ln p(\xi) = \sum_{i=1}^{n} \ln \lambda(t_i|\mathcal{H}_{t_i}) - \int_0^T \lambda(t|\mathcal{H}_t)dt \tag{2.2.5}$$

For the intensity function in Eq. (2.2.3), one could write down the explicit form of the likelihood function

$$\ln p(\xi) = \sum_{i=1}^{n} \ln \left(\lambda_0 + \alpha \sum_{j:t_j<t_i} \omega e^{-\omega(t_i-t_j)}\right) - \lambda_0 T - \alpha \sum_{i=1}^{n} [1 - e^{-\omega(T-t_i)}]. \tag{2.2.6}$$

The log-likelihood function (2.2.6) is the starting point of most *maximum likelihood estimators* for $\lambda_0$, $\alpha$, and $\omega$.

The following result is a generalization of Collorary 2.1.7.

**Theorem 2.2.4** (Random time change)**.** *Let* $\Pi$ *is a Hawkes process and* $\Lambda(t) = \int_0^t \lambda(\tau|\mathcal{H}_\tau)d\tau$*. Then the process* $\Lambda(\Pi)$ *is a Poisson process with constant unit intensity.*

### 2.2.2 Sample verification

Theorem 2.2.4 is important in the sense that one could convert any Hawkes process to the standared Poisson process with constant unit intensity, i.e. $\lambda(t) = 1, \forall t$. Therefore, once could find many useful properties of the interested point process by converting it to standard Poisson process and study them. Another advantage of this

result is that it provide a way to check the quality of any sampling method for Hawkes processes. More detailed application of this theorem could be found in Chapter 4.

Because of this importance, in this section, we will list a proof of Theorem 2.2.4 [7]. Then we will show how to apply this theorem.

*Proof.* Recall that the original process is $\Pi = (t_1, t_2, \ldots, t_n) \in [0, T]$ and the mapped process $\Lambda = (\Lambda(t_1), \Lambda(t_2), \ldots, \Lambda(t_n)) \in [0, \Lambda(T)]$. Let $\tau_i = \Lambda(t_i) - \Lambda(t_{i-1})$ be the duration between consecutive mapped points and $\tau_T = \Lambda(T) - \Lambda(t_n)$. All we need is to show the $\tau$'s random variables are independently and identically distributed by exponential distribution with *mean one*, i.e. Exp(1).

The density of the mapped points is

$$p(\tau_1, \tau_2, \ldots, \tau_n \cap \tau_{n+1} > \tau_T) = p(\tau_1, \tau_2, \ldots, \tau_n) \times \mathbb{P}\left\{\tau_{n+1} > \tau_T | \tau_1, \tau_2, \ldots, \tau_n\right\}$$

The second term is

$$\begin{aligned}\mathbb{P}\left\{\tau_{n+1} > \tau_T | \tau_1, \tau_2, \ldots, \tau_n\right\} &= \mathbb{P}\left\{t_{n+1} > T | t_1, t_2, \ldots, t_n\right\} \\ &= \exp\left\{-\int_{t_n}^{T} \lambda(t|\mathcal{H}_{t_n})dt\right\} = \exp\left\{-\tau_T\right\}.\end{aligned}$$

For the first term, we have

$$p(\tau_1, \tau_2, \ldots, \tau_n) = |J| p(t_1, t_2, \ldots, t_n \cap N([0, t_n]) = n)$$

where $J$ is the Jacobian matrix of transformation between the variables $\tau$'s and $t$'s. We have

$$p(t_1, t_2, \ldots, t_n \cap N([0, t_n]) = n) = \prod_{i=1}^{n} \lambda(t_i|\mathcal{H}_{t_i}) \exp\left\{-\int_0^{t_n} \lambda(t|H_t)dt\right\}.$$

Now, let us consider the Jacobian $J$. As $\tau_i$ is the function of $t_i, t_{i-1}, \ldots, t_1$, $J$ is a lower triangular matrix. Therefore the determinant $|J|$ is the product of the diagonal elements. We have

$$J_{ii} = \frac{\partial t_i}{\partial \tau_i} = \left(\frac{\partial \tau_i}{\partial t_i}\right)^{-1} = \lambda(t_i|\mathcal{H}_{t_i})^{-1}.$$

Therefore, the density of the mapped points is

$$p(\tau_1, \tau_2, \ldots, \tau_n \cap \tau_{n+1} > \tau_T) = \exp\left\{-\int_0^{t_n} \lambda(t|H_t)dt\right\} \exp\{-\tau_T\}$$
$$= \prod_{i=1}^{n} \exp\{-\tau_i\} \times \exp\{-\tau_T\}$$

which completes the proof. □

As mentioned, Theorem 2.2.4 allows one to verify the quality of any sampling method. This is done by computing the mapped points $\Lambda(t_i)$'s where $t_i$'s are the locations of the simulated sample. Then one could compute the duration $\tau_i = \Lambda(t_i) - \Lambda(t_{i-1})$ for $i = 1, 2, \ldots, n$ and compare the empirical quantiles of $\tau_i$'s with the theoretical quantiles of the *exponential distribution* with mean one. For visualization purpose, one could use graphs such as the Q-Q plot to see the quality of the sampling method. For example, Figure 2.1 shows the quality of a sampling method for one-dimensional Hawkes process. More details on the applications of Theorem 2.2.4 are discussed in Chapter 4.

### 2.2.3 Multi-dimensional Hawkes processes

The previously mentioned point processes are *one-dimensional* point processes. Their use is limited to one kind/type of events. In the cases where multiple kinds of events are dependent on each other, one needs to leverage the concept of *multi-dimensional* point processes. In this section, I will discuss a generalization of the one-dimensional Hawkes processes to multiple dimensions.

**Definition 2.2.5.** The multi-dimensional Hawkes processes are the superposition of $U$ dependent point processes where $U$ is the dimension of the processes. In particular, the intensity function of the $i$-th dimension ($i = 1, 2, \ldots, U$) process is given by

$$\lambda_i(t) = \lambda_i(t|\mathcal{H}_t) = \lambda_{i0} + \sum_{n:t_n<t} \alpha_{ii_n}\kappa(t - t_n), \tag{2.2.7}$$

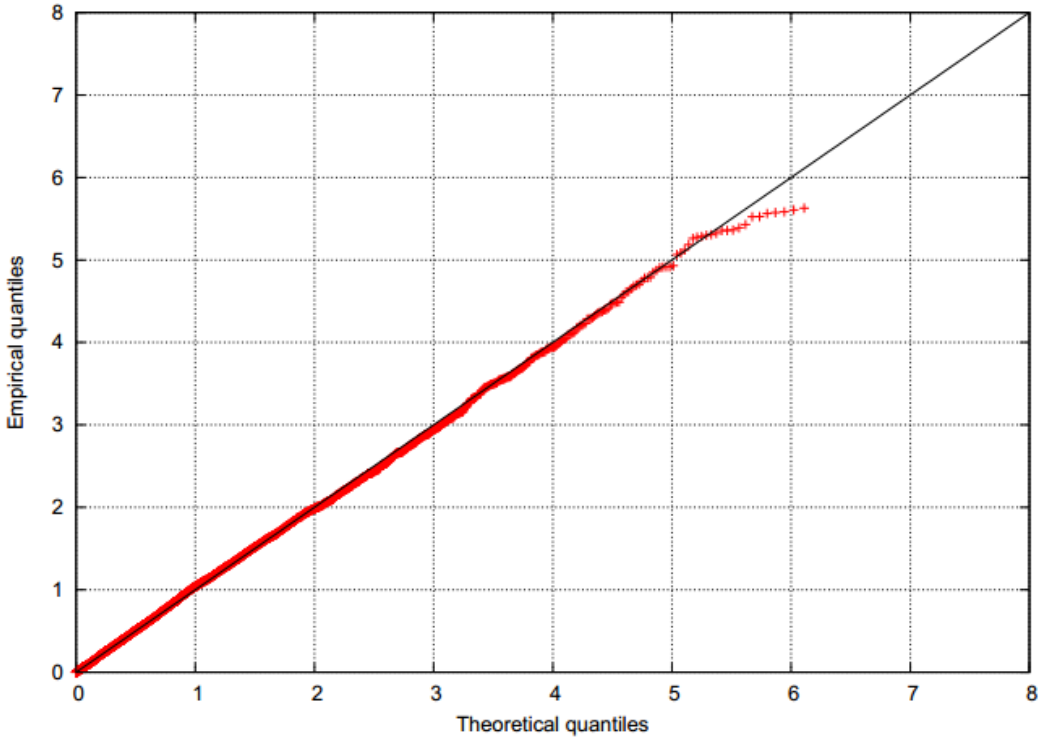



Figure 2.1: Quantile plot of a sample of the one-dimensional Hawkes processes.

where $\lambda_{i0} \geq 0$ is the deterministic constant base intensity for $i$-th dimension, $\alpha_{ij} \geq 0$ is the *influence coefficient* that events in $j$-th dimension have on future events in the $i$-th dimension, $t_n$ is an event that occurs before time $t$, and $i_n$ is the dimension of that event. The influence function $\kappa(t)$, also called the *triggering kernel*, is often chosen to be the exponential density function, $\kappa(t) = \omega e^{-\omega t}\mathbb{I}(t > 0)$.

If one collects all influence coefficients into a matrix $\mathbf{F} = [\alpha_{ij}]$, one could see that the structure of $\mathbf{F}$ represents the influence structure or influence pattern among events in different dimensions.

The following result specifies the likelihood function of a realization of the multi-dimensional Hawkes process.

**Theorem 2.2.6** (Likelihood function)**.** *Given an observing time frame* $[0, T)$*, the joint probability density of a realization* $\xi = (N_T = n_c, t_1 < t_2 < \ldots < t_{n_c} < T)$ *of a*

*U-dimensional Hawkes process is*

$$p(\xi) = \prod_{i=1}^{n_c} \lambda_{i_n}(t_n|\mathcal{H}_{t_n}) \prod_{i=1}^{U} \exp\left\{-\int_0^T \lambda_i(t|\mathcal{H}_t)dt\right\}, \text{ or} \tag{2.2.8}$$

$$\ln p(\xi) = \sum_{i=1}^{n} \ln \lambda_{i_n}(t_n|\mathcal{H}_{t_n}) - \sum_{i=1}^{U} \int_0^T \lambda_i(t|\mathcal{H}_t)dt \tag{2.2.9}$$

The following beautiful random time change theorem is very useful in proving other properties of the Hawkes processes. It provides a way to convert the Hawkes processes to the well-known and well-studied homogeneous Poisson processes.

**Theorem 2.2.7** (Random time change)**.** *Let* $\Pi$ *is a U-dimensional Hawkes process and* $\Lambda_i(t) = \int_0^t \lambda_i(\tau|\mathcal{H}_\tau)d\tau$*. Then the point processes* $\Lambda_i(\Pi_i), i = 1, \ldots, U$ *are U independent Poisson processes with constant unit intensity. Here* $\Pi_i$ *is the set of points in* $\Pi$ *that belong to the i-th dimension.*

### 2.2.4 Parameter Estimation

In this section, we consider the problem of estimating the Hawkes processes parameters: the base intensity $\lambda_{i0}, i = 1, \ldots, U$, the influence coefficients $\alpha_{ij}, i, j = 1, \ldots, U$, and the triggering kernel rate $\omega$ from observed data. To make the problem more practical, we consider the case where multiple realizations, called *cascades*, of the Hawkes processes are observed. In particular, our observed data consist of

- $C$ cascades, denoted by $\xi^c, c = 1, \ldots, C$.
- The $c$-th cascade is the observation in the time frame $[0, T_c)$

$$\xi^c = (n_c, (t_n^c, i_n^c), n = 1, \ldots, n_c) \tag{2.2.10}$$

  where $t_n^c$ is the $n$-th event, $i_n^c$ is the dimension of that event, $n_c$ is the number of events observed in the time frame $[0, T_c)$.

Our goal is to estimate the parameters $\Theta = (\lambda_{i0}, \alpha_{ij}, \omega)$ using the observed cascades $\{\xi^c\}, c = 1, \ldots, C$. To that end, we employ the *Maximum Likelihood Estimation*

(MLE) framework. In particular, we need to maximize

$$\max_{\Theta} \mathcal{L}(\Theta) \equiv \sum_{c=1}^{C} \ln p(\xi^c), \tag{2.2.11}$$

where $\ln p(\xi^c)$ is given by Eq. (2.2.9). To be more specific, let us first write down the explicit form of $\mathcal{L}(\Theta)$ by substituting the intensity in Eq. (2.2.7) to the log-likelihood in Eq. (2.2.9)

$$\begin{aligned}
\mathcal{L}(\Theta) &= \sum_{c=1}^{C} \left\{ \sum_{i=1}^{n_c} \ln \left( \lambda_{i_n^c 0} + \sum_{t_\ell^c < t_n^c} \alpha_{i_n^c i_\ell^c} \kappa(t_n^c - t_\ell^c) \right) - \sum_{i=1}^{U} \int_0^{T_c} \left( \lambda_{i0} + \sum_{t_n^c < t} \alpha_{i i_n^c} \kappa(t - t_n^c) \right) dt \right\} \\
&= \sum_{c=1}^{C} \left\{ \sum_{i=1}^{n_c} \ln \left( \lambda_{i_n^c 0} + \sum_{t_\ell^c < t_n^c} \alpha_{i_n^c i_\ell^c} \kappa(t_n^c - t_\ell^c) \right) - T_c \sum_{i=1}^{U} \lambda_{i0} - \sum_{u=1}^{U} \sum_{n=1}^{n_c} \alpha_{i i_n^c} K(T_c - t_n^c) \right\},
\end{aligned} \tag{2.2.12}$$

where $K(t) = \int_0^t \kappa(\tau) d\tau$ is the integral of the triggering kernel function.

In Eq. (2.2.12), the second and the third terms are linear functions with respect to the parameters $\lambda_{i0}, \alpha_{ij}$. The first term in is the logarithm of a linear function of $\lambda_{i0}, \alpha_{ij}$, therefore it is concave. In summary, the log-likelihood of the observed data is concave with respect to $\lambda_{i0}, \alpha_{ij}$. Therefore, if one fixes the triggering kernel rate $\omega$, the optimization problem (2.2.11) is a *convex optimization*. Therefore, one could apply various well-known and well-studied convex optimization techniques such as *Gradient Descent* or *L-BFGS* to solve for $\lambda_{i0}, \alpha_{ij}$.

In this thesis, we use a different optimization framework, called the *Minorization-Maximization* or MM. In this framework, we first derive a lower bound of the log-likelihood $\mathcal{L}(\Theta)$ and then maximize this lower bound. The advantages of MM framework are

- It works well in both convex case (fixed $\omega$) and non-convex case (variable $\omega$).
- It easily generalizes to more complicated models (see later chapters).
- It allows closed-form, fast, and parallel updates of the parameters.

**Algorithm 2.2** Minorization-Maximization framework

1: Initialize $\Theta_1$.
2: **for** $m = 1, 2, \ldots, n$ **do**
3: Maximize $Q(\Theta|\Theta_m)$ to get the next iterate

$$\Theta_{m+1} = \arg\max_{\Theta} Q(\Theta|\Theta_m)$$

4: **end for**

To be specific, let $\mathcal{L}(\Theta)$ be the function that we need to maximize. Assuming that we could find a *surrogate function* $Q(\Theta|\Theta_m)$ such that

$$Q(\Theta|\Theta_m) \leq \mathcal{L}(\Theta), \forall\Theta,$$

$$Q(\Theta_m|\Theta_m) = \mathcal{L}(\Theta_m).$$

Then the MM framework is summarized in Algorithm 2.2 where in each iteration one maximizes $Q(\Theta|\Theta_m)$ given the current guess $\Theta_m$. Convergence properties of the MM framework could be found in many monographs, for example [79]. One could easily see the monotonic convergence of the framework provided the objective function is bounded above.

**Theorem 2.2.8** (Convergence of MM)**.** *Assuming that $\mathcal{L}(\Theta)$ is bounded above, then the sequence $\mathcal{L}(\Theta_m), m = 1, 2, \ldots$ found by Algorithm 2.2 is a monotonically increasing convergent sequence.*

**Theorem 2.2.9** (Stationarity of MM)**.** *If additionally $Q(\Theta|\Gamma)$ is continuous in both $\Theta$ and $\Gamma$ then the sequence $\mathcal{L}(\Theta_m), m = 1, 2, \ldots$ converges to a stationary point of $\mathcal{L}(\Theta)$.*

In the case of convex optimization (i.e. fixed $\omega$), a stationary point is also the global maximum of $\mathcal{L}(\Theta)$. To continue with our optimization problem of the Hawkes process log-likelihood function (2.2.12), let us derive a lower bound for $\mathcal{L}(\Theta)$.

**Theorem 2.2.10** (Lower bound)**.** *The log-likelihood* (2.2.12) *is lower bounded by*

$$\sum_{c=1}^{C}\left\{\sum_{i=1}^{n_c}\left[\eta_{nn}^{c}\ln\frac{\lambda_{i_n^c 0}}{\eta_{nn}^{c}}+\sum_{t_\ell^c<t_n^c}\eta_{\ell n}^{c}\ln\frac{\alpha_{i_n^c i_\ell^c}\kappa(t_n^c-t_\ell^c)}{\eta_{\ell n}^{c}}\right]-T_c\sum_{i=1}^{U}\lambda_{i0}-\sum_{i=1}^{U}\sum_{n=1}^{n_c}\alpha_{ii_n^c}K(T_c-t_n^c)\right\} \tag{2.2.13}$$

*where the auxiliary variables satisfy* $\eta_{\ell n}^{c}\geq 0, 1\leq\ell\leq n\leq n_c$ *and* $\sum_{\ell=1}^{n}\eta_{\ell n}^{c}=1$*. The equality holds when*

$$\begin{aligned}\eta_{nn}^{c}&\propto\lambda_{i_n^c 0},\\ \eta_{\ell n}^{c}&\propto\alpha_{i_n^c i_\ell^c}\kappa(t_n^c-t_\ell^c),\end{aligned} \tag{2.2.14}$$

*where "*$\propto$*" means "proportional to". Note that, one needs to normalize these auxiliary variables such that their sums is equal to 1.*

*Proof.* By concavity of logarithm (or the log-sum inequality), one has

$$\begin{aligned}\ln\left(\lambda_{i_n^c 0}+\sum_{t_\ell^c<t_n^c}\alpha_{i_n^c i_\ell^c}\kappa(t_n^c-t_\ell^c)\right)&=\ln\left(\eta_{nn}^{c}\frac{\lambda_{i_n^c 0}}{\eta_{nn}^{c}}+\sum_{t_\ell^c<t_n^c}\eta_{\ell n}^{c}\ln\frac{\alpha_{i_n^c i_\ell^c}\kappa(t_n^c-t_\ell^c)}{\eta_{\ell n}^{c}}\right)\\&\leq\eta_{nn}^{c}\ln\frac{\lambda_{i_n^c 0}}{\eta_{nn}^{c}}+\sum_{t_\ell^c<t_n^c}\eta_{\ell n}^{c}\ln\frac{\alpha_{i_n^c i_\ell^c}\kappa(t_n^c-t_\ell^c)}{\eta_{\ell n}^{c}}.\end{aligned}$$

Equality holds if and only if

$$\frac{\lambda_{i_n^c 0}}{\eta_{nn}^{c}}=\frac{\alpha_{i_n^c i_\ell^c}\kappa(t_n^c-t_\ell^c)}{\eta_{\ell n}^{c}},\forall\ell<n$$

which completes the proof. □

Theorm 2.2.10 shows that one could apply the MM framework to maximize log-likelihood $\mathcal{L}(\Theta)$ in (2.2.12). To proceed, we need the following useful lemma.

**Lemma 2.2.11.** *Given* $a,b>0$*, the maximum of the function* $f(x)=a\ln x-bx$ *is attained at*

$$x^*=\frac{a}{b}. \tag{2.2.15}$$

*Proof.* As $f(x)$ is concave, setting the derivative of $f(x)$ to zero, we get (2.2.15). □

Using lemma 2.2.11, one could maximize the lower-bound (2.2.13) by alternating between the model parameters $\{\lambda_{i0}, \alpha_{ij}\}$ and the auxiliary variables $\{\eta^c_{\ell n}\}$. Given a set of auxiliary variables, applying lemma 2.2.11 on the lower bound (2.2.13), the update formulas for $\{\lambda_{i0}, \alpha_{ij}\}$ are

$$\begin{aligned} \lambda_{i0} &\leftarrow \frac{\sum_{c=1}^{C} \sum_{n=1}^{n_c} \eta^c_{nn} \mathbb{I}(i^c_n = i)}{\sum_{c=1}^{C} T_c}, \\ \alpha_{ij} &\leftarrow \frac{\sum_{c=1}^{C} \sum_{\ell<n\leq n_c} \eta^c_{\ell n} \mathbb{I}(i^c_n = i, i^c_\ell = j)}{\sum_{c=1}^{C} \sum_{n=1}^{n_c} K(T_c - t^c_n) \mathbb{I}(i^c_n = j)}, \end{aligned} \tag{2.2.16}$$

where $\mathbb{I}(\cdot)$ is the indicator function

$$\mathbb{I}(p) = \begin{cases} 1, & p \text{ is true,} \\ 0, & p \text{ is false.} \end{cases}$$

Once the model parameters are updated with formulas (2.2.16), the lower-bound (2.2.13) could be further tighten via updating the auxiliary variables with formulas (2.2.14).

Now, let us consider the problem of estimating the triggering kernel rate $\omega$ in the case that the triggering kernel function $\kappa(t) = \omega e^{-\omega t}$, the exponential density function. Its integral $K(t) = 1 - e^{-\omega t}$ is approximately 1 when $\omega \ll t$. This is true because the duration $T_c - t^c_n$ is often very large as one often chooses a large observation time frame $T_c$. As a result, with respect to optimizing $\omega$, one could consider the last term in the expression (2.2.13) a constant. Therefore, applying lemma 2.2.11, one has

$$\omega \leftarrow \frac{\sum_{c=1}^{C} \sum_{\ell<n} \eta^c_{\ell n}}{\sum_{c=1}^{C} \sum_{\ell<n} (t^c_n - t^c_\ell) \eta^c_{\ell n}}. \tag{2.2.17}$$

Algorithm 2.3 summarizes the estimation of Hawkes processes parameters under the MM framework.

**Algorithm 2.3** Estimate Hawkes process parameters under MM framework

1: Initialize $\Theta_1$.
2: **for** $m = 1, 2, \ldots, n$ **do**
3: Compute auxiliary variables $\eta_{\ell n}^c$ with (2.2.14)
4: Update $\Theta_{m+1}$ with (2.2.16) and (2.2.17).
5: **end for**

### 2.2.5 Limitations

Algorithm 2.3 have many good properties, especially when one needs to implement and deploy it in high performance computing environment. Let us list a few implementation issues for algorithm 2.3.

- *Closed form, fast update*: All update formulas (2.2.14), (2.2.16), and (2.2.17) are in closed form using simple arithmetics.

- *Parallel update*: The computation of $\eta_{\ell n}^c$ for different $n$'s are completely independent of each other. As the number of events $n_c$ is often very large, this is a great source of parallelism. Furthermore, the accumulation of the numerators and denominators in Eqs. (2.2.16) and (2.2.17) could be done in parallel and on the fly. That is, once $\eta_{\ell n}^c$ is computed, one could accumulate it into the mentioned numerators and denominators and discard it.

- *Truncated update*: As the triggering kernel $\kappa(t)$ is a fast decaying function, it is reasonable to only consider events that are close to each other. In our implementation, we only compute $\eta_{\ell n}^c$ for $\ell, n$ such that $\kappa(t_n^c - t_n^c) > 10^{-16}$. This reduction is very significant when the observation time frame $T_c$ is large.

- *Stopping criteria*: The current objective value of the log-likelihood (2.2.12) could be computed very efficiently via dynamic programming with complexity $O\left(U \sum_{c=1}^{C} n_c\right)$. One possible stopping criteria is the relative change of $\mathcal{L}(\Theta_{m+1})$

with respect to $\mathcal{L}(\Theta_m)$. We use the following stopping criteria

$$\left|\frac{\mathcal{L}(\Theta_{m+1}) - \mathcal{L}(\Theta_m)}{\mathcal{L}(\Theta_m)}\right| < 10^{-4}. \tag{2.2.18}$$

We observe that the algorithm often stops within 40 iterations for the data sets used in this thesis.

However, algorithm 2.3 and the Hawkes processes defined in Definition 2.2.5 have certain intrinsic weaknesses and limitations.

- The number of parameters one needs to estimate is $O(U^2)$ where $U$ is the number of dimensions. As the data size (i.e. the number of events) is not infinite, this lowers the confidence and increases the variance of the estimated parameters.

- For many problems, especially events from social networks, it is often true that only a few pairs of dimensions have influence on each other. In other word, the influence coefficient matrix $\mathbf{F}$ is often very sparse. While one could fix the formulation (2.2.11) with additional $\ell_1$ regularization, in the optimization, one still needs to work with $O(U^2)$ parameters.

- In another situation that the dimensions (e.g. users in social networks) form communities, the influence coefficient matrix $\mathbf{F}$ would be low rank as users in the same community often have similar affection/influence to users in another community. Formulation (2.2.11) provides no easy way to cope with this situation.

- The log-likelihood (2.2.12) is defined on the full data $\xi^c = (n_c, t_1, t_2, \ldots, t_{n_c})$, $c = 1, 2, \ldots, C$. It is not clear how one could estimate the parameters when only histogram data are available. The data in this situation is *interval-censored*. That is, we do not know the specific time of event $t_n^c$. Instead, we only know the number of events falling into pre-defined disjoint intervals in the observation time frame $[0, T_c)$.

In the next chapters, I will discuss novel models and novel inference algorithms that address these limitations.

# CHAPTER III

# COMMUNITY INFERENCE

## *3.1 Introduction*

The exponential growth of recorded social activities has inspired many interesting research directions. From individual activities, a curious analyzer would like to infer more about the social networks as a whole. For example, how contagious individuals' activities are on each other? Are people forming coherent groups or communities in their activities? What is a person's role in his/her perceived community? Is it possible to process the massively available data to answer these crucial questions? These are naturally very interesting and important research questions. The answers to these questions are already having significant impact in practice. For example, in viral marketing, one would like to maximize influence of product advertisement with the least cost. To that end, it is highly beneficial to correctly detect social communities and pinpoint popular individuals whose popularity assures maximized product adoption [32].

Both network infectivity inference and community detection from activities have been addressed extensively. While they are usually studied separately [81, 49, 5], event cascades and clusters are natural duals: clusters block the spread of influence, i.e., whenever a cascade of events comes to a boundry, there is a cluster that can be used to explain why [27]. On the other hand, if a cluster can justify a cascade comes to a stop, then past chain of events can find out something about the clusters.

Based on this fact, we propose a modeling approach that takes into account both network infectivity and community structure in modeling individual activities. Our modeling approach leverages a key observation that these characteristics of a social

network intertwine and knowing one would help better understanding and revealing the other. As a result, it is possible to simultaneously infer network infectivity and to detect community structure from individual activities. The proposed method also benefits from having fewer model parameters than existing approaches in literature. This is highly useful as one usually only has limited event data and having fewer model parameters often implies less variance and less algorithmic complexity.

In particular, we propose NetCodec (NETwork COmmunity DEteCtion), a scalable variational inference algorithm for simultaneous network infectivity inference and community detection from individual activities or events. The key idea of the algorithm is to factorize network infectivity into community participation and individual popularity and to leverage the *mean field variation inference framework* to estimate the community participations. Our algorithm can estimate the network infectivity and community structure of a network with $I$ nodes, $G$ groups with $O(kNG + IG)$ computations per iteration, where $N$ is the number of recorded events in a certain time frame, and $k$ is the average number of relevant historical events ($k \ll N$). We validate NetCodec in various simulated and real-world situations.

### 3.1.1 Problem settings

We assume that there are $I$ identities (e.g. individuals, users, sources) that could be grouped into $G$ groups and that their activities are contagious following some network infectivity rate. The community structure and network infectivity are *unknown* to us. Instead, we only know the time and the identity of events (e.g. posts, comments, purchases, earthquakes) occurred in a time frame. The natural question is that "Could we recover both community structure and network infectivity simultaneously from their activities?".

Specifically, let the time and identity of events form a set of $C$ cascades

$$\left\{(t_n^c, i_n^c)_{n=1\ldots N_c}\right\}_{c=1\ldots C},$$

Table 3.1: Notations

| | |
|---|---|
| $I$ | number of individuals/nodes |
| $G$ | number of groups |
| $N$ | number of events |
| $\lambda_i(t)$ | intensity at time $t$ of user $i$ |
| $\mu_i$ | spontaneous rate of user $i$ |
| $\beta_i$ | the celebrity index of user $i$ |
| $\mathbf{Z}_i \in \mathbb{R}_+^G$ | group participation vector of user $i$ |
| $\alpha_{ij}$ | infectivity rate from user $j$ to user $i$ |
| $\kappa(t)$ | triggering kernel |
| $K(t)$ | the integral $\int_0^t \kappa(\tau)d\tau$ |
| $a_g, b_g$ | Gamma distribution parameters |
| $\ell, n$ | event indices, $\ell < n$ if both present |

where $t$'s are the time of events and $i$'s are the identities. The observation time frame for the $c$-th cascade is $[0, T_c]$. We would like to find a *participation matrix* $\mathbf{Z} = [z_{ig}] \in \mathbb{R}_+^{I \times G}$ where $z_{ig}$ represents how strong the $i$-th node associates to the $g$-th group. We also want to find an *infectivity matrix* $\mathbf{F} = [\alpha_{ij}] \in \mathbb{R}_+^{I \times I}$ where $\alpha_{ij}$ represents how the $j$-th node influences the $i$-th node. In the following, the terms "identity", "user", "node" have the same meaning.

In Section 3.2, we discuss our approach and the modelling technique in more details. In Section 3.3 we derive NetCodec, a variational inference algorithm that efficiently infers network infectivity and detects coherent communities. In Section 3.4, we report the experiment results where we apply the model on various simulated and real world situations. In Section 3.5, we conclude the paper with some remarks on the proposed method and future directions. Before proceeding, let us discuss the related literature on the proposed problem.

### 3.1.2 Related Works.

Recently, there has been a growing interest in network inference from event data. Authors in [38] were one of the first who tackle the problem of inferring network from the event data. Given the times when nodes adopt pieces of information or become

infected, they approximate the optimal network that best explains the observed infection times. Perry et al. [68] introduced a model for treating directed interactions as a multivariate Cox intensity model with covariates that depend on the history of the process and learned the parameters using partial likelihood. Authors in [57] proposed a probabilistic model that combines mutually- exciting point processes with random graph models to infer latent networks. These models, while not being closely related, try to answer how nodes in the network are generally connected or how they influence each other. In contrast our model, directly involves community structure in the modeling.

More closely, authors in [2] proposed a generative model, Community-Cascade Network, based on mixture membership that can fit, at the same time, the social graph and the observed set of cascades. This model, nicely elaborates on the community detection and network inference, however, the nature of events data observed is too far from real applications. They require the data has been observed along with the chain of influence, i.e., which event causes this event. Furthermore, [56] aims at a similar problem, however, as the previous work the definition of event is far from the real data in hand. The event, contains some nodes participating in an event (eg. a party) along with the edges (friendships) between them. In their promising work, Zhou et al. [87], considered the community structure of the network in point process data via adding a regularization term based on nuclear norm. The community structure is only captured indirectly via regularization to enhance parameters estimation and thus cannot find the underlying modules in the network.

After Hawkes [40] originally proposed this mutually-exciting process it has been proved to be useful in various areas such as finance [29], seismology [66, 59], crime [70], civilian deaths in conflicts [54], and recently causal militant conflict events [55]. For social and influence networks, there are also recent uses of variants of Hawkes processes for analyzing Youtube movies [16], news websites [48, 87], and book sales

[21].

## *3.2 Modeling Network Activities*

In this section, we will discuss our approach to the problem set out in Section 6.1. We will first discuss our modeling technique where one could leverage community structure to help better revealing network infectivity. Then, we will described the technical aspect of the proposed model such as the likelihood function and the maximum likelihood estimator. The readers could refer to Table 3.1 for the notations used in this chapter.

### 3.2.1 The proposed model

We would like to model the activities in the network by the multi-dimensional Hawkes processes. Recall from Eq. (2.2.7), we have

$$\lambda_i(t) = \mu_i + \sum\nolimits_{t_\ell < t} \alpha_{ii_\ell} \kappa(t - t_\ell), \tag{3.2.1}$$

where $\mu_i > 0$ is the spontaneous intensity for the $i$-th dimension and $i_\ell$ is the dimension identity of the $\ell$-th event. The *nonnegative coefficient* $\alpha_{ij}$ captures the mutually-exciting property between the $i$-th and the $j$-th dimensions. It shows how much influence the events in $j$-th dimension has on future events in $i$-th dimension. Larger values of $\alpha_{ij}$ indicates that events in $j$-th dimension are more likely to trigger an event in the $i$-th dimension in the future.

From the modeling perspective, we would like to incorporate as many key characteristics of network infectivity as possible. Regarding *within-community infectivity*, naturally, individuals affiliated with same communities would have more influence on each other than individuals affiliated with different communities. This natural and key observation inspires us to make an assumption that network infectivity among users' activities depends on how strongly each individual *participates* in his/her community activities. The network infectivity matrix is also *asymmetric* in that a node

could have strong influence on another node but not vice versa. These popular nodes' activities tend to trigger a wider wave of events.

Regarding *cross-community infectivity*, individuals in a community often share some common understandings about individuals in other communities. For example, people in a country X have some stereotype about people in country Y. Therefore, a post by a person in country Y or about country Y will trigger certain common responses from people in country X. This situation happens regularly in chat rooms, blogs, and comment sections in the World Wide Web. The marginalization effect of the latent group identity therefore implies a *low-rank structure* of network infectivity. We also would like to incorporate this crucial observation in our modeling approach.

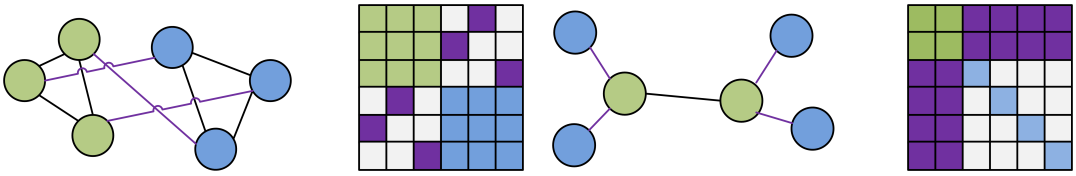
To proceed, let $\mathbf{Z}_i = (z_{i1}, \ldots, z_{iG}) > 0$ be user $i$'s degree of participation to the $G$ groups. Furthermore, let $\beta_i > 0$ represents how popular user $i$ is on the network, a *celebrity index*. We propose the following factorization of the infectivity of user $j$ to another user $i$'s activities

$$\alpha_{ij} = \beta_j \langle \mathbf{Z}_i, \mathbf{Z}_j \rangle = \beta_j \sum\nolimits_{g=1}^{G} z_{ig} z_{jg}, i \neq j.$$

As one could see, the more user $i$ and user $j$ participate in the same communities, the stronger the infectivity is. Besides, the popularity of user $j$ also boosts his/her influence on user $i$'s activities. The decomposition also shows the asymmetry as well as the low-rank implication of network infectivity. Note that, we only enforce the low-rank structure on the *off-diagonal elements* of network infectivity. This is a crucial difference in comparison to methods in matrix factorization literature.

Regarding the self-exciting property, we propose that one should not decompose the self-exciting rate $\alpha_{ii}$ and that one should consider it as a model parameter to infer from observed data. The reason is that self-exciting characteristic is an intrinsic property of each individual that is unrelated to his/her relation with other individuals. To keep the notation clear, we denote $\alpha_i = \alpha_{ii}, i = 1 \ldots I$.

To summarize, the previous reasoning leads to the following decomposition of the



(a) Cross-group infectivity (b) Core group and peripheral nodes

Figure 3.1: Different network scenarios and the corresponding infectivity matrices.

intensity function

$$\lambda_i(t) = \mu_i + \sum_{t_\ell<t}^{i_\ell \neq i} \sum_{g=1}^{G} \beta_{i_\ell} z_{ig} z_{i_\ell g} \kappa(t - t_\ell) + \alpha_i \sum_{t_\ell<t}^{i_\ell = i} \kappa(t - t_\ell). \tag{3.2.2}$$

Before we proceed, let us discuss some properties and advantages of this modeling approach. *First*, the most obvious advantage is that the number of parameters to infer from observed data is $O(I \times G)$ instead of $O(I^2)$ in the case of the original Hawkes process. This reduction is very beneficial given the fact that one often does not have infinite data. The reduction in number of parameters tends to make inference less variant. Besides, fewer number of parameters implies less complexity per iteration of the inference algorithm. *Second*, the decomposition of network infectivity $\alpha_{ij}$ still has more space for extensions. For example, in social networks, one could defines another decomposition that takes into account other activity's feature such as the post content and/or ratings. The interested reader could find some extensions to our model in the *supplemental material*. Another interesting observation is that one could factorize $\mathbf{F}$ into

$$\mathbf{F} - \text{diag}(\mathbf{F}) = \mathbf{Z}\mathbf{Z}^T \text{diag}(\boldsymbol{\beta}) - \text{diag}(\mathbf{Z}\mathbf{Z}^T \text{diag}(\boldsymbol{\beta})).$$

This is a *non-negative matrix factorization* (NMF) of the off-diagonal elements of $\mathbf{F}$ into the off-diagonal elements of $\mathbf{Z}\mathbf{Z}^T \text{diag}(\boldsymbol{\beta})$. Thus, one could view our modeling approach as an implicit factorization of the infectivity matrix where the infectivity matrix is unknown but we know the timestamps of users' activities. One could easily see that depending on the structure of the community participation $\mathbf{Z}$, this point of

view allows many interesting scenarios on network infectivity $\mathbf{F}$. For example, cross-group infectivity (Figure 3.1b); dominant rows/columns for a core group and that peripheral individuals only connect via this core group (Figure 3.1c). Note that, in these scenarios, network infectivity has a low-rank structure if we only consider the off-diagonal elements. This factorization perspective opens more research directions to investigate in the future.

The above reasoning inspires us to propose that one should conceptually views network infectivity and community structure being *two sides of the same problem*. We postulate that these characteristics intertwine and that knowing one characteristic of the network should help better revealing the other. In the subsequent sections, we will focus on the technical aspects of the proposed model. We will start with *joint likelihood* definition.

### 3.2.2 Joint likelihood.

In this section, we will define the joint likelihood of the event data. First, we choose a *conjugate prior* for the community participation matrix $\mathbf{Z}$. As it turns out later, we can choose a *Gamma distribution*, $\text{Gamma}(a_g^0, b_g^0)$, as conjugate prior for each of $z_{ig}, i = 1 \dots I, g = 1 \dots G$.

Let us assume that we observed set of $C$ cascades $\{(t_n^c, i_n^c)\}$, $n = 1 \dots N_c$, $c = 1 \dots C$, where $t$'s are the time of events and $i$'s are the identity of users. Given $\mathbf{Z}$, the likelihood of this set of cascades is [17]

$$L(\boldsymbol{t}|\mathbf{Z}) = \prod_{c=1}^{C} \left[ \prod_{n=1}^{N_c} \lambda_{i_n^c}^c(t_n^c) \times \exp\left( -\sum_{i=1}^{I} \int_0^{T_c} \lambda_i^c(t) dt \right) \right],$$

where $\lambda_i^c(t)$ is defined in Eq. (3.2.2) using history of events up to time $t$ in the $c$-th cascade. The *joint likelihood*, the basis of all derivations that follow, is[1]

$$L(\mathbf{Z}, \boldsymbol{t}) \propto L(\boldsymbol{t}|\mathbf{Z}) \times \prod_{i=1}^{I} \mathbb{P}(\mathbf{Z}_i).$$

[1]It is possible to put prior distributions on $\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ and to work in full Bayesian fashion. However, in this work, we only consider these parameters fixed for clarity.

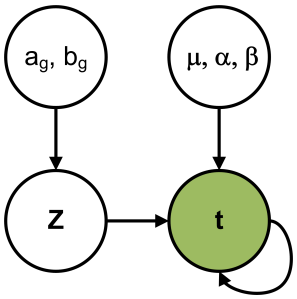



Figure 3.2: The simplified graphical model of the proposed Hawkes process: solid circle indicates observed time-stamped data.

In Figure 3.2, we present the simplified graphical model corresponding to the proposed Hawkes process. In later derivations of the proposed method, we will mainly work with the *log-likelihood* (detailed expression in supplemental material). We will first develop a method for inferring community participation $\mathbf{Z}$ from the observed cascades, i.e. finding the posterior distribution $\mathbb{P}(\mathbf{Z}|\boldsymbol{t})$.

## *3.3 Variational Inference*

As the posterior distribution $\mathbb{P}(\mathbf{Z}|\boldsymbol{t})$ does not have a nice factorized form, in order to proceed, one could apply the *mean field variational inference* framework [77]. Specifically, we use an approximation distribution $q$ to the posterior distribution on $\mathbf{Z}$ such that $\mathbf{Z}_i$'s are independent,

$$q(\mathbf{Z}) = \prod_{i=1}^{I} q_i(\mathbf{Z}_i).$$

Remarkably, this is the only assumption that one needs on the approximation distribution $q$. The goal here is to find a distribution $q$ as close as possible to the true posterior distribution $\mathbb{P}(\mathbf{Z}|\boldsymbol{t})$. To that end, one could utilize the following famous decomposition of the likelihood of observed data

$$\ln \mathbb{P}(\boldsymbol{t}) = \mathbb{E}_q \left[\mathcal{L}(\mathbf{Z}, \boldsymbol{t})\right] + \mathcal{E}[q] + \mathrm{KL}\left(q \| \mathbb{P}(\mathbf{Z}|\boldsymbol{t})\right),$$

where $\mathcal{E}[q]$ is the entropy of $q$ and $\mathrm{KL}\left(q\|p\right) = \mathbb{E}_q\left[\ln(q/p)\right]$ is the Kullback-Leibler divergence between two distribution $q$ and $p$. As one could see from this decomposition,

the better $\mathbb{E}_q\left[\mathcal{L}(\mathbf{Z}, \boldsymbol{t})\right] + \mathcal{E}[q]$ approximates the evidence of observed data, the closer $q$ is to $\mathbb{P}(\mathbf{Z}|\boldsymbol{t})$.

### 3.3.1 Evidence lower bound.

In the followings, we will bound the the expectation of the joint log-likelihood $\mathbb{E}_q\left[\mathcal{L}(\mathbf{Z}, \boldsymbol{t})\right]$ from below so that the inference of $\mathbf{Z}$ is tractable.

**Theorem 3.3.1** (ELBO). *The expectation of joint log-likelihood* $\mathbb{E}_q\left[\mathcal{L}(\mathbf{Z}, \boldsymbol{t})\right]$ *is lower-bounded by*

$$
\begin{aligned}
&\sum\nolimits_{i=1}^{I}\sum\nolimits_{g=1}^{G}(a_g^0-1)\mathbb{E}_q\left[\ln z_{ig}\right] - b_g^0\mathbb{E}_q\left[z_{ig}\right] \\
&+\sum_{c=1}^{C}\left\{\sum_{n=1}^{N_c}\left[\eta_n^c \ln\frac{\mu_{i_n^c}}{\eta_n^c} + \gamma_n^c \ln\left(\frac{\alpha_{i_n^c}\sum_{\ell<n}^{i_\ell^c=i_n}\kappa(t_n^c-t_\ell^c)}{\gamma_n^c}\right)\right.\right. \\
&\left.\left.+\sum_{\ell<n}^{i_\ell^c\neq i_n^c}\sum_{g=1}^{G}\eta_{\ell n}^{gc}\left\{\mathbb{E}_q\left[\ln(\beta_{i_\ell^c} z_{i_n^c g} z_{i_\ell^c g}\kappa(t_n^c-t_\ell^c))\right] - \ln\eta_{\ell n}^{gc}\right\}\right] \right. \\
&-T_c\sum_{i=1}^{I}\mu_i - \sum_{i=1}^{I}\sum\nolimits_{\substack{n=1\\ i_n^c\neq i}}^{N_c}\sum_{g=1}^{G}\beta_{i_n^c}\mathbb{E}_q\left[z_{ig}z_{i_n^c g}\right]K(T_c-t_n^c) \\
&\left.-\sum\nolimits_{i=1}^{I}\sum\nolimits_{\substack{n=1\\ i_n^c=i}}^{N_c}\alpha_i K(T_c-t_n^c)\right\},
\end{aligned}
$$

*in which for the n-th event in the c-th cascade, we have non-negative auxilliary variables* $\eta_n^c, \eta_{\ell n}^{gc}, \gamma_n$ *such that* $\eta_n^c + \sum_{\ell<n}^{i_\ell^c\neq i_n}\sum_{g=1}^{G}\eta_{\ell n}^{gc} + \gamma_c^c = 1$.

The proof could be found in the Appendix. Next, we will optimize the distribution $q(\mathbf{Z})$ and other model parameters (i.e. $\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\beta}$). As we are going to see, the optimal approximation to the posterior distribution turns out to have a nice factorization form.

### 3.3.2 Inferring community participation.

Following the procedure in [77] for mean field variation inference, given the lower bound in the previous section, the optimal distribution $q_i^\star(\mathbf{Z}_i)$ satisfies

$$
\ln q_i^\star(\mathbf{Z}_i) = \mathrm{E}_{q_{-\mathbf{Z}_i}}[\mathcal{L}(\mathbf{Z}, \boldsymbol{t})] + \text{const},
$$

where the expectation is over all $\mathbf{Z}_j, j \neq i$.

From the expression of $\ln q_i^\star(\mathbf{Z}_i)$ (details in supplemental material), one could easily verify that the optimal distribution for $\mathbf{Z}_i$ has a nice factorization into $G$ Gamma distributions. This is remarkable because we do not make any assumption on the parametric form of the distributions $q_i(\mathbf{Z}_i)$'s other than their independence. For each $z_{ig}, g = 1, \ldots, G$, one could update its Gamma distribution parameters as followings

$$a_{ig} = a_g^0 + \sum\nolimits_{c=1}^{C} \sum\nolimits_{n=1}^{N_c} \sum\nolimits_{\ell<n} \eta_{\ell n}^{gc} \delta_{\ell n}^{ic}, \tag{3.3.1}$$

$$b_{ig} = b_g^0 + \sum\nolimits_{c=1}^{C} \left[ \sum\nolimits_{\substack{n=1 \\ i_n^c \neq i}}^{N_c} \beta_{i_n^c} \mathbb{E}_q \left[ z_{i_n^c g} \right] K(T_c - t_n^c) \right. \tag{3.3.2}$$

$$\left. + \sum\nolimits_{j \neq i} \sum\nolimits_{\substack{n=1 \\ i_n^c = i}}^{N_c} \beta_i \mathbb{E}_q \left[ z_{jg} \right] K(T_c - t_n^c) \right], \tag{3.3.3}$$

where $\delta_{\ell n}^{ic} = \begin{cases} 1, & i_n^c = i, i_\ell^c \neq i \text{ or } i_n^c \neq i, i_\ell^c = i, \\ 0, & \text{otherwise} \end{cases}$.

The definition of $\delta_{\ell n}^{ic}$ represents the influence of both *past and future events* on the posterior distribution. The other terms involving $K(\cdot)$ come from the normalization term (also known as the survival term in the field of survival analysis) of the likelihood.

### 3.3.3 Updating auxilliary variables.

After each update of $q$, one could further tighten the bound by the following update formulas[2]

$$\eta_n^c \propto \mu_{i_n^c}, \quad \gamma_n^c \propto \alpha_{i_n^c} \sum\nolimits_{\ell<n}^{i_\ell^c = i_n^c} \kappa(t_n^c - t_\ell^c), \tag{3.3.4}$$

$$\eta_{\ell n}^{gc} \propto \beta_{i_\ell^c} \kappa(t_n^c - t_\ell^c) e^{\mathbb{E}_q\left[\ln z_{i_n^c g}\right] + \mathbb{E}_q\left[\ln z_{i_\ell^c g}\right]}. \tag{3.3.5}$$

Note that, one needs to normalize these auxiliary variables so that their sum is equal to 1. From Eq. (3.3.4), one could interpret these auxiliary variables as the responsibilities of spontaneous rate $\mu_{i_n^c}$, the previous events from other users (i.e. the infectivity

[2]Given $\sum_i x_i = 1$ and $x_i \geq 0, \forall i$, the function $\sum_i a_i x_i - \sum_i x_i \ln x_i$ attains maximum at $x_i^\star = e^{a_i} / \sum_j e^{a_j}, \forall i$.

$\alpha_{i_n^c i_\ell^c}$), and the self-exciting rate $\alpha_{i_n^c}$. In other words, these auxiliary variables are the probabilities that the $n$-th event is triggered by these characteristics of the network.

### 3.3.4 Inferring individual parameters.

For each individual, we need to estimate the spontaneous rate, self-exciting rate, and the celebrity index. As it turns out, these parameters also have the following nice closed-form updates[3]

$$\mu_i = \frac{\sum_{c=1}^{C} \sum_{\substack{n=1 \\ i_n^c = i}}^{N_c} \eta_n^c}{\sum_{c=1}^{C} T_c}, \;\; \alpha_i = \frac{\sum_{c=1}^{C} \sum_{\substack{n=1 \\ i_n^c = i}}^{N_c} \gamma_n^c}{\sum_{c=1}^{C} \sum_{\substack{n=1 \\ i_n^c = i}}^{N_c} K(T_c - t_n^c)}, \tag{3.3.6}$$

$$\beta_i = \frac{\sum_{c=1}^{C} \sum_{\substack{n=1 \\ i_n^c \neq i}}^{N_c} \sum_{\ell < n, \, i_\ell^c = i} \sum_{g=1}^{G} \eta_{\ell n}^{gc}}{\sum_{c=1}^{C} \sum_{j \neq i} \sum_{\substack{n=1 \\ i_n^c = i}}^{N_c} \sum_{g=1}^{G} \mathbb{E}_q \left[ z_{jg} z_{ig} \right] K(T_c - t_n^c)}. \tag{3.3.7}$$

Fortunately, one could compute the expectations in the updates (3.3.1), (3.3.4), and (3.3.6) efficiently as $z$'s are Gamma random variables[4]. To summarize, Algorithm 3.1 outlines the steps of our proposed community detection algorithm, NetCodec. In the output step, we output the mean of Gamma distributions $\mathbf{Z} = \mathbf{A} \oslash \mathbf{B}$ where $\oslash$ is the *element-wise division* operator.

**Algorithm 3.1** NetCodec

1. Input: Set of cascades $\{(t_n^c, i_n^c)_{n=1 \ldots N_c}\}_{c=1 \ldots C}$.
2. Initialization: $\mathbf{A}, \mathbf{B} \in \mathbb{R}_+^{I \times G}$, $\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}_+^I$.
3. While not converged
   (a) For all user $i$
      i. Update $i$-th row of $\mathbf{A}$ and $\mathbf{B}$ using (3.3.1).
      ii. Update auxiliary variables using (3.3.4).
   (b) Update $\boldsymbol{\mu}, \boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ using (3.3.6).
4. Output: $\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{Z} = \mathbf{A} \oslash \mathbf{B}$.

[3] We use the general result $\frac{a}{b} = \arg\max_{x \geq 0} a \ln x - bx, \forall a, b > 0$.

[4] Specifically, if $z \sim \text{Gamma}(a, b)$, $E[z] = \frac{a}{b}, E[\ln z] = \psi(a) - \ln b$, where $\psi(\cdot)$ is the *digamma* function.

### 3.3.5 Implementation issues.

*Stopping criteria.* The convergence detection involves computing the evidence lower bound, ELBO, to $\mathbb{E}_q\left[\mathcal{L}(\mathbf{Z}, \boldsymbol{t})\right] + \mathcal{E}[q]$, where $\mathcal{E}[q]$ is the entropy of the current approximation distribution $q$. In our implementation, we stop the iterations when the relative change of ELBO is below a threshold (e.g. $10^{-4}$). In our experience, the algorithm often stops after less than 40 iterations.

*Number of data sweeps.* From Algorithm 3.1, we could see that, for every update of $\mathbf{Z}_i$ (i.e. the update of the $i$-th row of $\mathbf{A}$ and $\mathbf{B}$), one needs to update the auxiliary variables. This results in one sweep over the data for every update of $\mathbf{Z}_i$. However, to scale to large number of individuals and lengthy cascades, one could leverage a key observation on the evidence lower bound. That is, the lower-bound is valid for *any set* of auxiliary variables. Using careful book-keeping technique, one could reduce the number of data sweeps to *one* in order to update all Gamma distributions of all users.

*Number of relevant historical events.* The computation of the auxiliary variables and the accumulation of the denominators and numerators of model parameters (i.e. $\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\beta}$) involves a nested loop over indices $\ell$ of events that happened before the $n$-th event leading to undesirable $O(N^2)$ complexity. This results in the complexity of each iteration being proportional to $N^2$, where $N$ is the number of events in a cascade. Luckily, one could skip irrelevant historical events where the kernel value $\kappa(t_n - t_\ell)$ is small because the corresponding auxilliary variables would also be very small. This greatly reduces the complexity of the computation to $O(kNG + IG)$ per iteration where $k$ is the average number of relevant historical events.

*Speed up with parallelization.* The computation of auxiliary variables for each event is completely independent of each other. The accumulation of Gamma distribution parameters as well as individual parameters are also independent. These observations are great sources for a parallelized implementation.

*Number of clusters.* One drawback of this modeling technique is the predefined number of clusters. It is possible to address this drawback by using complexity measure such as AIC or BIC score.

## *3.4 Experiment results*

### 3.4.1 Performance Evaluation.

We evaluate the performace of the proposed method using the following criteria

- *Normalized Mutual Information* (NMI): We compare the estimated clusterings $\Omega$ with the ground truth clusterings $\Gamma$ using the NMI score

$$\mathrm{NMI}(\Omega, \Gamma) = \frac{\sum_k \sum_j \mathbb{P}(\Omega_k \cap \Gamma_j) \log \frac{\mathbb{P}(\Omega_k \cap \Gamma_j)}{\mathbb{P}(\Omega_k)\mathbb{P}(\Gamma_j)}}{(\mathcal{E}[\Omega] + \mathcal{E}[\Gamma])/2},$$

  where $\Omega_k, \Gamma_j$ is the $k$-th and $j$-th clusters in $\Omega$ and $\Gamma$, respectively, and $\mathcal{E}[\Omega]$, $\mathcal{E}[\Gamma]$ are their entropies. The NMI score is a value between 0 and 1, with 1 representing perfect cluster matching. To assign users to clusters, we use the maximum elements in each row of $\mathbf{Z}$.

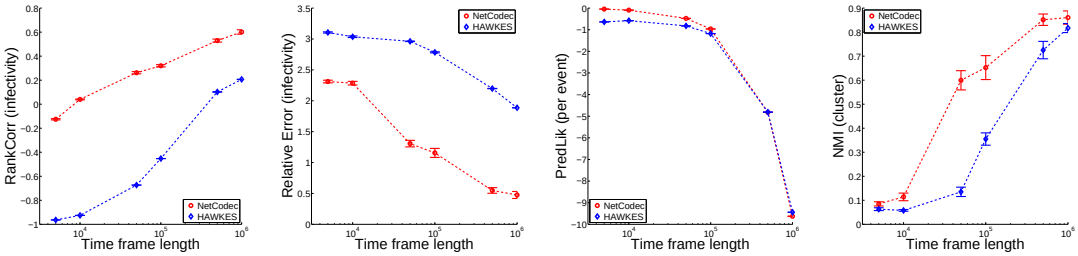
- *Kendall Rank Correlation* (RankCorr): We compare the estimated celebrity index $\boldsymbol{\beta}$ with the ground truth using the following score

$$\mathrm{RankCorr}(\boldsymbol{x}, \boldsymbol{y}) = \frac{N_{\text{concordant}} - N_{\text{discordant}}}{I * (I-1)/2},$$

  where $N_{\text{concordant}}$ is the number of pairs of indices $(i, j)$ that $x_i > x_j$ and $y_i > y_j$, or $x_i < x_j$ and $y_i < y_j$. The RankCorr score is a value between -1 and 1, with 1 representing perfect rank matching.

- *Relative error* (RelErr): We compare the infectivity matrices $\mathbf{F}$ using the average relative errror of their elements. Specifically, we have

$$\mathrm{RelErr}(\mathbf{F}_1, \mathbf{F}_2) = \frac{1}{I^2} \sum\nolimits_{i,j=1}^{I} |\alpha_{ij}^2 - \alpha_{ij}^1| / |\alpha_{ij}^1|$$




(a) RankCorr (b) RelErr (c) PredLik (d) NMI

Figure 3.3: Cross-group infectivity scenario: comparison to ground truth (left) average RankCorr of columns of network infectivity matrix; (middle) average RelErr of elements of the infectivity matrix; (right) predictive log-likelihood on test data.

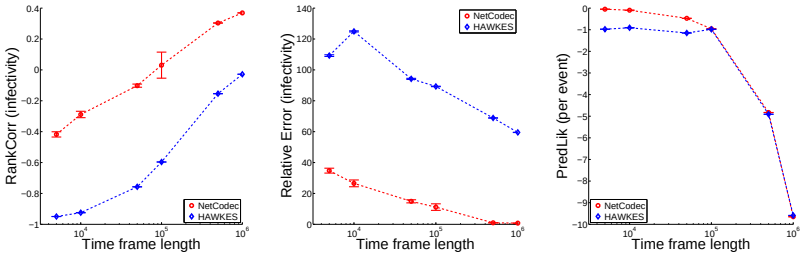
- *Predictive log-likelihood* (PredLik): We also compute the log-likelihood of a hold-out test data set in order to show the predictive power of the compared models.

Note that, because of the factorization, at best, one could only recover $\mathbf{Z}$ and $\boldsymbol{\beta}$ up to a constant factor. Therefore, the NMI and RankCorr scores are more suitable criteria than the absolute error or squared error when comparing the participation matrices and the vectors of celebrity indices.

### 3.4.2 Synthetic data.

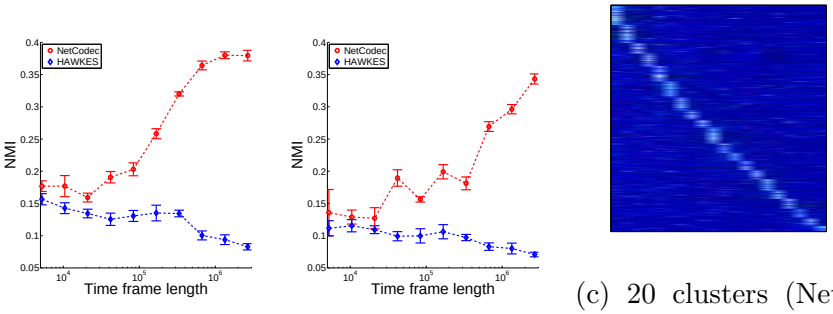
We start with experiments with simulated data where we know the ground truth network infectivity. We generate the ground truth parameters $\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ randomly to satisfy certain stability conditions[5]. The parameters form a network of 500 nodes. We then generate event cascades with different time frame length settings and also generate a hold-out set of the same size to use as test set. The time frame lengths are $(10^3, 5 \times 10^3, 10^4, 5 \times 10^4, 10^5, 5 \times 10^5, 10^6)$. In total, there are about $3 \times 10^5$ events when $T = 10^6$. We run each experiment 10 times and take the average of the scores over all the 10 runs. We then verify the convergence of the proposed method by varying the time frame of the simulations.

[5]The spectral norm of $\mathbf{F}$ is less than 1.




(a) RankCorr (b) RelErr (c) PredLik

Figure 3.4: Core group scenario.




(a) NMI (10 clusters) (b) NMI (20 clusters) (c) 20 clusters (NetCodec)

Figure 3.5: Clustering results on MemeTracker dataset.

We generate data according to two scenarios:

- The nodes form 10 clusters and there are some cross-group infectivity.
- There is a core group and the remaining nodes only connect via this core group.

In Figure 3.3 and 3.4, we report the performance of the proposed method in comparison with the Hawkes MLE solver (denote HAWKES in the figures) in [87] in the two aforementioned scenarios. The figures show that both NetCodec and HAWKES are able to increase their performance when the time frame length increases. However, in comparison to the ground truth, NetCodec outperforms HAWKES in all performance measures given the same time frame length. This could be attributed to the fact that NetCodec models the low rank assumption directly and as a result, it needs to estimate fewer parameters, hence the better performance in both area. Especially in the case that there is a core group (Figure 3.4), there are a lot of near zero

elements in the infectivity matrix making accurate recovery of these elements very difficult. This explains the high RelErr that both algorithms encounters. However, when there are enough data, NetCodec is able to recover the infectivity matrix much better than HAWKES.

In Figure 3.3d, we show that NMI score of NetCodec and HAWKES with respect to the ground truth clusterings. As HAWKES provides no clustering, its clusterings are computed via a spectral clustering [65] of the infectivity matrix. One could see that while both algorithms are able to recover the clusterings with enough data, NetCodec outperforms HAWKES when data are insufficient.

### 3.4.3 Real-world event data.

**MemeTracker.** We extract events of the most active sites from the MemeTracker dataset[6]. This dataset contains the times that articles are published in various websites/blogs from August 2008 to April 2009. We select most active 500 sites with about 8 million events from these sites.

We use the MemeTracker data provided links between articles to establish an estimated ground truth of the clusters. To this end, we count the number of links between all pairs of sites to build a similarity matrix. We then run a spectral clustering algorithm [65] on this similarity matrix with different settings on the number of clusters. While one could choose the number of clusters based on model scores (i.e. data log-likelihood plus model complexity) such as Bayesian or Akaike information criterion, here, for demonstration purpose, we set the number of clusters to be 10 and 20. We then run NetCodec and HAWKES on the timestamped data only (i.e. without the link information) to find out if these algorithms could recover the estimated ground truth clusterings. As mentioned in the experiments on synthetic datasets,

[6]http://www.memetracker.org/data.html

the clusterings for HAWKES are computed via spectral clustering on the estimated infectivity matrix.

In Figure 3.5b and Figure 3.5b we shows the NMI scores of these algorithms with respect to the ground truth estimated from the similarity (count) matrix when the number of clusters set to 10 and 20. One could see that in both settings NetCodec is able to recover part of the clusterings while HAWKES fails on this dataset.

In Figure 3.5c, we visualize the clustering result (i.e. the participation matrix $\mathbf{Z}$). Detailed examination (Figure 3.2) of the clusters produced by NetCodec shows some consistent clusters spanning common categories. Examples of clusters found by NetCodec and their respective popular websites having with high celebrity index are news (reuters.com, npr.org), business (businessweek.com, forbes.com, cbsnews.com), and technology (hardwarezone.com, digitalcamerareview.com). There are consistent clusters with nationality identity such as Brazilian sites, Japanese sites, Italian sites. One should note that the clusters are formed using purely timestamps of activities/events happened on this sites. The results show that the activities on these sites allow us to group them into meaningful clusters.

**Earthquake.** The next dataset that we investigate is the Earthquake dataset[7]. We download 16000 earthquakes that have minimum magnitude 4 in the 12 months from Oct. 2013 to Oct. 2014. The earthquake information contains location (i.e. longtitude, lattitude) and timestamps in seconds (see Figure 3.6, red dots are big cities, colored bigger dots are earthquake locations). In this experiments, we only use the timestamps of the earthquakes (divided by 3600 to convert to hours) as input to the inference algorithms to investigate if timestamped information results in a coherent clustering. To establishes the identities of events (i.e. the $i$'s variables), we draw a longtitude/lattitude grid on the global map and all earthquakes that occur in a grid square (of size $2 \times 2$) will have same identity. In total we have 1021 identities and

[7]http://earthquake.usgs.gov/earthquakes/

Table 3.2: Representative sites (high celebrity index) in 10 clusters.

| Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 |
|---|---|---|---|
| seattle.craigslist.org<br>sfbay.craigslist.org<br>losangeles.craigslist.org<br>newyork.craigslist.org<br>sandiego.craigslist.org<br>boston.craigslist.org | ameblo.jp<br>blog.livedoor.jp<br>rss.rssad.jp<br>pheedo.jp | economia.uol.com.br<br>noticias.uol.com.br<br>esporte.uol.com.br | latimes.com<br>cnn.com<br>reuters.com |
| Cluster 5 | Cluster 6 | Cluster 7 | Cluster 8 |
| businessweek.com<br>news.google.com<br>financial.de<br>forbes.com | ar.answers.yahoo.com<br>it.answers.yahoo.com<br>fr.answers.yahoo.com<br>de.answers.yahoo.com | torrentportal.com<br>torrentz.com<br>torrentreactor.net | cbsnews.com<br>telegraph.co.uk |
| Cluster 9 | Cluster 10 | | |
| google.com<br>chicagotribune.com<br>thefacts.com<br>boston.com | topix.net<br>indeed.com<br>indiatimes.com | | |

our goal is to classify these identities into clusters. We run NetCodec with exponential kernel ($\lambda = 0.04$) and report the clustering result in Figure 3.6. The parameter $\lambda$ was chosen to be the average occurence rate of the identities.

One could see that there are geological regions where earthquakes form clusters. This is remarkable as we use only timestamped information. The location information are used only to form identies and then discarded. More detailed discussion could be found in the Appendix.

Our model only works with timestamp information of the earthquakes. It is possible to augment the proposed model to incorporate more information such as magnitude, location. The possible directions that we want to explore in the near future are

- Using a different triggering kernel function that take into account the additional information, after all, the triggering kernel function could be considered as similarity function between events.

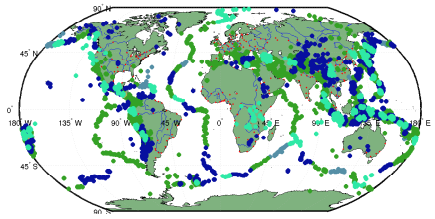


Figure 3.6: Clustering results on Earthquake dataset.

- Using a different factorization of the intensity function similar to what was discussed in previous sections.

## *3.5 Conclusion*

In this work, we propose that one could infer the network of social influence along with its community structure from the observed recurrent events in the social networks. To that end, we utilize the key observation that regular activities often raise influence among users in the same group. The proposed model based on the Hawkes model is designed to take into account this observation and other assumptions such as the low-rank structure. The inference algorithm following the mean-field variational principle nicely consists of closed form updates that could be sped up by various implementation techniques including parallelism. The experiments on simulated dataset show that the proposed model could estimate both network infectivity and and community structure and produce better predictive model with less training samples than the baseline methods. Experiments on real dataset show that the proposed method are able to produce meaningful clusters using only activities from websites.

There are interesting paths to extend this study: First, we plan to investigate the extensions that cover other features of an event, for example, document content and ratings. The content and ratings effects on community structure could be expressed in the factorization of the influence between events. Moreover, it is also interesting to incorporate the memes/trends and community structure in one model.

# CHAPTER IV

# INTERVAL-CENSORED INFERENCE

## *4.1 Introduction*

Recently, multi-dimensional Hawkes processes have been shown to be an effective method to model the self-exciting property — an existing event can trigger future events — which exists in a lot of natural and social phenomena, such as disease epidemics and information propagation in social networks. Hawkes processes are quite different from the well-know Poisson processes whose memoryless property (i.e. number of events after time $t$ is independent of the number of events before time $t$) restricts one from modeling the long range influence of individual events over time.

Existing algorithms to estimate the parameters of multi-dimensional Hawkes processes rely on the exact time-stamps of the events (the times at which the individual events occur). For several applications, this type of information is readily available; for example, Twitter can use its logging system to record the exact time when a user posts a tweet or when a user retweets a tweet from another user; it can also export the information to a third party through a well-designed API. However, in other real world applications, it is usually difficult or even impossible to obtain these exact time-stamps due to limitations in resources, technology, or user privacy. Instead, it is relatively easy to obtain interval-censored event data, i.e., the aggregated number of events over time intervals, in a lot of applications. For example, for search engines, it is usually difficult to know the exact time-stamps of newly created links. However, by comparing the snapshots from two crawls of a web graph, it is easy to obtain how many new links that a certain page has obtained during the corresponding time period. Similarly, in disease epidemics research, it is often impossible to obtain the

exact time of the infections, but the aggregated statistics during a certain time period are easily available. In these cases, traditional *maximum likelihood estimator* (MLE) for Hawkes process models is not applicable since the exact time stamps are not available. This drawback largely limits the applications of multi-dimensional Hawkes processes, which is a quite unsatisfactory situation considering the importance of these applications.

Can we estimate the parameters in Hawkes processes based on interval censored event data? Although MLE for Poisson processes with interval-censored data has been derived for the likelihood [71, 31], these analysis frameworks are not applicable to Hawkes process models. One challenge is that it is not easy to formulate the likelihood function of interval-censored data for Hawkes process models. A second challenge is that the memoryless property of Poisson processes, which is key to its tractability, is not present in Hawkes process models.

To address these challenges, we propose a latent variable model where each censored event is associated with a hidden time-stamp. Furthermore, we propose a new machine learning algorithms, based on Monte-Carlo EM, that expands the capability of Hawkes process models to handle interval-censored event data. In the Monte-Carlo EM, we sample the event time-stamps (hidden variables) based on current estimates of the model parameters and observed event counts, and then use the sampled event times to re-estimate the model parameters. One key technical challenge is how to sample the event time-stamps given the event counts in an interval, since all event time-stamps are inter-dependent. We propose a Gibbs sampling algorithm, which we show that it produces much higher quality samples compared to four simple baselines sampling methods. This increased quality of the samples also leads to increased quality of the estimated parameters.

### 4.1.1 Related Works

The Hawkes process [39] is an important model for capturing the self and mutual exciting properties of temporal event data. Hawkes processes have been used to model association of temporal events in various fields, such as finance events [29], seismic events [66, 59], social interactions [61, 87], crime modeling [70], civilian deaths in conflicts [54], and recently causal militant conflict events [55].

Beside situations where exact time-stamp data are avaiable , interval-censored situations where the exact time-stamps are not available are also abundant. For example, disease/epidemiology studies often encounter situations where patients are assessed only at pre-scheduled visits. Therefore, the exact time of infection is only known to happen between the last visit and the following visit. Parameter estimation using interval censored data for simpler point processes mainly focuses on the Poisson processes [71] using the maximum likelihood estimator (MLE) [71] or local likelihood [31]. Non-parametric approaches for non-homogeneous Poisson processes use the pseudo MLE [72] or full MLE [76]. Consistency for the MLE has been established in some special cases [76]. However, one disadvantage of the Poisson processes is the memoryless property by which number of events after any time $t$ is independent of the number of events before time $t$. This restricts one to model influence of individuals over time.

Multiple imputation (MI) [69] is a general framework to stochastically impute incomplete or missing data from the current model in order to build a surrogate dataset of observations. Tanner and Wong [73] explore the notion of MI for nonparametric estimation of the hazard function using grouped and interval-censored data. While there has been many studies on *multivariate interval-censored data* [13], [52], current literature still lacks analysis of *social interval-censored data* where events of an individual have influence on later events of him or herself and other individuals. Next,

we will discuss the Hawkes process, its maximum likelihood estimator, the interval-censored data estimation problem and the imputation approach for this problem.

## *4.2 Estimation from Interval-censored Data*

Recall from Chapter 2, the intensity functions for an $U$-dimensional Hawkes processes is

$$\lambda_u^*(t) = \mu_u + \sum\nolimits_{i:t_i<t} a_{uu_i} g(t - t_i), \tag{4.2.1}$$

where $\mu_u \geq 0$ is the base intensity for the $u$-th Hawkes process and $g(t)$ is the decaying kernel. For $m$ cascades where each cascade $c = 1, 2, \ldots, m$ is a sequence of events observed in the time interval $[0, T_c]$ in the form $\{(t_i^c, u_i^c)\}_{i=1}^{n_c}$, the log-likelihood is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}) = \sum_c \Bigg[ \sum_{i=1}^{n_c} \log \left( \mu_{u_i} + \sum_{j:t_j<t_i} a_{u_i u_j} g(t_i - t_j) \right) \\ - T_c \sum_{u=1}^{U} \mu_u - \sum_{u=1}^{U} \sum_{j=1}^{n_c} a_{uu_j} G(T_c - t_j) \Bigg].\end{aligned} \tag{4.2.2}$$

where $G(T_c - t_j) := \int_0^{T_c} g(t - t_j) dt$.

When the exact timestamps are given, one could use the MLE algorithm described in Chapter 2 to estimate the parameters. However, the MLE algorithm does not work when the event are interval-censored. In the next section, we will discuss this interval-censored estimation problem.

### 4.2.1 Problem statement

Let us partition the observation window $[0, T]$ into $K$ non-overlapping interval $[a_i, b_i), i = 1, 2, \ldots, K$. The counting process $N_t$ is not continuously observed in $[0, T]$. Instead, only the panel count $c_{iu}$ (or the number of events in $u$-th dimension) in the interval $[a_i, b_i)$ is available. Let $\boldsymbol{c}_i = (c_{iu})_{u=1}^n$ be the count vector in the $i$-th interval and

$\boldsymbol{C} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K)$. We need to find the parameters that maximizes the likelihood of the *interval-censored* data.

$$\boldsymbol{\Theta}^{ic} = \arg\max_{\boldsymbol{\Theta}} \mathcal{L}^{ic}(\boldsymbol{\Theta}) \triangleq \log \boldsymbol{P}(\boldsymbol{C}; \boldsymbol{\Theta}). \tag{4.2.3}$$

Since the event time-stamp and dimension are not observed, we introduce latent variables $\mathbf{t}$ and $\mathbf{u}$ to account for that. The maximization in Eq. (4.2.3) is harder to optimize than the maximum likelihood of the complete data (i.e. with exact time-stamps) in Eq. (**??**) because of the integral in $\boldsymbol{P}(\boldsymbol{C}; \boldsymbol{\Theta})$:

$$\mathcal{L}^{ic}(\boldsymbol{\Theta}) = \log \sum_{\mathbf{u}} \int_{\mathbf{t}} \mathbb{P}(\mathbf{t}, \mathbf{u}, \mathbf{C}; \boldsymbol{\Theta}) d\mathbf{t},$$

where $\mathbb{P}(\mathbf{t}, \mathbf{u}, \mathbf{C}; \boldsymbol{\Theta})$ equals to the likelihood of the complete data (see Eq. (4.2.2)) whenever $\mathbf{t}, \mathbf{u}$ satisfies the count $\mathbf{C}$ and equals to 0 otherwise.

### 4.2.2 Monte-Carlo EM

Considering $\mathbf{t}, \mathbf{u}$ as hidden variables and $\mathbf{C}$ observed, and following the EM paradigm, at iteration $k$, we have

- *E-step*: The posterior distribution

$$\mathbb{P}^{(k)} \equiv \mathbb{P}(\mathbf{t}, \mathbf{u} | \mathbf{C}; \boldsymbol{\Theta}^{(k)}).$$

- *M-step*:

$$\boldsymbol{\Theta}^{(k+1)} = \arg\max_{\boldsymbol{\Theta} \geq 0} Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(k)}) \triangleq E_{\mathbb{P}^{(k)}} \log \mathbb{P}(\mathbf{t}, \mathbf{u}, \mathbf{C}; \boldsymbol{\Theta}).$$

Even with closed form posterior distribution in the E-step, direct maximization in the M-step is still intractable because of the integral in expectation. The Monte-Carlo EM (MC-EM) algorithm [75, 53], a modification of the EM algorithm in which the expectation in the M-step is computed numerically via sampling, could potentially overcome this obstacle. Once the samples are available, the empirical M-step becomes

$$\max_{\boldsymbol{\Theta} \geq 0} Q^{\mathrm{MC}}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(k)}) \triangleq \frac{1}{S} \sum_{s=1}^{S} \log \mathbb{P}(\mathbf{t}^s, \mathbf{u}^s, \mathbf{C}; \boldsymbol{\Theta}), \tag{4.2.4}$$

**Algorithm 4.1** Parameter estimation via imputation

Input: $a_i, b_i, c_{iu}, i = 1 \dots K, u = 1 \dots U$
Initialize $\boldsymbol{\Theta}^{(1)} = (\boldsymbol{\mu}^{(1)}, \mathbf{A}^{(1)})$
**for** $k = 1, 2, \dots$ **do**
  Impute $\mathbf{t}, \mathbf{u}$ satisfying the counts $\mathbf{C}$ (Algs. 4.3, 4.2).
  Re-estimate $\boldsymbol{\Theta}^{(k+1)}$ with MLE solver.
**end for**

where $\mathbf{t}^s, \mathbf{u}^s$ are samples following the posterior distribution $\mathbb{P}(\mathbf{t}, \mathbf{u}|\mathbf{C}; \boldsymbol{\Theta}^{(k)})$, and $S$ is the number of samples. One could easily notice that the solver for the MLE problem (**??**) could be re-used with minor changes in order to solve problem (4.2.4). The advantage of using the MC-EM algorithm is that we could re-use high performance solver tailored for the Hawkes MLE problem discussed in the previous section. We summerise the MC-EM algorithm for interval-censored data in Algorithm 4.1. In the following, we will discuss the sampling methods for the distribution in the E-step.

## *4.3 Posterior distribution in E-step*

Let a sample $\boldsymbol{t} = (t_1, t_2, \dots, t_n)$ be a sample from Hawkes process and let us define a mapping

$$\Lambda_u(t) = \int_0^t \lambda_u^*(s) ds.$$

Let $c_{iu} = \sum \mathbf{1}_{t_i \in [a_i, b_i), u_i = u}$, $\boldsymbol{C} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K)$, and $d_i^u = \sum \mathbf{1}_{\Lambda(t_i) \in [\Lambda(a_i), \Lambda(b_i)), u_i = u}$, $\boldsymbol{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_K)$. By the *time-change theorem* [7, 18], $\Lambda_u(\boldsymbol{t}) = (\Lambda_u(t_i))_{u_i = u}$ is a Poisson process with *unit intensity*. Because $\Lambda(\cdot)$ is a monotone, one-to-one mapping, we have

$$\begin{aligned}
\boldsymbol{P}_{\boldsymbol{t}}(\boldsymbol{C}) &= \boldsymbol{P}_{\Lambda}(\boldsymbol{D} = \boldsymbol{C}) \\
&= \prod_{i=1}^{K} \prod_{u=1}^{U} \frac{\exp\left\{-\Lambda_u(t) \left|_{a_i}^{b_i}\right.\right\} \left(\Lambda_u(t) \left|_{a_i}^{b_i}\right.\right)^{c_{iu}}}{c_{iu}!}, \\
&= \prod_{i=1}^{K} \prod_{u=1}^{U} \frac{\exp\left\{-\int_{a_i}^{b_i} \lambda_u^*(t) dt\right\} \left(\int_{a_i}^{b_i} \lambda_u^*(t) dt\right)^{c_{iu}}}{c_{iu}!},
\end{aligned}$$

where $\Lambda_u(t)\Big|_{a_i}^{b_i} = \Lambda_u(b_i) - \Lambda_u(a_i)$ and $\frac{\exp\left\{-\Lambda_u(t)\Big|_{a_i}^{b_i}\right\}\left(\Lambda_u(t)\Big|_{a_i}^{b_i}\right)^{c_{iu}}}{c_{iu}!}$ is the probability of $c_{iu}$ events in the interval $[\Lambda_u(a_i), \Lambda_u(a_i))$ under unit intensity Poisson process.

The posterior distribution of a sequence of events given the panel count could be computed via the Bayes formula by dividing $\boldsymbol{P_t}(\boldsymbol{C})$ from the exponential of the log-likelihood in Eq. (4.2.2). Noticing that the exponentials cancel out and the counts are constant, we have

$$\begin{aligned}\boldsymbol{P}(\boldsymbol{t}, \boldsymbol{u}|\boldsymbol{C}) &= \frac{\boldsymbol{P}(\boldsymbol{t}, \boldsymbol{u}, \boldsymbol{C})}{\boldsymbol{P}(\boldsymbol{C})} \\ &= \prod_{i=1}^{K} \prod_{t_j \in [a_i, b_i)} \frac{\lambda^*_{u_j}(t_j)}{\int_{a_i}^{b_i} \lambda^*_{u_j}(t)dt} \times \text{const.} \end{aligned} \tag{4.3.1}$$

## *4.4 Sampling methods for M-step*

### 4.4.1 Gibbs sampling

The difficulty in sampling given the count of events lies in the fact that all $\lambda^*_u(t)$'s depend on the history of events before time $t$ and that the number of hidden random variables (i.e. the events) is large. At first sight, one may consider sampling using the Metropolis-Hastings (M-H) algorithm. However, with large number of events, the MH algorithm needs to discard many samples (jumping) in order to get rid of correlated samples. Furthermore, the jumping width is also a heuristic that often requires manual tuning. Let us consider a more tractable version of M-H algorithm which is the Gibbs sampling algorithm [10]. In our case, the Gibbs algorithm needs the conditional probability of an event given other events, $p(t_j, \mathbf{u}|\mathbf{t}_{-j})$. This conditional probability is not readily available. Instead, we could compute joint density $p(\mathbf{t}, \mathbf{u}) = \text{const} \times p(t_j, \mathbf{u}|\mathbf{t}_{-j})$ which is the likelihood of the timestamps (exponential of the log-likelihood in Eq. (4.2.2)). The new timestamp $t'_j$ could be sampled using Metropolis-Hastings algorithm in *one dimension*. Specifically, we sample $t'_j$ using a proposal distribution $Q(t'_j|t_j)$ around $t_j$. This distribution should have support on $[a_i, b_i]$, the

**Algorithm 4.2** Imputation with Gibbs sampling

```
Input: μ, A, a_i, b_i, c_iu, i = 1...K, u = 1...U
Initialize: Generate t, u satisfying C with RAND.
for i = 1, 2, ..., K do
  for u = 1, 2, ..., U do
    for j = 1, 2, ..., c_iu do
      Sample t'_j from proposal distribution Q(t'_j|t_j)
      Compute α = P(t',u) / P(t,u)
      if α ≥ 1 then
        Accept t ← t'.
      else
        With probability α, accept t ← t'.
      end if
    end for
  end for
end for
```

interval containing the timestamp $t_j$. In the experiment, the proposal distribution $Q(t'_j|t_j)$ is chosen to be uniform on $[a_i, b_i]$. With Metropolis-Hastings sampling, we accept the new timestamp based on the ratio $\alpha = \frac{p(\mathbf{t}')}{p(\mathbf{t})}$. If $\alpha \geq 1$, we accept $t'_j$, otherwise we accept only with probability $\alpha$. The detailed steps for Gibbs sampling is provided in Algorithm 4.2. We also denote this sampling method GIBBS. The most important property of Gibbs sampler is that the random sequence it generates forms a Markov chain with stationary state being the joint distribution $P(\mathbf{t}, \mathbf{u}|\mathbf{C})$. To achieve a close approximate of the stationary distribution, one needs to ignore some samples generated at the beginning (we skip 100 samples).

### 4.4.2 Approximate sampling

We next discuss a few simple heuristic sampling methods which will be used as baselines.

- Mid-point (MID): The first method one could think of is to impute all timestamps at the *mid-point* of their corresponding intervals. Although this method is naive, it still has potential when the interval size is small, as smaller the interval size is, the more exact the timestamps are regardless of the simulation

method.

- Equal-spacing (EQUAL): The second simple sampling method we will use as baseline is to sample all timestamps that are equally spaced in the corresponding interval. Specifically, let there be $c_i = \sum_u c_{iu}$ timestamps in interval $[a_i, b_i]$. The distance between timestamps will be $d_i = (b_i - a_i)/c_i$. One could set the timestamps at $t_j = a_i + jd_i - d_i/2$ for $j = 1, 2, \ldots, c_i$. The identities $u_j$ of each timestamps are set randomly so that there are $c_{iu}$ events in dimension $u$.

- Uniformly random (RAND): The third sampling method is inspired by an important result for Poisson processes [18]. It states that given the count, the timestamps from a homogeneous Poisson process is uniformly distributed on the interval. Although our process is not homogeneous Poisson, one could still use it as a baseline. In this sampling method, one samples $c_i$ timestamps from the uniform distribution $\mathcal{U}[a_i, b_i]$. The identities of these events are also selected randomly satisfying the count $c_{iu}$ for each dimension $u$.

- Intensity-based simulation (INTSIM): The fourth method is based on the intensity function of the Hawkes process. The main difficulty in sampling is that the intensity function is not known because it depends on the samples itself. Equation (4.3.1) suggests a simple sampling method: At the beginning of each interval, we form the density function $\frac{\lambda_u*(t)}{\int_{a_i}^{b_i} \lambda_u^*(t)dt}, t \in [a_i, b_i]$ for each dimension $u$. Then one could sample $c_{iu}$ timestamps from this density for events in dimension $u$ in this interval. We denote this method INTSIM (Algorithm 4.3).

### 4.4.3 Quality of sampling methods

Let us first discuss an intuitive result on the quality of the previously discussed sampling methods. In order to verify how a sampling method works, one could again leverage the *time change theorem*. It states that, using the time-mapping

**Algorithm 4.3** Intensity based imputation

Input: $\boldsymbol{\mu}, \mathbf{A}, a_i, b_i, c_{iu}, i = 1 \dots K, u = 1 \dots U$
**for** $i = 1, 2, \dots, K$ **do**
  **for** $u = 1, 2, \dots, U$ **do**
    Compute $\lambda_u(t), t \in [a_i, b_i)$
    Sample $c_{iu}$ events of dimension $u$ in $[a_i, b_i)$ with probability density proportional to $\lambda_u(t)$.
  **end for**
**end for**

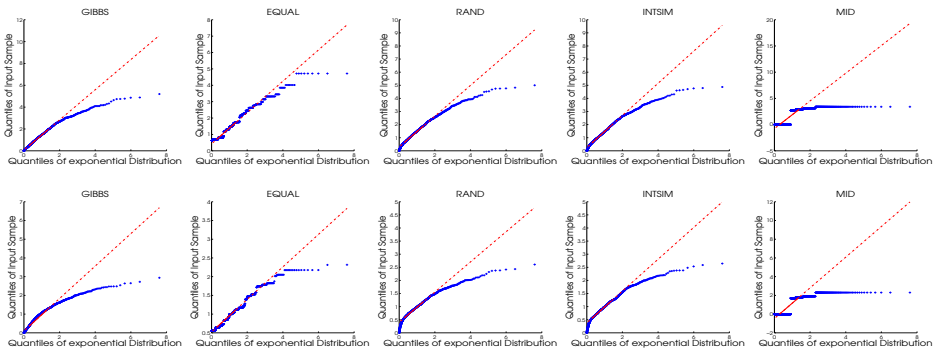



Figure 4.1: Quality of different simulation methods for 2D Hawkes process: QQ-plot of samples from different simulation method (after a time-change mapping) in comparison to the exponential distribution with mean parameter 1. Each row corresponds to one dimension of the 2D Hawkes process. GIBBS sampling method matches Exponential distribution up to the third quantile.

$\Lambda_u(t) = \int_0^t \lambda_u^*(\tau) d\tau$, the mapped timestamps $\Lambda_{u_1}(t_1), \Lambda_{u_2}(t_2), \dots, \Lambda_{u_n}(t_n)$ of a casacde $\{(t_i, u_i)\}, i = 1, \dots, n$ is $U$ independent homogeneous Poisson processes with uniform intensity 1. This provides a neat way to verify the quality of the sampling methods because the difference between two consecutive mapped timestamps in the same dimension should follow Exponential distribution with mean parameter 1. Figure 4.1 shows the qq-plot of samples from the previously discussed sampling methods with respect to the Exponential distribution. One could see that the GIBBS matches Exponential distribution better at the expense of more intensive computation. INTSIM and RAND provide good approximation while EQUAL and MID are worse. This shows that the Gibbs sampler is a good candidate for using in the MC-EM framework. The reason

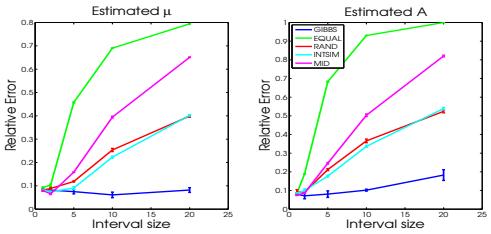



Figure 4.2: Effect of interval size on the estimation quality.

for its better quality lies in the fact that it samples timestamps based on all *past and future timestamps* while other methods use only the current interval or past timestamps. In the next section, we will investigate empirically if this intuitive result holds while using the sampling methods as sub-routines in the MC-EM algorithm to estimate Hawkes processes parameters.

## *4.5 Experiments*

### 4.5.1 Synthetic data

We first investigate the capability of our proposed methods in comparison with known ground truth parameters using synthetic data.

**Evaluation metric.** We use the relative error to measure the performance of the methods. `RelErr` is defined as the averaged relative error between the estimated parameters and the true parameters,

$$e_{uv} = \begin{cases} \frac{|a_{uv} - a'_{uv}}{|a_{uv}|}, & a_{uv} \neq 0 \\ a'_{uv}, & a_{uv} = 0 \end{cases}, \text{RelErr} = \frac{1}{U^2} \sum_{u,v=1}^{U} e_{uv}.$$

To show the confidence of the reported measures, we repeat all following experiments 10 times and compute the average. The standared deviation is shown as error bars in the graphs.

**Interval-censored data.** We generate event data from a $U$-dimensional Hawkes process with $U = 10$ with the true parameters $\boldsymbol{\mu}$ and $\mathbf{A}$ with random nonzero elements

following uniform distribution Uniform$(0, 1/U)$. The nonzero elements of $\mathbf{A}$ are chosen randomly. The influence matrix $\mathbf{A}$ is then scaled so than its *spectral radius* is 0.8 to ensure that the point process is well-defined, i.e., with finite intensity. Then, 100 cascades are sampled from the multi-dimensional Hawkes process specified by $(\boldsymbol{\mu}, \mathbf{A})$ with observation window $T = 100$. While we have tried other kernel functions with similar results, in the following experiments, we focus on the exponential triggering kernel $g(t) = \lambda e^{-\lambda t}$ and generate cascades with mean parameter $\lambda = 1$.

In this experiment, we investigate the capability of the simulation methods in reconstructing the ground truth parameters from interval-censored data. To this end, we censored the 100-cascade data described above with different interval size $I = 1, 2, 5, 10, 20$. We include in the comparison 5 simulation methods: GIBS, EQUAL, RAND, INTSIM, and MID described in previous section.

Figure 4.2 plots the accuracy of the estimated $\boldsymbol{\mu}$ and $\mathbf{A}$ from the methods we proposed. All methods perform well when the interval size is small. This is expected because when the interval size is small, especially when smaller than the mean parameter $\lambda$, all simulation methods, even the naive ones could guess the location of the events on the time-line accurately. Except for GIBBS, all baseline methods perform much worse when the interval size increases. When the interval size raises to 20, the relative errors on $\mathbf{A}$ of EQUAL and MID are more than 0.8 meaning that they cannot recover any ground truth parameter at all. RAND and INTSIM still could recover part of the parameters. GIBBS shows outstanding performance as it is much less sensitive to interval size than other simulation methods.

**Partially censored data.** In this experiment, we investigate the effects of the proportion of censored interval on the quality of the estimation. Given $p \leq 100$ and number of intervals $K$, we randomly choose $\lfloor pK/100 \rfloor$ intervals to be censored from the algorithm. Exact timestamps of events in the remaining intervals are known to the algorithm. Figure 4.3 shows the effects of $p$ on the estimation accuracy. It again

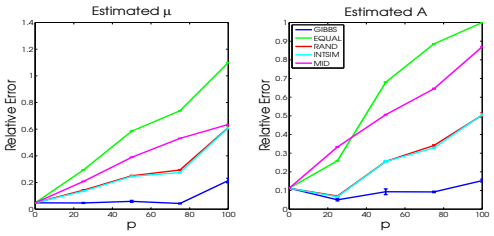



Figure 4.3: Relative error with respect to the percentage of cencored intervals.

shows that MC-EM with GIBBS sampling method is less sensitive to the number of censored intervals. The relative error of GIBBS increases much less than that of the baseline methods when $p$ increases. Another observation is that MC-EM with GIBBS sampling provides close solution to the solution when all timestamps are known exactly (i.e. MLE with uncensored data, or $p = 0$). This is desirable as it shows one still can estimate Hawkes processes parameters with reasonable accuracy using only the counts of events.

### 4.5.2 Karate club's network

We evaluate the proposed methods on a real-world graph, the Karate club's graph[1] [83, 58]. This graph represents the social network of friendships among 34 members of a karate club at a US university in the 1970s (Figure 4.4). We use the degree of each node to generate a matrix $\mathbf{A}$ of influence between club members. We then generate events following a Hawkes process $(\boldsymbol{\mu}, \mathbf{A})$ in observation window $T = 1000$. The events are interval-censored in equal-length intervals $0 : 20 : 1000$. We then threshold the estimated matrix $\mathbf{A}$ from the MC-EM algorithm to generate an estimated graph. We compute the true positive (TP) and true negative (TN) measures

$$\text{TP} = \frac{\#\text{correctly detected edges}}{\#\text{edges}},$$
$$\text{TN} = \frac{\#\text{correctly detected non-edges}}{\#\text{non-edges}}.$$

[1] http://www-personal.umich.edu/ mejn/netdata/

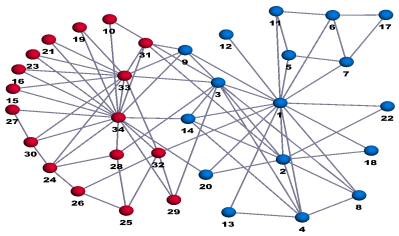


Figure 4.4: Karate club's graph [83].

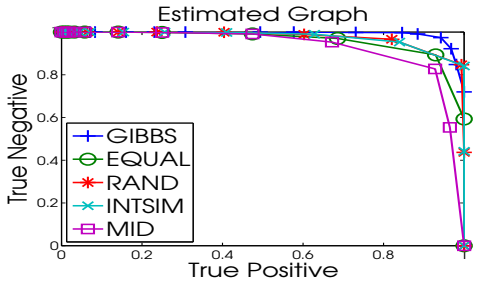



Figure 4.5: ROC curves: True Positive (detected edge) vs. True Negative (detected non-edge) on the Karate graph.

Figures 4.5 shows the ROC curves when varying the threshold. The MID sampling method while being worse than other methods, performs reasonably well. The GIBBS sampling method outperforms other approximate methods while the INTSIM method is slightly better than RAND. This shows that the closer one samples from the true posterior distribution $P(\mathbf{t}, \mathbf{u}|\mathbf{C})$ the more accurate the parameter estimation is. However, one should note that the best performance of GIBBS is achievable at the expense of more intensive computation than the baseline methods.

We also carried out further experiments with real-world networks and data namely the MemeTracker dataset[2] and the American college football network[3] [35]. The interested reader could find more details in the supplement material.

[2]http://www.memetracker.org/
[3]http://networkdata.ics.uci.edu/data.php?id=5

## *4.6 Conclusion*

In this paper, we propose to infer the network influence from the interval censored events of user activity in the social networks. The underlying event triggering mechanism is modeled by the self-exciting multi-dimensional Hawkes process which is able to capture the temporal patterns of user behavior under influence of other users. We propose an imputation approach in which events are sampled under the count constraints and the maximum likelihood estimator is utilized to re-estimate the parameters. We then propose a Gibbs sampling method that could impute timestamps given the count of events. The proposed method is compared to four baseline sampling methods that not only have good intuitions but also reasonably good performance on test data. The experiment results show that the proposed method is able to estimate the influence among nodes when only counts of events in observed intervals are available.

There are several interesting directions for future studies: First, we would like to make the proposed Gibbs sampling method more efficient as it now has to cycle through the timestamps without any parallelization. This is contrary to the MLE solver for Hawkes process which could be sped up tremendously via parallel implementation. Second, we plan to investigate a variational inference approach in which events could be sampled from a simpler distribution to compute a lower bound of the likelihood of the data. Moreover, we can also investigate a problem of estimating the parameters with left censored data (i.e. no information on the first events).

# CHAPTER V

# CLICK-TO-CONVERSION MODELING

## *5.1 Introduction*

Today online advertisement campaigns are generating huge amount of data about customers online shopping behavior. It is the desire of both publishers and and advertisers to leverage this new source of knowledge to evaluate the effectiveness of an advertisement campaign, and also to improve their return. Traditionally, advertisement market mostly based on long-term contract between advertisers (i.e. who need to sell a product or service) and publishers (e.g., newspaper, search engine). Recently, demand-side platforms (DSB) and real-time bidding exchanges (RTBs) gradually become the dominant alternative due to increased liquidity for publishers, and increased audience targeting capabilities for advertisers [11].

There are various pricing options that the publishers offer to the advertisers on the advertisement markets. In this work we primarily works with the cost-per-conversion (CPA) scheme. This pricing scheme allows the advertisers to pay only if the customer takes a specific action on their website after clicking on the advertisement provided by the publishers, i.e. a conversion. This is different from the cost-per-click (CPC) scheme where the advertisers pay for every click. It is therefore critical for the advertisement markets to have reliable estimate of the conversion rate (i.e. the expected number of conversions per click) so that the advertisers could compute a good bid for the advertisement.

The understanding and modeling of customers' behavior is therefore crucial to

judge the effectiveness of an advertisement campaign and to provide reliable prediction on future conversion volume. In this work, we aim at modeling the product adoption process from the customers' first impression with the product (i.e. the customer clicks on an advertisement) to the moment that they convert (i.e. buying/adopting the product). We show that one could model the click-to-conversion mechanism as a thinning process [17] and that there is an efficient inference algorithm to learn this thinning process from given data.

The organization of this chapter is as followings. In the next section, we will introduce our click-to-conversion model using Hawkes processes and thinned processes. In Section 5.3, we proposed an efficient Maximum Likelihood Estimation algorithm. In Section 5.4, we report the experiment on real life data that verify the proposed models. In Section 5.5, we conclude the chapter with some remarks and future works.

### 5.1.1 Terminology

The following terminology are used throughout this chapter.

- *Advertisement*: A set of features consists of advertisement features (e.g. movie, text) and customer features (e.g. age, gender).
- *Impression/Click*: the moment a customer looks at an advertisement and click on it.
- *Conversion*: the moment a customer buy or adopt a product or service.
- *Advertisement campaign*: a set of impressions and conversions.
- *Click-to-Conversion rate*: The expected number of conversion per impression.
- *Click-to-Conversion delay*: The duration from a click to the related conversion.

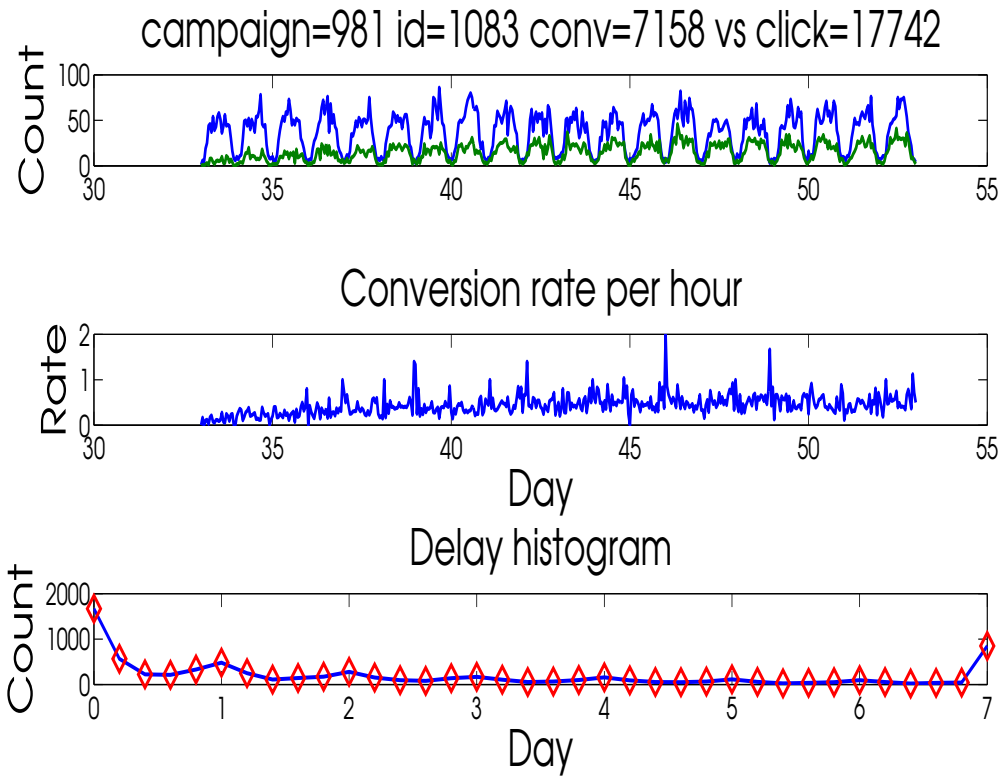


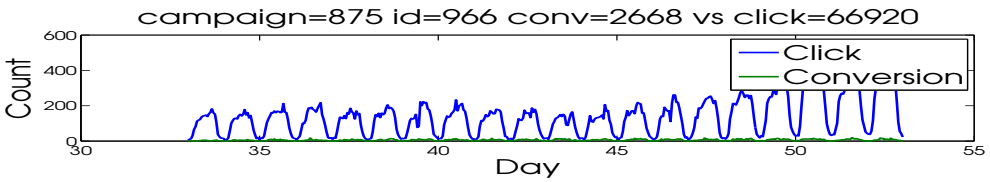

Figure 5.1: Click (blue) and conversion (green) counts per hour, conversion rate per hour, and delay distribution of an advertisement campaign in the Criteo lab data.

## *5.2 Model*

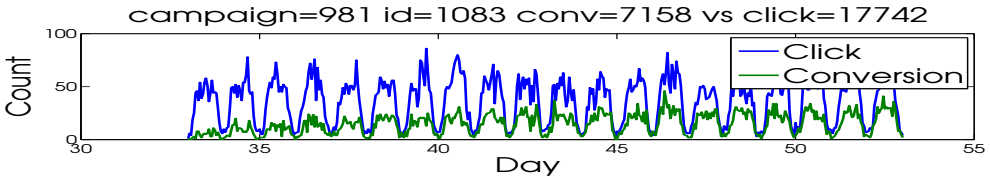
In the following, we demonstrate our proposed model using the Criteo labs data[1]. This data consists of advertisement campaign logs by the Criteo lab in about two months. The timestamps of clicks and related conversions (if any) along with the features of the impressions are recorded. Figure 5.1 shows statistics of a campaign and the click-to-conversion delay distribution of a campaign. Figure 5.2 shows examples of ad campaigns in a selected period (21 days).

From Figure 5.1 and Figure 5.2, we have the following observations. Firstly, we see that Click and Conversion behavior is periodic, there is a similar pattern everyday

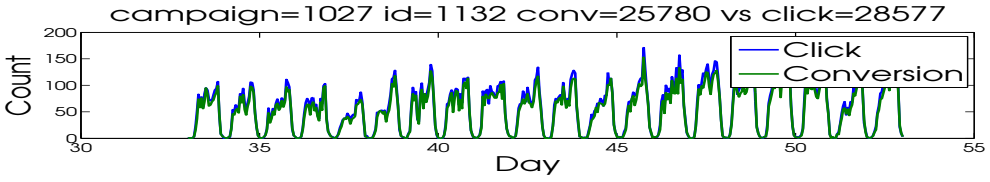
[1]http://labs.criteo.com



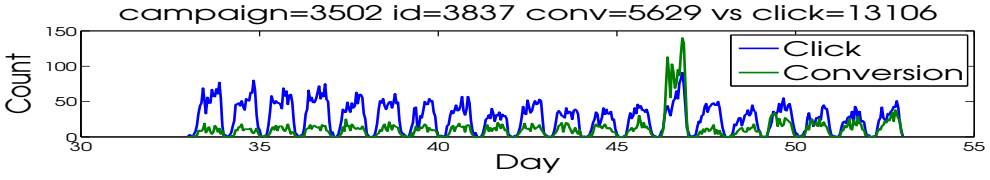

(a) Bad campaign



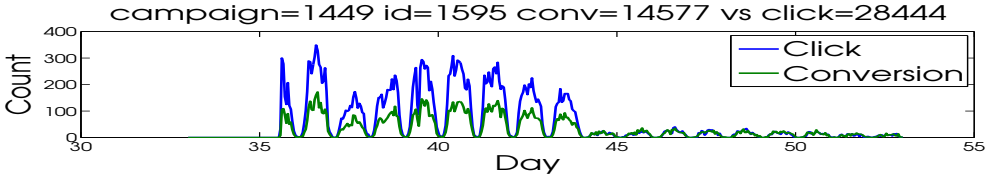

(b) OK campaign



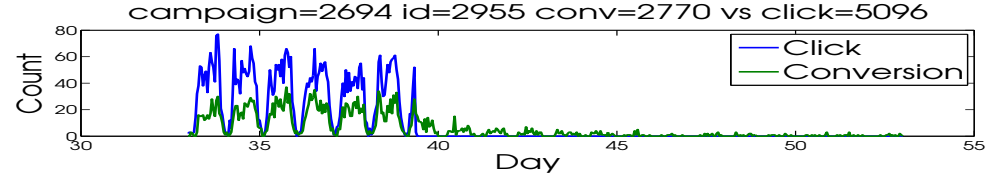

(c) Perfect campaign



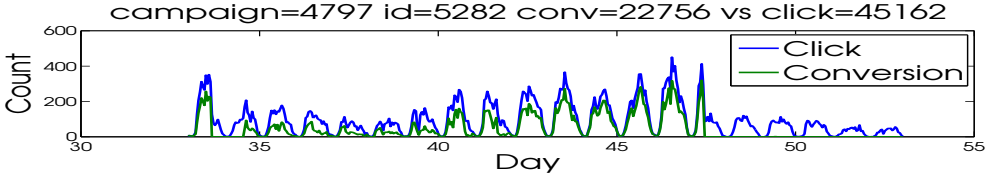

(d) More conversion than clicks in a day




(e) Campaign aftermath




(f) Campaign aftermath




(g) Interest lost

Figure 5.2: Examples of ad campaigns: Number of clicks and number of conversions per hour in 21 days.

(low volume at night, higher volume during the working hours). Secondly, each campaign has a different click-to-conversion rate. There are two possible reasons for this behaviour

1. The first reason is that the features of the ads change from campaign to campaign (i.e. each campaign covers a different feature region in the feature space).

2. The second reason is that the click-to-conversion mapping (or function) changes from campaign to campaign. For example, each campaign has a different weight vector for the features.

Finally, one could see that the click-to-conversion delays mostly concentrate in the first few hours. It is therefore could be approximated by an exponential distribution.

Before we proceed with the detail of our model, we need to introduce some notations that we will use throughout this chapter.

- An impression/click is a pair $(t_n, x_n)$ where $t_n$ is the click time, $x_n$ is the features of the impression (indexed by $n$).

- A click may lead to a conversion. In that case, the conversion time is denoted by $\tau$.

Our main assumption is that one does not know which click is related to a specific conversion. Therefore, the data consists of a set of clicks $\{(t_n, x_n)\}$ and a set of conversions $\{\tau_\ell\}$. This assumption is different from [11] where each conversion is associated with a specific click. Our assumption results from the fact that a customer may click on the advertisement many times before adopting the product.

We propose that one could consider the click timestamps and the conversion timestamps two different temporal point processes. As a click may result in a conversion or not, the conversion process could be considered as a thinned process [17] of the click process. The thinning mechanism is described in the followings.

Inspired by previous works, we use two parametric models

- An impression $(t_n, x_n)$ has click-to-conversion probability $p(x_n)$ that only depends on the features $x_n$. Most previous works used the logistic model

$$p(x_n) = \frac{1}{1 + e^{-\langle w_c, x_n \rangle}}, \tag{5.2.1}$$

  where $w_c$ is a weight vector.

- If converted, the delay from click time $t_n$ to conversion time $\tau$ has distribution $\kappa(\tau - t_n, x_n)$ that depends on the delay $\tau - t_n$ and also the features $x_n$. In [11], the author suggests an exponential distribution with the rate parameter $r(x_n)$ depending on the feature $x_n$ as followings

$$r(x) = e^{\langle w_d, x \rangle}, \kappa(\delta, x) = r(x)e^{-r(x)\delta}. \tag{5.2.2}$$

To restrict the power of the proposed model and avoid over-fitting, one could use regularization on $w_c$ and $w_d$. In this work we use the $\ell_2$ regularization

$$\Omega(w_c, w_d) = \frac{\gamma}{2}(\|w_c\|^2 + \|w_d\|^2).$$

### 5.2.1 Click modeling

Inspired by the Hawkes processes modeling of financial transactions, we propose that one could model the click events using Hawkes processes. Hawkes processes are point processes that model the rate/intensity of new events using the occurrence of past events. This model is based on the assumption that a click increases the chance of future clicks in an advertisement campaign. Specifically, the let $\xi(t)$ be the intensity of click at time $t$ (see Figures 5.3, 5.4, 5.5), we use the model

$$\xi(t) = \xi_0 + \beta \sum_{t_n < t} g(t - t_n). \tag{5.2.3}$$

where $g(t) = \omega e^{-\omega t}$ is the triggering kernel function, $\xi_0$ is the base intensity, and $\beta$ is the coefficient connecting past click events with future click events.

### 5.2.2 Click-to-Conversion modeling

Similarly, let $\lambda(\tau|t)$ be the intensity of conversion at a time $\tau$ given the past click events (not to be confused with click-to-conversion rate $p(x_n)$). We propose that

$$\lambda(\tau|t) = \mu + \alpha \sum_{t_n<\tau} p(x_n)\kappa(\tau - t_n, x_n), \tag{5.2.4}$$

where $\mu$ is the base intensity of conversions and $\alpha$ is the coefficient connecting past click events with future conversion events. This model shows that a click $(t_n, x_n)$ increases the rate of future conversions $\lambda(\tau|t)$ by an amount being equal to $\alpha p(x_n)\kappa(\tau - t_n, x_n)$. This increase depends on the probability of conversion $p(x_n)$ and the delay distribution $\kappa(\tau - t_n, x_n)$.

In summary, we model the click process with a Hawkes process and the conversion process conditional on the click process is a Poisson process resulting from thinning with probability $p(x)$ and click-to-conversion delay distribution $\kappa(\delta, x)$. With this modeling technique, we models the click-to-conversion process in aggregate terms such as rate/intensity or volume of clicks/conversions in a certain time frame. This is different from previous works where the click-to-conversion mechanism is modeled as a (modified) classification problem. In the next section, we will discuss the Maximum Likelihood Estimator (MLE) for this model.

## *5.3 Maximum Likelihood Estimator*

Using the general formula (2.2.3), given a observation time frame $[0, T]$, the full log-likelihood of the data $\{t_n, x_n\}_{n=1}^{N_1}, \{\tau_\ell\}_{\ell=1}^{N_2}$ ($N_1$ clicks, $N_2$ conversions) is

$$\begin{aligned} L(t, \tau) &= \ln\left(P(t) \times P(\tau|t)\right) \\ &= \sum_{n=1}^{N_1} \ln \xi(t_n) - \int_0^T \xi(t)dt + \sum_{\ell=1}^{N_2} \ln \lambda(\tau_\ell|t) - \int_0^T \lambda(\tau|t)d\tau, \\ &= L_{click} + L_{conversion}. \end{aligned} \tag{5.3.1}$$

where

$$\xi(t_n) = \xi_0 + \beta \sum_{t_k < t_n} g(t_n - t_k),$$
$$\int_0^T \xi(t)dt = \xi_0 T + \beta \sum_{n=1}^{N_1} G(T - t_n), \tag{5.3.2}$$
$$\lambda(\tau_\ell) = \mu + \alpha \sum_{t_n < \tau_\ell} p(x_n)\kappa(\tau_\ell - t_n, x_n),$$
$$\int_0^T \lambda(\tau)d\tau = \mu T + \alpha \sum_{n=1}^{N_1} p(x_n)K(T - t_n, x_n), \tag{5.3.3}$$

with $K(\delta, x) = \int_0^\delta \kappa(t, x)dt$ and $G(\delta) = \int_0^\delta g(t)dt$.

### 5.3.1 Click MLE

Maximizing $L_{click}$ with respect to $\xi_0, \beta, \omega$ is the standard Hawkes MLE. One could use Algorithm 2.3 restricted to 1-dimensional Hawkes to estimate these parameters.

### 5.3.2 Conversion MLE

For maximizing $L_{conversion}$, we apply the Minorization-Maximization framework (see Algorithm 2.2) where one derives a tight lower-bound of the objective function (i.e. with possible equality) and maximizes this lower-bound. We use the following inequality

$$\lambda(\tau_\ell) \geq \eta_\ell^0 \ln \frac{\mu}{\eta_\ell^0} + \sum_{t_n < \tau_\ell} \eta_\ell^n \ln \frac{\alpha p(x_n)\kappa(\tau_\ell - t_n, x_n)}{\eta_\ell^n}$$

where the positive weights $\eta_\ell^n$'s satisfy $\eta_\ell^0 + \sum_{t_n < \tau_\ell} \eta_\ell^n = 1$. The equality holds when

$$\begin{aligned} \eta_\ell^0 &= \frac{\mu}{\mu + \sum_{t_n < \tau_\ell} \alpha p(x_n)\kappa(\tau_\ell - t_n, x_n)}, \\ \eta_\ell^n &= \frac{\alpha p(x_n)\kappa(\tau_\ell - t_n, x_n)}{\mu + \sum_{t_{n'} < \tau_\ell} \alpha p(x_{n'})\kappa(\tau_\ell - t_{n'}, x_{n'})}. \end{aligned} \tag{5.3.4}$$

The lower-bound above leads to the following updates for the conversion parameters

- Update $\mu$[2]

$$\mu = \frac{\sum_{\ell=1}^{N_2} \eta_\ell^0}{T}. \tag{5.3.5}$$

- Update $\alpha$:

$$\alpha = \frac{\sum_{\ell=1}^{N_2} \sum_{t_n<\tau_\ell} \eta_\ell^n}{\sum_{n=1}^{N_1} p(x_n) K(T-t_n, x_n)}. \tag{5.3.6}$$

- Update $p(x)$, using the logistic model $p(x) = \dfrac{1}{1+e^{-\langle w_c, x\rangle}}$, the optimization problem is

$$\min_{w_c} F \triangleq -\sum_{\ell=1}^{N_2} \sum_{t_n<\tau_\ell} \eta_\ell^n \ln p(x_n) + \sum_{n=1}^{N_1} p(x_n) K(T-t_n, x_n) + \frac{\gamma}{2}\|w_c\|^2. \tag{5.3.7}$$

From the gradients $\nabla_{w_c} p = p(1-p)x$ and $\nabla_{w_c} \ln p = (1-p)x$, we have the gradient

$$\nabla_{w_c} F = -\sum_{\ell=1}^{N_2} \sum_{t_n<\tau_\ell} \eta_\ell^n [1-p(x_n)]x_n + \sum_{n=1}^{N_1} p(x_n)[1-p(x_n)]K(T-t_n, x_n)x_n + \gamma w_c. \tag{5.3.8}$$

- Update $\kappa(\delta, x)$, using the exponential model

$$r(x) = e^{\langle w_d, x\rangle}, \kappa(\delta, x) = r(x)e^{-r(x)\delta}, K(\delta, x) = 1 - e^{-r(x)\delta},$$

the optimization problem is

$$\begin{aligned} \min_{w_d} G \triangleq & -\sum_{\ell=1}^{N_2} \sum_{t_n<\tau_\ell} \eta_\ell^n \left[\ln r(x_n) - r(x_n)(\tau_\ell - t_n)\right] \\ & + \alpha \sum_{n=1}^{N_1} p(x_n) \left(1 - e^{-r(x_n)(T-t_n)}\right) + \frac{\gamma}{2}\|w_d\|^2. \end{aligned} \tag{5.3.9}$$

From the gradients $\nabla_{w_d} r = rx$ and $\nabla_{w_d} \ln r = x$, we have the gradient

$$\begin{aligned} \nabla_{w_d} G = & -\sum_{\ell=1}^{N_2} \sum_{t_n<\tau_\ell} \eta_\ell^n \left[1 - r(x_n)(\tau_\ell - t_n)\right] x_n \\ & + \alpha \sum_{n=1}^{N_1} p(x_n) e^{-r(x_n)(T-t_n)} r(x_n)(T-t_n)x_n + \gamma w_d. \end{aligned} \tag{5.3.10}$$

[2]The optimization problem $\max_{x\geq 0} a\ln x - bx$ has maximum at $x = a/b$.

**Algorithm 5.1** Click-to-Conversion MLE

**Require:** $\{t_n, x_n\}_{n=1}^{N_1}, \{\tau_\ell\}_{\ell=1}^{N_2}$
**Ensure:** $\xi_0, \beta, \omega$ maximize $L_{click}$ and $\mu, \alpha, w_c, w_d$ maximize $L_{conversion}$.
  Solve max $L_{click}$ with standard Hawkes MLE for $\xi_0, \beta, \omega$ (Algorithm 2.3).
  Initialize $\mu, \alpha, w_c, w_d$.
  **while** change in $L_{conversion}$ is more than tolerance **do**
    Compute $\eta_\ell^n$'s with Eq. (5.3.4).
    Compute $\mu, \alpha$ with Eq. (5.3.5) and Eq. (5.3.6).
    Optimize $w_c, w_d$ using gradients from Eq. (5.3.8) and Eq. (5.3.10) (L-BFGS algorithm).
  **end while**

### 5.3.3 Efficient MLE algorithm

The disscussion above leads to a MLE algorithm which is summarized in Algorithm 5.1. The efficient implementation of Algorithm 5.1 requires book-keeping of the following quantities

$$A = \sum_{\ell=1}^{N_2} \eta_\ell^0,$$

$$B(n) = \sum_{\ell:\tau_\ell > t_n} \eta_\ell^n,$$

$$C(n) = \sum_{\ell:\tau_\ell > t_n} \eta_\ell^n (\tau_\ell - t_n).$$

These quantities could be accumulated right after each $\eta_\ell^n$ is available. One does not need to store $\eta_\ell^n$ after it has been accumulated into $A, B(n)$, and $C(n)$. This book-keeping reduces memory requirement from quadratic $O(N_1 \times N_2)$ to linear $O(N_1 + N_2)$. All other quantities could be computed using $A, B(n)$, and $C(n)$ and the model

parameters. The updates become

$$\mu = \frac{A}{T},$$
$$\alpha = \frac{\sum_{n=1}^{N_1} B(n)}{\sum_{n=1}^{N_1} p(x_n)K(T-t_n, x_n)},$$
$$\nabla_{w_c} F = -\sum_{n=1}^{N_1} B(n)[1-p(x_n)]x_n + \sum_{n=1}^{N_1} p(x_n)[1-p(x_n)]K(T-t_n, x_n)x_n + \gamma w_c,$$
$$\nabla_{w_d} G = -\sum_{n=1}^{N_1} [B(n) - r(x_n)C(n)]x_n + \alpha \sum_{n=1}^{N_1} p(x_n)e^{-r(x_n)(T-t_n)} r(x_n)(T-t_n)x_n + \gamma w_d.$$

Therefore one could carry out these computations efficiently in $O(N_1)$ time.

## *5.4 Experiments*

### 5.4.1 Data preparation

We use the Criteo lab's data in these experiments. The data consists of the timestamps (in seconds) of the clicks and the timestamps of the associated conversions (if any). While the data associate every conversion with a click, we do not use this information in our experiments. Instead, we consider the data as two separated sets, namely, the set of clicks and the set of conversions.

Every click/impression is associated with a feature vector that has 8 numeric features and 9 categorical features. Following suggestion in [12], we adopt the hashing trick to take care of the categorical features and also the *missing values*. The number of bins that we used is $2^{22} \sim 4$ millions. That is, the feature vector $x_n$ used in our algorithm has about 4 million dimensions.

In total the data consists of 15 million clicks of 13000 advertisement campaigns over the period of about 2 months. In the following experiments, we only works with campaigns that have overall conversion rate at least 0.1 percent and at least 1000 clicks.

### 5.4.2 Predicting click and conversion volume

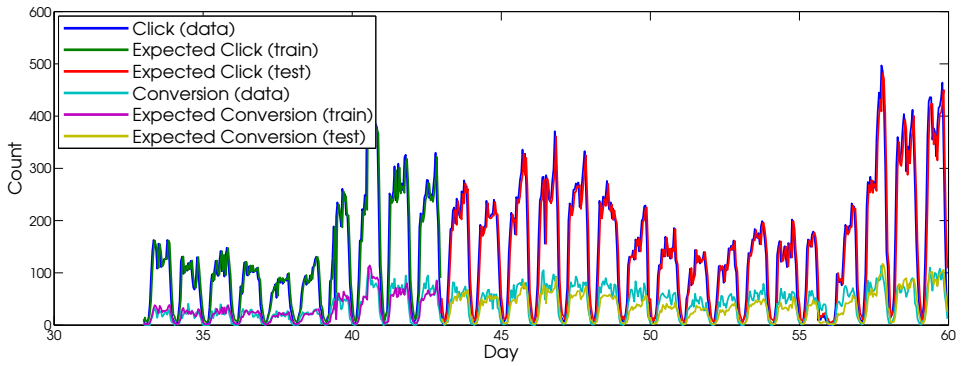
The first set of experiments focuses on how well the proposed model approximates real life data. We test the model with campaigns with high conversion rate, medium conversion rate, and low conversion rate. We first select the log data from a set of consecutive days as the training set and run Algorithm 5.1. Then we compute the expected counts of events in *each hour* of the training and testing period (a set of days after the training period). The expected counts in an interval $[0, T)$ are given by Eq. (5.3.2) and Eq. (5.3.3). To compute the count in an arbitrary interval $[T_1, T_2)$, one only needs to substract the corresponding integrals.

Figure 5.3 shows the promising modeling result of our proposed model. We computed the true counts of clicks and conversions per hour from the data and compare them with the expected counts that the model predicts in the *next hour* (i.e. the look-ahead interval is 1 hour). In all three cases (i.e. different conversion rate), the model approximates the true click and conversion counts very well.

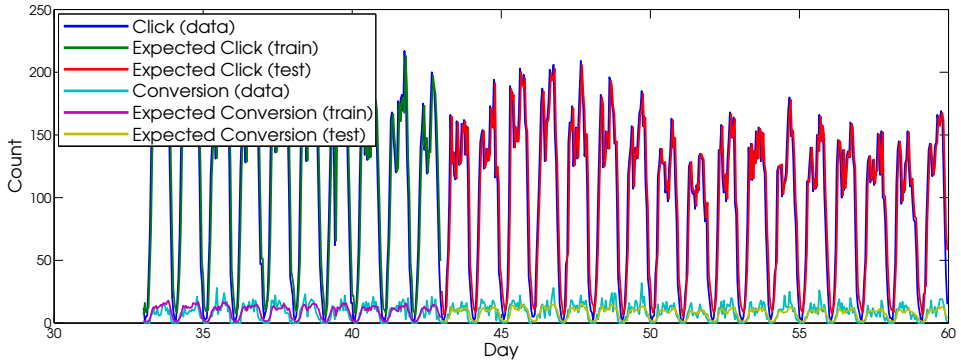
We then increase the look-ahead interval to 24 hours (Figure 5.4) and 72 hours (Figure 5.5) to see how well the model predicts future prediction volumes. As expected, the predicted click and conversion volume are less accurate but are still reasonable. This could be explained by the exponential distribution of the delays. The nature of exponential distribution requires that one needs more recent data for more accurate prediction. For campaigns with medium conversion rate and low conversion rate, one could see the prediction is very close to that with 1-hour look ahead.

### 5.4.3 Conversion prediction

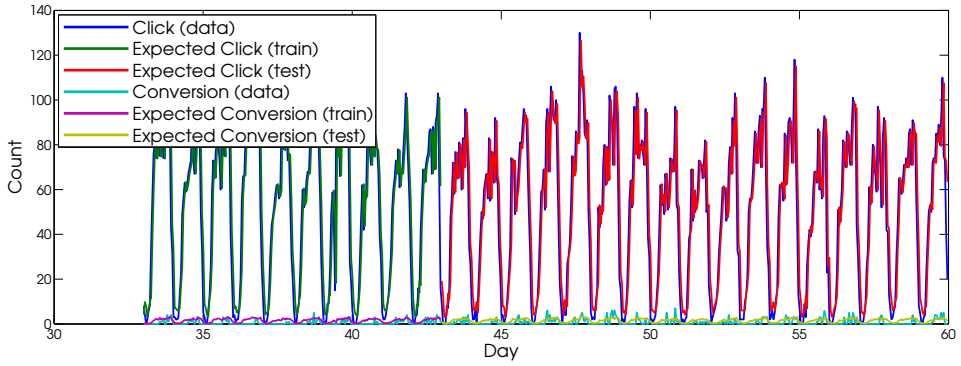
In this set of experiments, we are concerned with the logistic model $p(x)$ that models the conversion probability of the click with feature vector $x$. As in the Criteo data a click either has an associated conversion or not, we could compute the *negative log-likelihood* of the model $p(x)$ for a set of samples $\{(x_n, y_n)\}$ where $y_n = 1$ if $x_n$ is




(a) Campaign with conversion rate is 0.28: 17 test days.

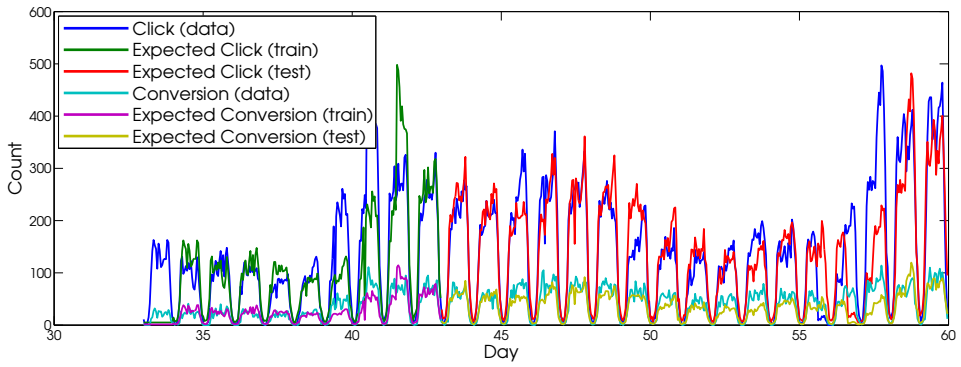



(b) Another campaign with conversion rate is 0.08.

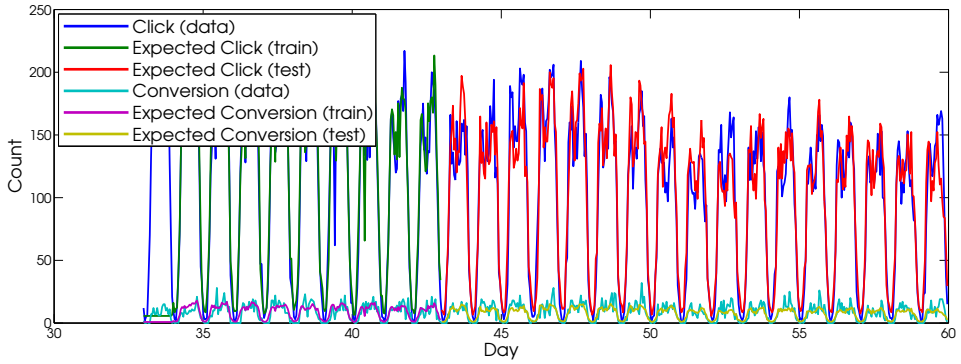



(c) Another campaign with conversion rate is 0.03.

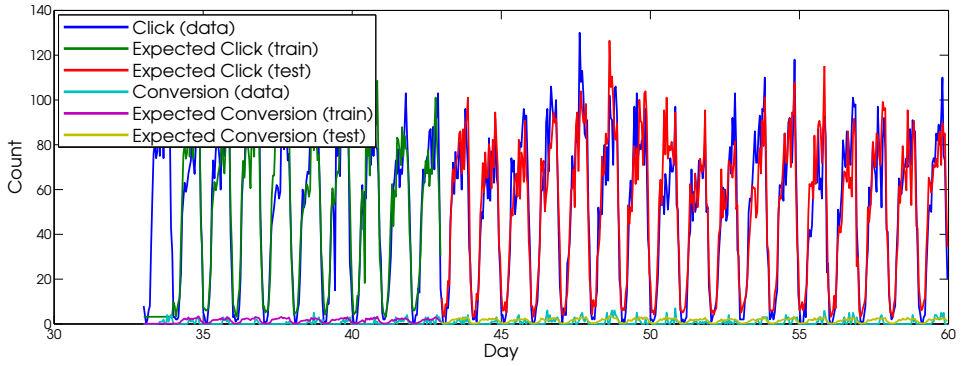
Figure 5.3: 1 hour look-ahead: Number of clicks and number of conversions per hour, real counts and expected counts computed from the MLE models in training period and testing period.




(a) Campaign with conversion rate is 0.28: 17 test days.

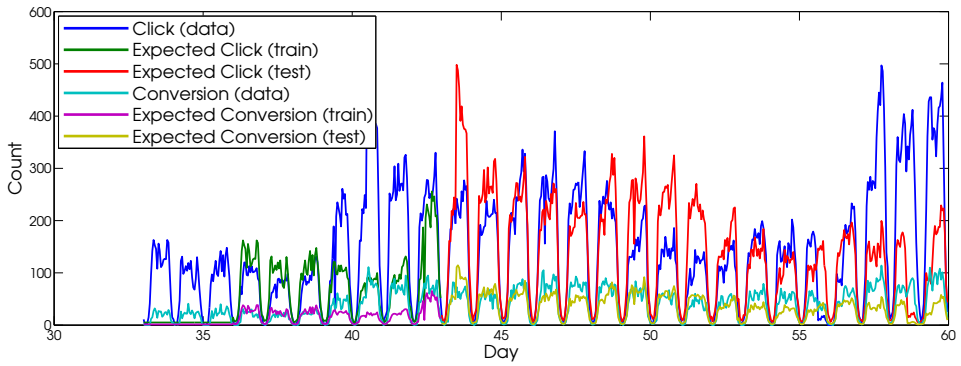



(b) Another campaign with conversion rate is 0.08.

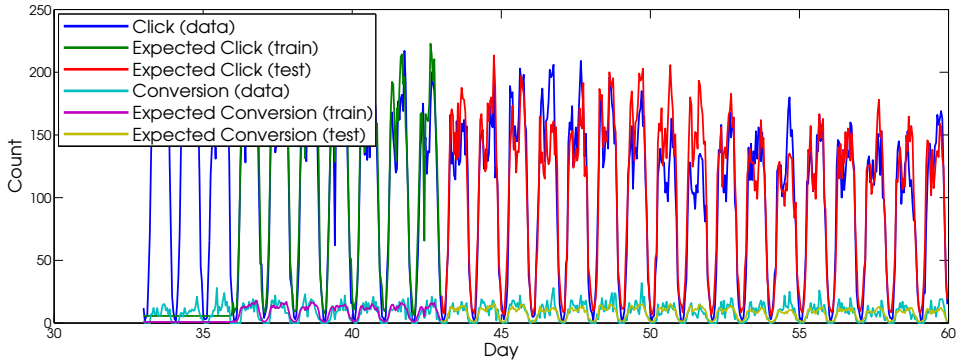



(c) Another campaign with conversion rate is 0.03.

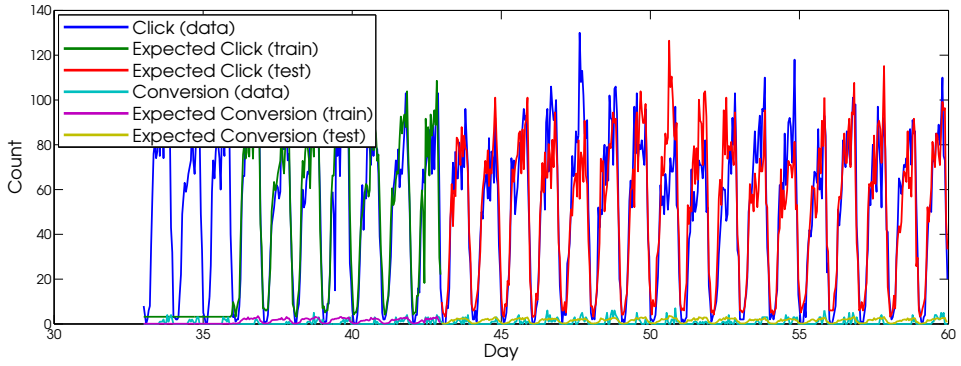
Figure 5.4: 1 day look-ahead: Number of clicks and number of conversions per hour, real counts and expected counts computed from the MLE models in training period and testing period.




(a) Campaign with conversion rate is 0.28: 17 test days.




(b) Another campaign with conversion rate is 0.08.




(c) Another campaign with conversion rate is 0.03.

Figure 5.5: 3 day look-ahead: Number of clicks and number of conversions per hour, real counts and expected counts computed from the MLE models in training period and testing period.

converted and $y_n = 0$, otherwise. The formula for the negative log-likelihood is

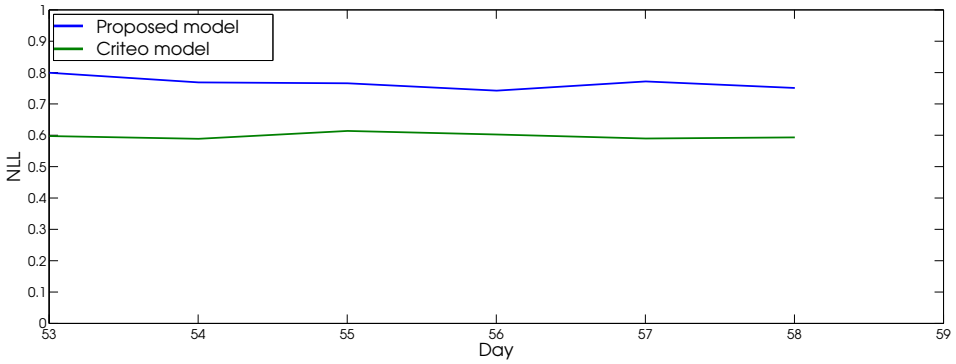
$$L_p = -\frac{1}{N_1}\sum_{n=1}^{N_1} y_n \ln p(x_n) + (1 - y_n)\ln(1 - p(x_n)). \tag{5.4.1}$$

The smaller $L_p$ is, the better the conversion model $p(x)$ predicts whether a click results in a conversion. We follow the experiment settings in [11] where the test days are days in the last week of the data and models are trained on the previous 3 weeks of the test days. Eq. (5.4.1) is used to compute the negative log-likelihood (NLL) on the test days. We denote the model in [11] by Criteo model.

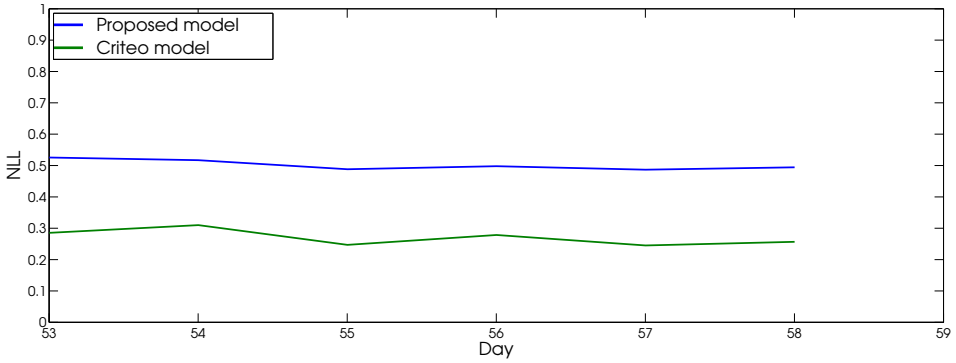
Figure 5.6 reports the NLL of three campaigns. It could be seen that our proposed model performs worse than the Criteo model in all cases. This is expected because of the fact that our proposed model tries to approximate the counts instead of the labels (i.e. conversion or no conversion) of the clicks. Meanwhile, the Criteo model utilizes the association between the click and the conversion in order to find the weight $w_c$ for the model $p(x)$. Therefore, the Criteo model is expected to be more accurate in term of prediction for each click.

This experiment shows a drawback in our approach that it only approximates that aggregate volume (or equivalently, the rate) of conversions. This results from the fact that we do not take into account the association between clicks and conversions available in the Criteo lab data. The reader should note that without this association, the Criteo model is not able to carry out any estimation. Our model is still able to come up with a set of model pameters that approximates the observed conversion rate without the association between clicks and conversions.

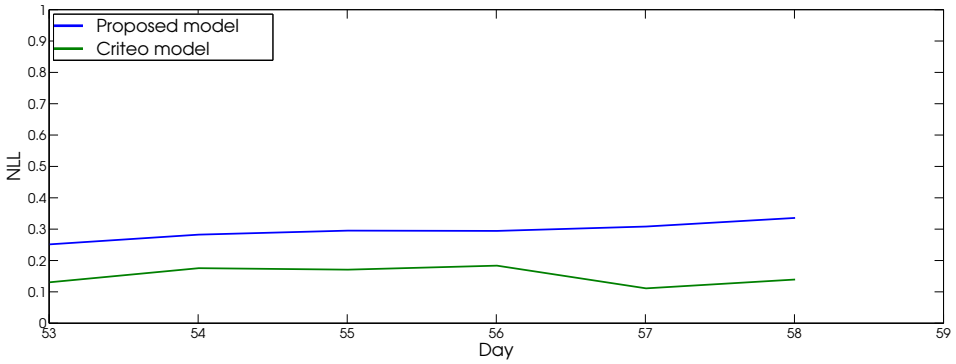
A possible fix to our model could be the semi-supervised approach where for a subset of the conversions, one know the associated clicks to this subset. While for the remaining conversions, one do not know the associated clicks. The resulting model is then a hybrid model between our thinned process model and other previously proposed classification model.




(a) Campaign with conversion rate is 0.28: 17 test days.




(b) Another campaign with conversion rate is 0.08.




(c) Another campaign with conversion rate is 0.03.

Figure 5.6: Negative log-likelihood on test day: The models are trained on 21 days and tested on the next day.

## *5.5 Conclusion*

In this chapter, we proposed a Click-to-Conversion model based on the thinned processes framework. The click process is represented by a Hawkes process under the assumption that customer often clicks many times on an advertisement. The conversion process is represented by a thinned process of the click process with a logistic model specifying the thinning probability and an exponential distribution for the click-to-conversion delay. We then derive an efficient Maximum Likelihood Estimation algorithm to estimate the model paramters.

The experiments show that the proposed model has good predicting capability with respect to future click and conversion volume. In the near future, we would like to enhance the model by considering the influence of previous conversions on future conversions. Besides, we also would like to introduce into the model the association between clicks and conversions so that it could have better prediction power on each click.

# CHAPTER VI

# DISTRIBUTED CONCENSUS OPTIMIZATION

## *6.1 Introduction*

Data-distributed learning is an important problem that arises in many real-world applications. For example, in many large-scale machine learning systems, data samples are distributed over hundreds or thousands of general purpose servers. Locally accessing data is typically faster than the remote access due to the latency of network communication and limited bandwidth. The same problem can happen in wireless sensor networks where the data is collected locally by each sensor node and the resource constraints preclude any learning algorithm that demands high volumes of inter-sensor communications. In both these realistic scenarios, there is no pragmatic or desirable way to move data to a central node or move large amount of data between nodes. Despite long-standing efforts to federate data in various ways, in reality for large-scale problems, data will always be distributed for various reasons.

We formulate the distributed learning problem as a consensus constrained optimization problem and solve it using the general methodology of Alternating Direction Method of Multipliers (ADMM) [34]. As surveyed in the monograph [6], ADMM is a flexible algorithmic framework for solving constrained problems. Its unique characteristic of "separability" can be utilized to explore various structures of the learning problems. For our distributed consensus learning problem, the main structure of concern is the underlying communication topology, which can be easily modeled as equality constraints in ADMM. Topology is one of the most critical issues in implementing consensus learning for two reasons: First, different topologies might lead to different iteration complexities for the algorithms. Second, the distribution and

number of edges in the communication graph will result in different communication overloads. A practical system should always make a proper balance between these factors.

One of the central themes in distributed learning is the question "What is the *best* communication topology?" To reach a definitive answer to this question, one still needs to overcome major hurdles because the convergence behavior of ADMM in this context not only depends on the communication topology, but also on the penalty parameter $\beta$ used in the augmented Lagrangian. The main focus of this paper is to characterize the interplay between these factors, and to this end we present a new convergence analysis for ADMM with Lipschitz smooth and strongly convex functions (Section 6.4). Based on the derived convergence rates, we design an adaptive scheme to choose $\beta$ (Section 6.5). In Section 6.6 we use several sets of numerical examples to show: a) to what extent does $\beta$ affect the convergence rates; b) given the "optimal" $\beta$, which topology achieves faster convergence rates; c) the effectiveness of the proposed adaptive $\beta$ strategy; and d) a practical selection for $\beta$ for simple ADMM cases.

### 6.1.1 Related Work

There are generally two classes of methods for the distributed learning in the literature. The first class includes the gradient-based primal methods: e.g. the distributed subgradient descent methods [64, 19] and the distributed dual averaging methods [25, 1, 26]. The second class are primal-dual methods based on the augmented Lagrangian method [88] or ADMM [6, 60, 63]. In gradient-based methods, the (sub)gradients are transmitted and aggregated in the hope that all workers will asymptotically obtain information from all data samples. While for the second class, the consensus requirements are *explicitly* encoded as constraints, and all data samples are kept local. The starting point for our work is the D-ADMM algorithm [63] which belongs to the second class. However in this paper we focus on the convergence

Figure 6.1: Left: Two ways to formulate bipartite graphs. Right: Consensus constraints expressed in matrix form.

behavior of the algorithm and we want to investigate how it will be affected by the various factors of our problem.

## *6.2 Problem Settings and Notations*

We are interested in the following distributed consensus learning problem:

$$\min f(\mathbf{x}) \equiv \sum_{i=1}^{N} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_N, \tag{6.2.1}$$

where $\mathbf{x}_i \in \mathbf{R}^D$ and each worker $i$ is associated with an individual function $f_i(\mathbf{x}_i)$ and its corresponding subset of data. The $N$ distributed workers are connected via a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of $N$ indexed vertices and $\mathcal{E}$ is the set of edges of the network. Each vertex $v_i$ is associated with a local variable $\mathbf{x}_i$. Information can be transferred between $v_i$ and $v_j$ in either direction as long as they are connected by edge $e_{ij}$. Note that despite the connectivity via $e_{ij}$, $v_i$ and $v_j$ have the freedom to choose whether they want to exchange information or not. In other words, $\mathcal{G}$ only reflects the connectivity, but not communications.

We propose to solve problem (6.2.1) by ADMM in parallel. To take advantage of ADMM's capacities in dealing with separable functions, we have at least the following two structural options, as illustrated in Fig.6.1 (Left), where we use a case with 24 workers as an example.

1. **Centralized Learning.** We use axillary global (central) variables $\mathbf{z} \equiv \{\mathbf{z}_j\}$ such that every $\mathbf{x}_i$ are connected to some $\mathbf{z}_j$. In this way we can reproduce

equivalent connectivities represented by the original graph $\mathcal{G}$. When $|\mathbf{z}| = 1$, this is called master-slave consensus optimization, where the global variable $\mathbf{z}$ is hosted by the master, and all $\mathbf{x}_i$ variables are updated by the slaves. When $|\mathbf{z}| > 1$, the paradigm is called general form consensus optimization [6].

2. **Decentralized Learning.** Global variables are not necessary in this paradigm, hence there is no master node. The $N$ local functions $f_i$ are simply divided into groups, where communication only happens between different groups, but not within each group. For simplicity, we divide them into 2 groups. Following the work of [63] we design a *bipartite graph* for communication. The bipartite graph provides a model that is general enough for many practical distributed learning problems, and fast convergence is guarenteed. In comparison, only very few results are applicable to a general decentralized communication topology, e.g. those with sublinear rates [64, 25].

In this work we focus on the second paradigm since the centralized learning can be regarded as a special case of the decentralized learning where the master nodes do not have their own data samples.

Both the above two distributed learning paradigms can be conveniently formulated as the following problem that can be solved by ADMM:

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} \theta_1(\mathbf{x}) + \theta_2(\mathbf{y}), \quad \text{s.t. } A\mathbf{x} + B\mathbf{y} = \mathbf{b}, \tag{6.2.2}$$

where $\theta_1$ and $\theta_2$ are convex functions, $\mathcal{X}$ and $\mathcal{Y}$ are closed convex sets. In this paper, instead of using the classic ADMM [6], we follow the scheme of generalized ADMM (Alg.6.1) as discussed in [42]. The only difference is the additional term for the proximity function $\frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|_G^2$, where the $G-$norm is defined as $\|\mathbf{x}\|_G = \sqrt{\mathbf{x}^T G\mathbf{x}}$. Variations of ADMM can be derived from different $G$, e.g. the linearized ADMM [37, 84]. We use $\|\cdot\|$ to denote the $l_2$ norm. The *augmented Lagrangian* in Alg.6.1 is defined as: $\mathcal{L}_\beta(\mathbf{x}, \mathbf{y}, \lambda) \equiv \theta_1(\mathbf{x}) + \theta_2(\mathbf{y}) - \langle \boldsymbol{\lambda}, A\mathbf{x} + B\mathbf{y} - \mathbf{b}\rangle + \frac{\beta}{2}\|A\mathbf{x} + B\mathbf{y} - \mathbf{b}\|^2$,

**Algorithm 6.1** Generalized ADMM ($G \succeq 0$)

[0.] Initialize $\mathbf{y}^0$ and $\boldsymbol{\lambda}^0$.
**for** $k = 0, 1, 2, \ldots$ **do**
[1.] $\mathbf{x}^{k+1} \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}} \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}^k, \boldsymbol{\lambda}^k) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|_G^2$.
[2.] $\mathbf{y}^{k+1} \leftarrow \arg\min_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}_\beta(\mathbf{x}^{k+1}, \mathbf{y}, \boldsymbol{\lambda}^k)$.
[3.] $\boldsymbol{\lambda}^{k+1} \leftarrow \boldsymbol{\lambda}^k - \beta\left(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}\right)$.
**end for**

where $\beta$ is a pre-defined penalty parameter that is crucial in achieving faster rates of convergence. We make the following assumptions for the rest of this paper.

**Assumption 1.** Functions $\theta_1$ and $\theta_2$ are $L_1$ and $L_2$ Lipschitz smooth, and are $\mu_1$ and $\mu_2$ strongly convex.

## *6.3 Distributed Consensus Learning*

As discussed in Section 6.2, we are interested in the decentralized learning paradigm where the $N$ workers constitute a bipartite graph $\mathcal{B} \equiv \{\mathcal{V}_L, \mathcal{V}_R, \mathcal{C}\}$ with left part $\mathcal{V}_L$ and right part $\mathcal{V}_R$. The communication edge set $\mathcal{C} \subseteq \mathcal{E}$ represents the communication scheme: if there is an edge $c_{pn}$, then worker $v_p$ and $v_n$ will exchange information in each iteration of ADMM. Note that even if $v_p$ and $v_n$ is connected by the network edge $e_{pn} \in \mathcal{E}$, no communication will be carried out if they are not connected by $c_{pn}$.

The distributed consensus learning can thus be formulated as an optimization problem with $|\mathcal{C}|$ equality constraints $\{\mathbf{x}_p = \mathbf{y}_n : \forall c_{np} \in \mathcal{C}\}$. Writing these constraints in ADMM's matrix form $A\mathbf{x} + B\mathbf{y} = \mathbf{0}$, we can see that $A \in \mathbb{R}^{D|\mathcal{C}|\times D|\mathcal{V}_L|}$ is a matrix of $|\mathcal{C}|$ block-rows, with each block row containing only one identity matrix $I$ and 0 for others. Matrix $B$ is defined similarly, with each block-row containing only one $-I$. The positions of $I$ and $-I$ in each block-row of $A$ and $B$ indicates the consensus between two specific workers. An example is illustrated in Fig.6.1 (Right). Since there are $|\mathcal{C}|$ consensus constraints, we introduce $|\mathcal{C}|$ Lagrangian multipliers $\boldsymbol{\lambda}_{pn}$ for each edge $c_{pn}$. The ADMM based distributed consensus learning is given in Alg.6.2,

where the augmented Lagrangians are

$$\mathcal{L}_\beta(\mathbf{x}_i, \mathbf{y}^k, \boldsymbol{\lambda}^k) = f_i(\mathbf{x}_i) - \sum_{n=1}^{\mathcal{N}_i} \langle \boldsymbol{\lambda}_{in}^k, \mathbf{x}_i \rangle + \frac{\beta}{2} \sum_{n=1}^{\mathcal{N}_i} \|\mathbf{x}_i - \mathbf{y}_n^k\|^2, \text{and } \mathcal{L}_\beta(\mathbf{x}^{k+1}, \mathbf{y}_i, \boldsymbol{\lambda}^k) = f_i(\mathbf{y}_i) + \sum_{p=1}^{\mathcal{P}_i} \langle \boldsymbol{\lambda}_{pi}^k, \mathbf{y}_i \rangle + \frac{\beta}{2} \sum_{p=1}^{\mathcal{P}_i} \|\mathbf{x}_p^{k+1} - \mathbf{y}_i\|^2. \tag{6.3.1}$$

Here $\mathcal{N}_i$ represents the number of right workers (in $\mathcal{V}_R$) connected to the left worker $i$, and $\mathcal{P}_i$ represents the number of left workers (in $\mathcal{V}_L$) connected to the right worker $i$.

In Alg.6.2, all $\mathbf{x}_i$ are updated in parallel by the left workers, followed by the parallel updates of $\mathbf{y}_i$ by the right workers. In practice, all the updates of $\boldsymbol{\lambda}$ are computed in parallel by the right workers, since they have access to the latest copies of $\mathbf{y}^{k+1}$ and $\mathbf{x}^{k+1}$ in each iteration $k$, while the left workers only have $\mathbf{x}^{k+1}$ and the old copy of $\mathbf{y}^k$.

**Algorithm 6.2** Distributed Consensus Learning

[0.] Initialize $\mathbf{y}^0$ and $\boldsymbol{\lambda}^0$.
**for** $k = 0, 1, 2, \ldots$ **do**
  [1.] $\forall i$ (parallel) $\mathbf{x}_i^{k+1} \leftarrow \arg\min_{\mathbf{x}_i} \mathcal{L}_\beta(\mathbf{x}_i, \mathbf{y}^k, \boldsymbol{\lambda}^k)$.
  [2.] $\forall i$ (parallel) $\mathbf{y}_i^{k+1} \leftarrow \arg\min_{\mathbf{y}_i} \mathcal{L}_\beta(\mathbf{x}^{k+1}, \mathbf{y}_i, \boldsymbol{\lambda}^k)$.
  [3.] $\forall p, n$ (parallel) $\boldsymbol{\lambda}_{pn}^{k+1} \leftarrow \boldsymbol{\lambda}_{pn}^k - \beta \left(\mathbf{x}_p^{k+1} - \mathbf{y}_n^{k+1}\right)$.
**end for**

### 6.3.1 Three Dimensions of the Problem Space

Taking a closer look at Alg.6.2 we can find that there are actually three factors for the implementation of this algorithm. Firstly, we can choose any communication topology that is encoded in matrices $A$ and $B$. Secondly, the penalty parameter $\beta$ can be any positive number. Thirdly, it is free to change the updating order for $\mathbf{x}$ and $\mathbf{y}$ (the update of $\boldsymbol{\lambda}$ should also be modified accordingly). In order to investigate the interactions among these factors, we use both theoretical analysis (Section 6.4, 6.5) and numerical examples (Section 6.6) to study the convergence of Alg.6.2.

## 6.4 Iteration Complexities of ADMM

The theory of ADMM remains a hard open problem for decades. The global convergence of ADMM was established in the literature [33, 36, 28]. The $O(1/k)$ convergence rate was established by [43, 42, 67] where the authors only assume that $\theta_1$ and $\theta_2$ are convex. When these functions are both Lipschitz smooth and strongly convex, preliminary results on linear convergence are reported very recently. In [44], the authors derived $R$-linear rates for the sum of primal and dual gaps for a setting that is more general than (6.2.2). However, the constants in the bound are not directly applicable to our setting. In [20], the authors present linear rates only for the case when $G = 0$, and as a consequence no rate is given for $\mathbf{x}$. In the following we present explicit formulas of linear rates for all the primal variables $\mathbf{x}$, $\mathbf{y}$ and dual variable $\boldsymbol{\lambda}$.

The following key lemma states that $\|\mathbf{w}^k - \mathbf{w}^*\|_M$ is monotonically non-increasing, and the reduction of $\mathbf{w}^k - \mathbf{w}^*$ is faster than $\mathbf{w}^k - \mathbf{w}^{k+1}$. Variations of this lemma have been presented several times in the literature under different settings and assumptions [41, 6, 42, 20]. Our result is more general in the sense that this lemma is applicable to convex feasible sets $\mathcal{X}$ and $\mathcal{Y}$, not just $\mathbb{R}^x$ and $\mathbb{R}^y$. The proof is fairly simple and only relies on the optimality conditions.

**Lemma 1.** Under Assumption 1 we have

$$\|\mathbf{w}^k - \mathbf{w}^*\|_M^2 - \|\mathbf{w}^{k+1} - \mathbf{w}^*\|_M^2 \geq \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_M^2 + 2\mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2, \tag{6.4.1}$$

where $\mathbf{w}^k \equiv (\mathbf{x}^k, \mathbf{y}^k, \boldsymbol{\lambda}^k)^T$, $\mathbf{w}^* \equiv (\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*)^T$ is the optimal solution of (6.2.2), and $M \equiv \text{Diag}\left(G,\ \beta B^T B,\ \frac{I}{\beta}\right)$.

**Remark 1.** For the general convex cases, i.e. $\mu_1 = \mu_2 = 0$, the $O(1/k)$ convergence rate of ADMM can be easily derived from Lemma 1 [42].

### 6.4.1 Linear Convergence Rates

For strongly convex ($\mu_1, \mu_2 > 0$) and Lipschitz smooth functions, linear convergence rates can also be obtained from Lemma 1. Note that all the results in this section rely on the assumption that $\mathcal{X} = \mathbb{R}^x$ and $\mathcal{Y} = \mathbb{R}^y$. In the following results we use $\Lambda_{\max}(M)$ and $\Lambda_{\min}(M)$ to denote the maximum and minimum eigenvalues of a matrix $M$.

We are interested in the following two cases that will be presented separately: $G = 0$ for the classic ADMM and $G \succ 0$ for the generalized ADMM.

**Theorem 1.** When $G = 0$, $\mathcal{X} = \mathbb{R}^x$ and $\mathcal{Y} = \mathbb{R}^y$, $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\lambda}$ converge linearly:

$$\|\mathbf{w}^{k+1} - \widetilde{\mathbf{w}}^*\|_P^2 \leq \left(\frac{1}{1+\tau}\right)^k \|\mathbf{w}^0 - \mathbf{w}^*\|_M^2, \tag{6.4.2}$$

where

$$P \equiv \left(2\mu_1 + \frac{\mu_1^2}{2\beta\Lambda_{\max}(AA^T)}, 2\mu_2 + \beta\Lambda_{\min}(B^TB), \frac{1}{\beta}\right) I, \quad \widetilde{\mathbf{w}}^* \equiv (\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^k)^T, \text{and}$$

$$\tau \equiv \frac{2\mu_2}{\frac{L_2^2}{\beta c} + \beta\Lambda_{\max}(B^TB)}. \tag{6.4.3}$$

Here $c > 0$ is the largest positive constant that satisfies

$$\|B^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)\|^2 \geq c\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 \ \ \forall k. \tag{6.4.4}$$

We observe that larger $\mu_1$ and $\mu_2$ leads to faster rates. The $1/\beta + \beta$ in the denominator of $\tau$ (6.4.3) means that $\beta$ must not be too large nor too small. This is also observed empirically, and we have more discussions in Sec.6.5 and 6.6.

**Theorem 2.** When $G \succ 0$, $\mathcal{X} = \mathbb{R}^x$ and $\mathcal{Y} = \mathbb{R}^y$, $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\lambda}$ converge linearly:

$$\|\mathbf{w}^{k+1} - \mathbf{w}^*\|_M^2 \leq \left(\frac{1}{1+\tau_G}\right)^k \|\mathbf{w}^0 - \mathbf{w}^*\|_M^2, \text{where } \tau_G \equiv \min\left\{\frac{2\mu_1}{\Lambda_{\max}(G)}, \tau\right\}, \tag{6.4.5}$$

$M$ is defined in Lemma 1 and $\tau$ is defined in (6.4.3).

Comparing with Theorem 3.1 of [44], our rate has a clearer form that captures more underlying characteristics of the problem. The constants in our bound is easily computable, which are used in our proposed Alg.6.3. Comparing with Theorem 3.4 in [20] which provides linear rates for y and lambda only, our linear rate is for both $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\lambda}$ (as shown in the $P$-norm). Also the constant in our linear rate is much tighter than [20], since the minimum eigenvalue of $AA^T$ as used in bound (3.14) of [20] might be 0 due to the rank deficiency of $A$, leading to a meaningless bound.

## *6.5 Strategy for Choosing $\beta$ Adaptively*

Despite of many efforts towards finding a good penalty parameter $\beta$ [41, 74, 9], it still remains a serious issue in implementing any instance of ADMM. This parameter controls the balance between the reductions of the dual residual $\mathbf{s}^{k+1} \equiv \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k)$ and the primal residual $\mathbf{r}^{k+1} \equiv A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}$ as defined in [6]. A large $\beta$ enforces more the primal feasibility $A\mathbf{x}_k - B\mathbf{y}_k = \mathbf{b}$, but results in a larger violation in the dual feasibility. A small $\beta$ tends to reduce the difference between $\mathbf{y}^{k+1}$ and $\mathbf{y}^k$, leading to a faster satisfaction of the dual feasibility, at the expense of a larger violation of the primal feasibility.

Moreover, a bad choice of $\beta$ might lead to very slow convergence rates for *both* the primal and dual feasibilities. A numerical example for consensus least squares is shown in Fig.6.2, where the bipartite graph consists of only two workers and the consensus constraint is simply $\mathbf{x} = \mathbf{y}$. Increasing $\beta$ from the optimal value 0.47 to 3 not only results in a significantly higher dual residual than the primal residual, but also slows down both residuals from $10^{-6}$ to $10^{-3}$ (primal) and $10^{-2}$ (dual), all measured at iteration 20. Decreasing $\beta$ to 0.1 makes the primal residual higher than the dual residual, but both are around $10^{-4}$ at iteration 20, which are still much higher than those using the optimal $\beta$.

Since the optimal parameter $\beta$ is essentially data-dependent, a natural idea is to search it adaptively during the iterations of ADMM. However we still need to answer two questions: 1. What is a good initial value $\beta^0$ that we shall start with? 2. What updating rule shall we adopt?

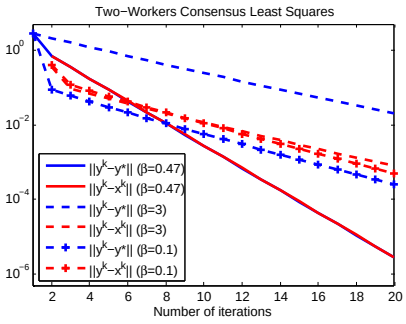



Figure 6.2: Values of $\beta$ significantly affect convergence rates for both primal and dual residuals.

Towards the first question, we can use our convergence results that are presented in Theorem 1 and 2. For simplicity, we assume that in Theorem 2 ($G \succ 0$), we always choose a $G$ such that $\frac{2\mu_1}{\Lambda_{\max}(G)} > \tau$. Then in both cases the linear convergence rate is upper bounded by $\left(\frac{1}{1+\tau}\right)^k$ where $\tau \equiv \frac{2\mu_2}{\frac{L_2^2}{\beta c}+\beta\Lambda_{\max}(B^TB)}$. Here $c > 0$ is the largest positive constant that satisfies $\|B^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)\|^2 \geq c\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2$. Since a large $\tau$ results in a faster rate, we can let $\frac{L_2^2}{\beta c} = \beta\Lambda_{\max}(B^TB)$ and take the "optimal" $\beta^* = \frac{L_2}{c\Lambda_{\max}(B^TB)}$. Although $BB^T$ is positive semidefinite, yet $B$ is not always of full row-rank. Hence in the worst case $BB^T$ could be singular and $c = \Lambda_{\min}(BB^T)$ can be as small as 0, resulting in a $\beta^* = \infty$. However, in practice a very large $\beta$ is rarely a good choice, implying that $c = \Lambda_{\min}(BB^T)$ might be too pessimistic. It is very hard to estimate $c$, since we do not know $\boldsymbol{\lambda}^*$, nor the relation between $B$ and $\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*$. Our proposed strategy is to find an underestimated $\beta$ by taking the most optimistic $\hat{c} = \Lambda_{\max}(BB^T) > c$ and the initial guess

$$\beta^0 = L_2/(\Lambda_{\max}(B^TB) * \Lambda_{\max}(BB^T)). \tag{6.5.1}$$

We can see that this underestimated $\beta^0$ is always smaller than $\beta^*$.

Towards the updating rule, we proposed a multiplicative method (Alg.6.3) that is inspired by [41, 74]. In these two papers, the authors proposed to choose $\beta$ adaptively by $\beta^{k+1} \leftarrow \beta^k * m$ if $q^k = \frac{\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|}{\|A^TB(\mathbf{y}^{k+1}-\mathbf{y}^k)\|}$ is larger than some threshold $q^{\text{th}}$, where $m > 1$ is a fixed and predefined constant. Typical choices might be $q^{\text{th}} = 10$ and

$m = 2$ [6]. In comparison, we propose to update $\beta^k$ by multiplying an adaptive number $\sqrt{q^k} \equiv \sqrt{\frac{\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|}{\|B(\mathbf{y}^{k+1}-\mathbf{y}^k)\|}}$. This simple method is motivated by the idea of balancing the convergence rates of the primal residual $\mathbf{r}^{k+1} \equiv A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}$ and the dual residual $\mathbf{s}^{k+1} \equiv \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k)$. Intuitively, the more $q^k$ is deviated from 1, the further $\beta^k$ is from $\beta^*$, hence deserving a larger scaling. Concrete examples that support this intuition are given in Sec. 6.6.

**Algorithm 6.3** Adaptive $\beta$ for ADMM

Initialize $\beta^0 = L_2/(\Lambda_{\max}(B^TB) * \Lambda_{\max}(BB^T))$.
**for** $k = 0, 1, 2, \ldots$ **do**
  $q^k = \frac{\|A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}\|}{\|A^TB(\mathbf{y}^{k+1}-\mathbf{y}^k)\|}$
  **if** $q^k \geq q^{\text{th}}$ or $q^k \leq \frac{1}{q^{\text{th}}}$ **then**
    $\beta^{k+1} \leftarrow \beta^k * \sqrt{q^k}$
  **end if**
**end for**

Additionally, for our distributed consensus learning (Alg.6.2), it is extremely easy to obtain $\Lambda_{\max}(B^TB)$ and $\Lambda_{\max}(BB^T)$. They are simply the maximum degree of the right nodes of the bipartite graph, as summarized in the following result.

**Proposition 1.** Let matrix $B \in \mathbb{R}^{D|\mathcal{C}| \times D|\mathcal{V}_R|}$ be of $|\mathcal{C}|$ block-rows and $|\mathcal{V}_R|$ block-columns, with each row block having only one $-I$, and $\mathbf{0}$ for others (Figure 6.1 (Right)). Then $\Lambda_{\max}(B^TB) = \Lambda_{\max}(BB^T) = \max\{\text{Degree}(v \in \mathcal{V}_R)\}$.

## *6.6 Numerical Results*

In this section, several sets of numerical examples will be used to: a) empirically demonstrate how ADMM's three degrees of freedom affect our proposed consensus learning algorithm; b) illustrate how well the proposed adaptive $\beta$ updating strategy works. In addition, we proposed a practical $\beta$ that works quite well for simple ADMM instances where $A = I$ and $B = -I$.

### 6.6.1 Experimental Settings

In all examples presented in this section, we generate a dataset for the following distributed regression task: $\min_{\mathbf{x}} f(\mathbf{x}) \equiv \sum_{i=1}^{N}(S_i^T\mathbf{x}_i - \mathbf{l}_i)^2, \quad \text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_N$. We assume that the total $48,000$ data samples are evenly distributed among $N = 24$ workers. Each worker $i$ has $2,000$ samples of $D = 50$ dimensions. Components of the data matrix $S_i$ of each worker are generated from the normal distribution $\mathcal{N}(0, 1)$. The real regression coefficients $\mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_{\text{true}} \in \mathbb{R}^D$ have 10% zeros, and each non-zero dimension is draw from the normal distribution $\mathcal{N}(0, 1)$. The dependent variables (labels) are perturbed by Gaussian white noise $\mathcal{N}(0, 10^{-4})$.

For comparison purposes, we consider the following communication topologies:

- *Complete bipartite.* The 24 workers are divided into two groups: 12 are on the left $\mathcal{V}_L$ and 12 on the right $\mathcal{V}_R$. Each worker communicates with all the other 12 workers on the other partition.

- *Master-salve.* The 24 workers are divided into two groups of 1 and 23 workers each. The master communicates with all the 23 slaves on the other partition. It is $(23, 1)$- or $(1, 23)$-biregular.

- $(3, 3)$-*Biregular.* The bipartition is the same as the complete bipartite. Each worker has the same degree 3.

- *Bucky spanning tree.* The 24 workers form a spanning tree, where is taken from a buckyball

- *Ring.* A ring is also a $(2, 2)$-biregular graph.

- *Ring+1edge.* An additional edge of the longest chord is added to the ring, making it not biregular.

- *Chain.* A chain is the spanning tree with the largest diameter.

### 6.6.2 Varying $\beta$

We have already presented a very simple example in Section 6.5 showing that a bad choice of $\beta$ can significantly slow down the convergence of ADMM. Now we use the complete bipartite communication topology to show that $\beta$ is still a crucial parameter for the distributed consensus learning with more than 2 workers. The primal residual $\|\mathbf{r}^k\|$ and dual residual $\|\mathbf{s}^k\|$ are plotted in Fig.6.3 as functions of both the number of iterations and $\beta$. We have several observations. First, both residuals converge linearly for any $\beta$ values we tried from $10^{-2} \sim 10^2$, although some $\beta$ converge faster than the others. This is expected, since our linear convergence rates in Theorem 1 and 2 are simple functions of $\beta$, no matter how large or small it is. Second, for small $\beta$, the primal residual $\|\mathbf{r}^k\|$ is larger than the dual $\|\mathbf{s}^k\|$, and for large $\beta$ the reverse holds. Third, the "optimal" $\beta^* = 0.4467$ is neither too large nor too small. It is the parameter that achieves the lowest values for *both* $\|\mathbf{r}^k\|$ and $\|\mathbf{s}^k\|$, and these two lowest values are very close to each other. This observation provides some evidences for the effectiveness of our proposed strategy of adaptive $\beta$ (Alg.6.2).

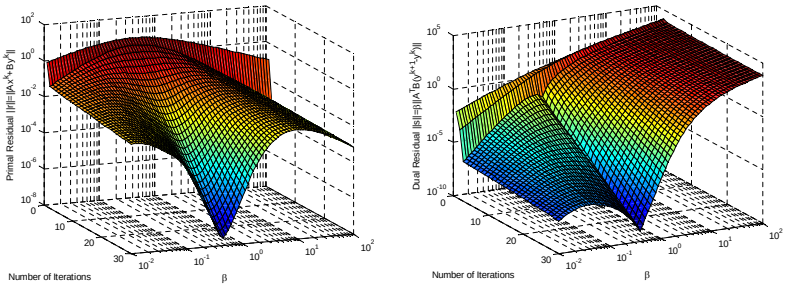



Figure 6.3: Primal and Dual residuals as functions of $\beta$ and number of iterations. Topology: complete bipartite graph.

### 6.6.3 Comparing Communication Topologies Using Optimal $\beta$s

As we discussed in Section 6.3.1, the three degrees of freedom of ADMM all contribute to the convergence speed of the algorithm. Their interplay is so complex that it is

not easy to draw a clear conclusion of which communication topology is the "best". Here we simplify this problem by fixing the other two degrees and only explore the effects of communication topologies. For each topology, we seek the "optimal" $\beta$ from a set of $1,000$ candidates ranging from $\beta^0/10$ to $100\beta^0$, where the formula for the underestimated $\beta^0$ is given in (6.5.1) and Proposition 1 can be used to calculate the maximum eigenvalues.

The fastest possible primal and dual convergences for each topology are plotted in Fig.6.4 (Left). Again we can observe that all residuals converge linearly, and the values of $\|\mathbf{r}^k\|$ and $\|\mathbf{s}^k\|$ are very close at the same iteration given the optimal $\beta$ of each topology. It is also very clear that the complete bipartite and master-slave topologies converge at almost the same rate, and they are both faster than the others. This is an interesting observation, since the complete bipartite graph has 144 edges, which is higher than the master-slave's 23, however the master-slave have a higher bandwidth requirement for the master node than the complete bipartite where the bandwidth requirement is balanced for all workers. The $(3,3)$-biregular graph is much faster than the bucky spanning tree, although they have the same maximum degree 3 for each bipartition. This might be due to the fact that the spanning tree taken from the buckyball graph has a minimum degree 1 for some workers. The spanning tree is even slower than the $(2,2)$-biregular ring, implying that a biregular graph might be preferred for the faster convergence rates of consensus learning. This preference can be also observed from the comparison between the ring and the ring+one edge, where more edges do not necessarily lead to faster rates. The chain topology is the slowest one, which is expected, since it has the smallest number of edges and the smallest minimum (1) and maximum (2) degrees.

### 6.6.4 Adaptive $\beta$ using Alg.6.3

The above observations verify that an effective implementation of our consensus learning (Alg.6.2) heavily relies on a good $\beta$. Hence in the follows we use the distributed consensus learning task as a testbed for our proposed adaptive $\beta$ strategy (Alg.6.3). Note that this method is very general and can be used as a plug-in for other ADMM instances. All the experimental settings are the same as Subsection 6.6.3, except that we replace the fixed "optimal" $\beta$ with the adaptive strategy. As a comparison, we implemented He et al.'s adaptive $\beta$ proposed in [41, 74] using the parameters suggested in [6], and take the initial $\beta^0 = 1$ for all topologies. We plot the convergence history of the primal and dual residuals in Fig.6.4. Comparing Fig.6.4 (Mid) with Fig.6.4 (Left) one can observe that the proposed strategy for $\beta$ works very well. The convergence rates are very close to those with "optimal" $\beta$s. Residuals for the master-slave topology are not monotonically decreasing, but the overall rates are still comparable with the optimal case, if not any faster. He et al.'s method (Fig.6.4 (Right)) works reasonably well for some topologies, but is still much slower than our proposed method, except for the master-slave. One reason might be that the uninformative initial guess $\beta^0 = 1$ is improper, and it should be both data- and topology-dependent as we suggested in Alg.6.3.

Figure 6.4: Primal and dual residuals by the optimal $\beta$s (Left), proposed Alg.6.3 (Mid) and method of [41, 74, 6] (Right).

### 6.6.5 Changing the Updating Order

The third degree of freedom for ADMM is the order with which $\mathbf{x}$ and $\mathbf{y}$ are updated. Although we have no pointers coming directly from our theoretical results, empirically it is the weakest factor comparing with the communication topology and the value of $\beta$. We test it using the same settings as in Subsection 6.6.3. We observe that for all topologies except the master-slave, after changing the updating order, the changes of convergence rates are tiny, and the optimal $\beta^*$ are essentially the same as before. For the master-slave topology, similar convergence rates can also be obtained, although we have to reduce the optimal $\beta^*$ from 4.71 to 4.33.

### 6.6.6 Practical $\beta$ for the Simple Case: $\mathbf{x} = \mathbf{y}$

In this last set of experiments, we present a practical $\beta$ for the case where the constraint of ADMM is simply $\mathbf{x} = \mathbf{y}$, i.e. $A = I, B = -I$ and $\mathbf{b} = 0$. We found that taking the fixed penalty parameter $\beta = \sqrt{\mu_1 L_2}$ works quite well in practice although currently we do not have any theoretical evidence to support its effectiveness. To satisfy the smoothness and strongly-convex assumptions, we use the ridge regression $\min_{\mathbf{x}} \sum_{i=1}^{N} (\mathbf{x}^T \mathbf{s}_i - l_i)^2 + \frac{\alpha}{2}\|\mathbf{x}\|^2$ as our objective function. Putting it in ADMM's canonical form (6.2.2) we have $\theta_1(\mathbf{x}) = \sum_{i=1}^{N} (\mathbf{x}^T \mathbf{s}_i - l_i)^2$ and $\theta_2(\mathbf{y}) = \frac{\alpha}{2}\|\mathbf{y}\|^2$. We test this $\beta$ using $2,000$ samples of dimension 50. In this simulated dataset, $\mu_1 = 1,436.5$. When $\alpha = L_2 = 1$, $\sqrt{\mu_1 L_2} = 37.90$, and the optimal $\beta^*$ is 39.64. When $\alpha = L_2 = 100$, $\sqrt{\mu_1 L_2} = 379.02$, and the optimal $\beta^*$ is 384.42.

## *6.7 Summary and Future Work*

In this paper, we presented an ADMM-based consensus learning method for training distributed data samples in parallel. We used bipartite communication topologies to take advantage of ADMM's capacities in dealing with separable functions. We identify the three degrees of freedom in implementing this method: communication topology,

penalty parameter $\beta$ and the order for updating variables. In order to investigate the joint effects of these factors, we provided an analysis of ADMM's convergence behavior. The analysis demonstrates that all the primal and dual variables enjoy a linear rate of convergence. Due to the difficulty in obtaining a very sharp rate from which the optimal $\beta^*$ can be derived, we proposed a strategy for choosing $\beta$ adaptively, with an underestimated initial guess $\beta_0$ that is derived from our bound. Numerical experiments show that $\beta^*$ is achieved at a point where the norms of primal and dual residuals are close and decrease at the fastest rate. With $\beta^*$, the complete bipartite and the master-slave graphs converge fastest, followed by bi-regular graphs. The proposed strategy of adaptive $\beta$ is very efficient.

There are several interesting directions that remain to be explored. A tighter and more instructive bound is deserved. It is possible to extend our method to asynchronous variants. It is also promising to investigate the possibilities with assumptions weaker than Assumption 1. A potential application is the distributed consensus Lasso.

# CHAPTER VII

# SOFTWARE PACKAGE

To facilitate reproducible research, we develop an open-source software package that consists efficient implementation of the point process inference algorithms proposed in this dissertation. We also include inference algorithms such as the vanilla Hawkes processes reviewed in Chapter 2 or the Criteo model mentioned in Chapter 5 for benchmarking purpose.

The overview of our software package is given in Figure 7.1. At current stage, the software package consists of the following components

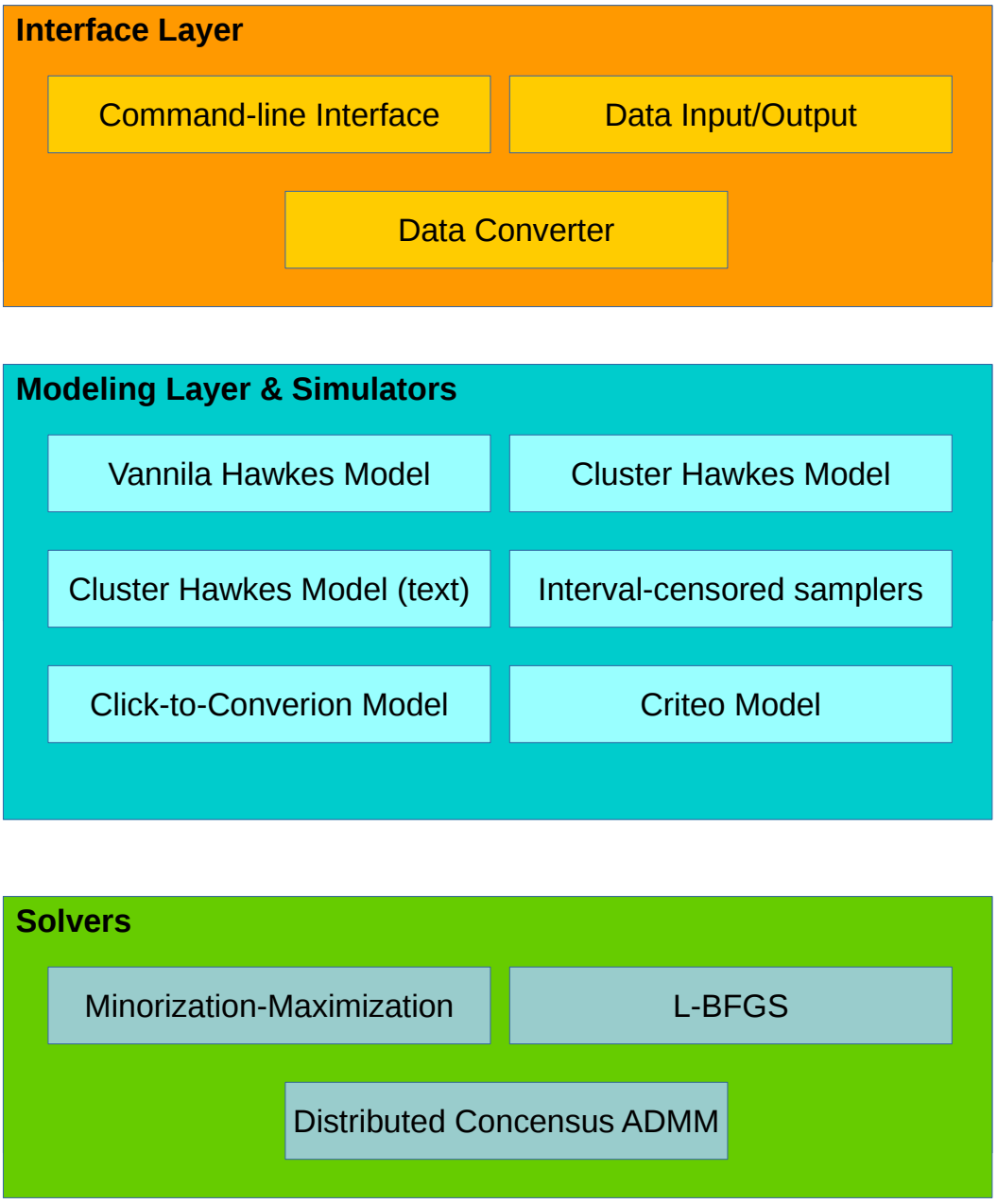
- Data: Data input/output capabilities for classification data, point process data, advertisement data.
- Optimization: template for optimization methods such as Line Search, Gradient Descent, and L-BFGS.
- Point Process: various Hawkes processes implementations
- Conversion: the click-to-conversion models.

Further details for each module are shown in Figures 7.2, 7.3, 7.4, 7.5.

## *7.1 Efficient implementation*

The key advantages of our software package are *performace* and *scaling*. The software allows users to get insights into timestamped events much faster than naive implementation. To that end, we employ C++ as the programming langaguage due to its efficiency and multi-thread programming with the Thread Building Block (TBB)

Figure 7.1: Software package design.




library (Intel) for scalable parallel implementation. The Intel TBB library is used extensively in all model implementations to speed up with parallelism.

With huge amount of data available, for statistical models to be useful, both training (i.e. parameter estimation) and testing (i.e. on unseen data) needs to be

Figure 7.2: Data module.


Figure 7.3: Optimization module.


fast. Our implementations not only optimize the training and testing phase but also employ special data I/O techniques such as memory-mapped files, compressed files to speed up the data preparation phase of the propose algorithms. We also convert the text-based dataset into binary-based dataset for much more efficient input/output.

Besides, the users also like to be able to alter the models once new data are available. Using our software package, the user could save models to file, load them from file and continue training with new data at any time.

Figure 7.4: Point process module.

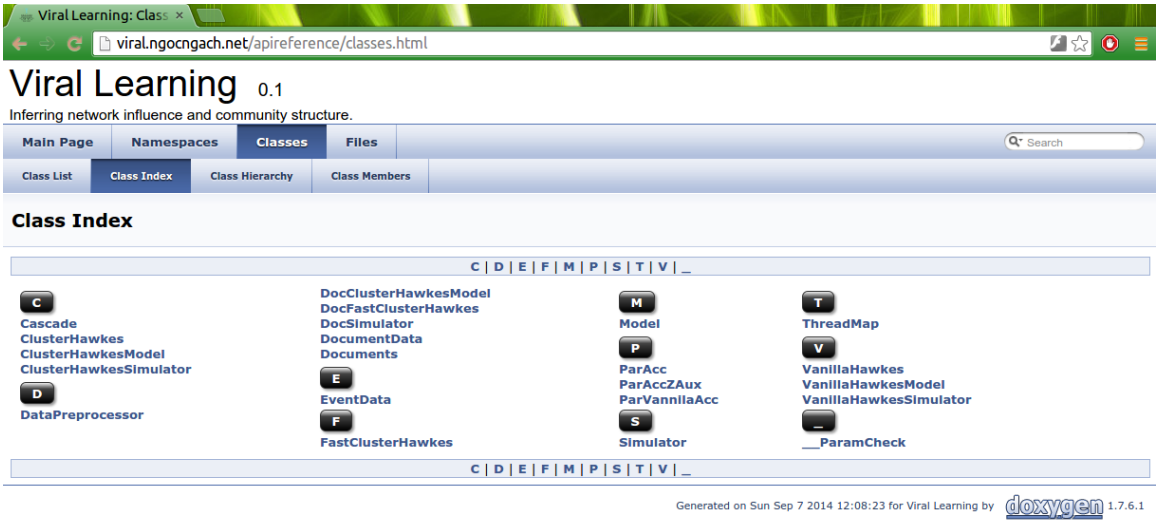
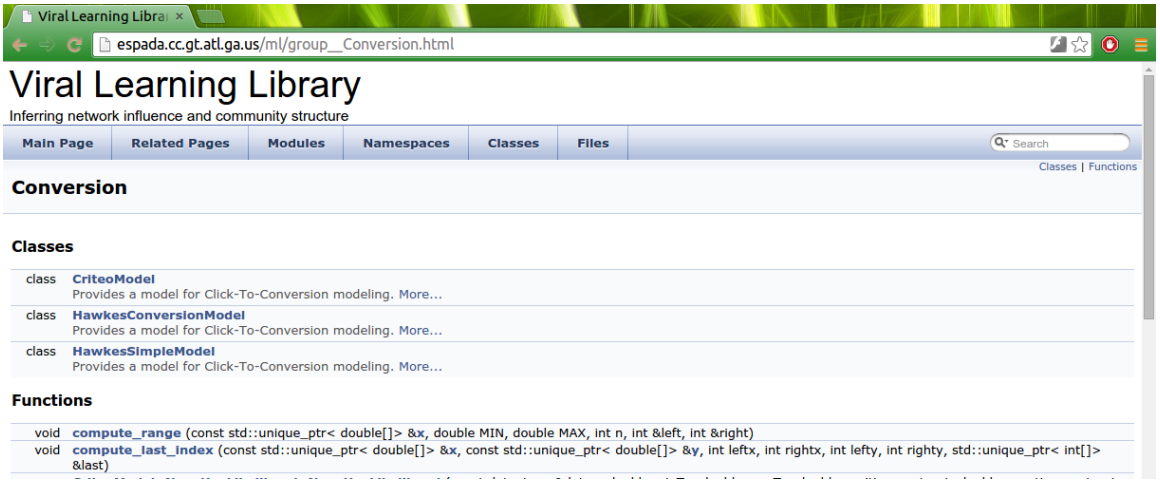



Figure 7.5: Conversion module.




## 7.2 *Detailed code documentation*

To facilitate code usage and modification, we provide extensive documentation in the code base. We utilize Doxygen code-documentation tool and provide formatted comments on all classes, functions, and parameters that are used in the code. Doxygen is then run to generate formatted code documentation website shown in the figures. The website not only consists of detailed documentation but also class diagrams of the code base.

The document website for our software could be found at “http://viral.ngocngach.net”. More details about the software package could be found here along with the associated publications of the proposed algorithms in this dissertation.

# CHAPTER VIII

# CONCLUSION

In this thesis, we investigate the inference problems of different temporal point processes, namely the Hawkes processes and thinned processes. We show that, in the case that the activities generating network forms communities or in the case that only interval censored activity data are available, there are efficient inference algorithms that estimate the intensity function via parametric models. We also show that by modeling the Click-to-Conversion mechanism using thinned processes, one could efficiently predict the future conversion volume and that one could estimate the conversion rate of an advertisement campaign.

In Chapter 3, we propose that one could infer the network of social influence along with its community structure from the observed recurrent events in the social networks. To that end, we utilize the key observation that regular activities often raise influence among users in the same group. The proposed model based on the Hawkes model is designed to take into account this observation and other assumptions such as the low-rank structure. The inference algorithm following the mean-field variational principle nicely consists of closed form updates that could be sped up by various implementation techniques including parallelism. The experiments on simulated dataset show that the proposed model could estimate both network infectivity and and community structure and produce better predictive model with less training samples than the baseline methods. Experiments on real dataset show that the proposed method are able to produce meaningful clusters using only activities from websites. There are interesting paths to extend this study: First, we plan to investigate the extensions that cover other features of an event, for example, document

content and ratings. The content and ratings effects on community structure could be expressed in the factorization of the influence between events. Moreover, it is also interesting to incorporate the memes/trends and community structure in one model.

In Chapter 4, we propose to infer the network influence from the interval censored events of user activity in the social networks. The underlying event triggering mechanism is modeled by the self-exciting multi-dimensional Hawkes process which is able to capture the temporal patterns of user behavior under influence of other users. We propose an imputation approach in which events are sampled under the count constraints and the maximum likelihood estimator is utilized to re-estimate the parameters. We then propose a Gibbs sampling method that could impute timestamps given the count of events. The proposed method is compared to four baseline sampling methods that not only have good intuitions but also reasonably good performance on test data. The experiment results show that the proposed method is able to estimate the influence among nodes when only counts of events in observed intervals are available. There are several interesting directions for future studies: First, we would like to make the proposed Gibbs sampling method more efficient as it now has to cycle through the timestamps without any parallelization. This is contrary to the MLE solver for Hawkes process which could be sped up tremendously via parallel implementation. Second, we plan to investigate a variational inference approach in which events could be sampled from a simpler distribution to compute a lower bound of the likelihood of the data. Moreover, we can also investigate a problem of estimating the parameters with left censored data (i.e. no information on the first events).

In chapter 5, we proposed a Click-to-Conversion model based on the thinned processes framework. The click process is represented by a Hawkes process under the assumption that customer often clicks many times on an advertisement. The conversion process is represented by a thinned process of the click process with a logistic

model specifying the thinning probability and an exponential distribution for the click-to-conversion delay. We then derive an efficient Maximum Likelihood Estimation algorithm to estimate the model paramters. The experiments show that the proposed model has good predicting capability with respect to future click and conversion volume. In the near future, we would like to enhance the model by considering the influence of previous conversions on future conversions. Besides, we also would like to introduce into the model the association between clicks and conversions so that it could have better prediction power on each click.

In chapter 6, we presented an ADMM-based consensus learning method for training distributed data samples in parallel. We used bipartite communication topologies to take advantage of ADMM's capacities in dealing with separable functions. We identify the three degrees of freedom in implementing this method: communication topology, penalty parameter and the order for updating variables. In order to investigate the joint effects of these factors, we provided an analysis of ADMM's convergence behavior. The analysis demonstrates that all the primal and dual variables enjoy a linear rate of convergence. Due to the difficulty in obtaining a very sharp rate from which the optimal penalty parameter can be derived, we proposed a strategy for choosing it adaptively, with an underestimated initial guess that is derived from our bound. Numerical experiments show that the optimal penalty parameter is achieved at a point where the norms of primal and dual residuals are close and decrease at the fastest rate. With this choice, the complete bipartite and the master-slave graphs converge fastest, followed by bi-regular graphs. The proposed strategy of adaptive penalty is very efficient. There are several interesting directions that remain to be explored. A tighter and more instructive bound is deserved. It is possible to extend our method to asynchronous variants. It is also promising to investigate the possibilities with assumptions weaker than currently proposed. A potential application is the distributed consensus Lasso.

# APPENDIX A

# NETCODEC PROOFS

## *A.1 Expression of $\mathcal{L}(\mathbf{Z}, \boldsymbol{t})$*

$$\begin{aligned}\mathcal{L}(\mathbf{Z}, \boldsymbol{t}) =& \text{const} + \sum_{i=1}^{I}\sum_{g=1}^{G}(a_g^0 - 1)\ln z_{ig} - b_g^0 z_{ig} \\ &+ \sum_{c=1}^{C}\sum_{n=1}^{N_c}\ln\left(\mu_{i_n^c} + \sum_{\substack{\ell<n \\ i_\ell^c \neq i_n}} \alpha_{i_n^c i_\ell^c}\kappa(t_n^c - t_\ell^c) + \alpha_{i_n^c}\sum_{\substack{\ell<n \\ i_\ell^c = i_n}}\kappa(t_n^c - t_\ell^c)\right) \\ &- \sum_{c=1}^{C}\sum_{i=1}^{I}\int_0^{T_c}\left[\mu_i + \sum_{\substack{t_n<t \\ i_n^c \neq i}}\alpha_{i i_n^c}\kappa(t - t_n^c) + \alpha_i\sum_{\substack{t_n<t \\ i_n^c = i}}\kappa(t - t_n^c)\right]dt,\end{aligned} \tag{A.1.1}$$

where $\alpha_{ij} = \sum_{g=1}^{G}\beta_j z_{ig} z_{jg}, i \neq j$, is the infectivity rate from user $j$ to user $i$.

## *A.2 Proof of Theorem 3.3.1*

From Eq. (A.1.1), for the first term (i.e. prior term), we have

$$\mathbb{E}_q\left[\sum_{i=1}^{I}\ln\mathbb{P}(\mathbf{Z}_i)\right] = \sum_{i=1}^{I}\sum_{g=1}^{G}(a_g^0 - 1)\mathbb{E}_q\left[\ln z_{ig}\right] - b_g^0\mathbb{E}_q\left[z_{ig}\right].$$

For the second term (i.e. occurred events), let $A_n^c$ be the expression inside the logarithms

$$A_n^c = \mu_{i_n^c} + \sum_{\substack{\ell<n \\ i_\ell^c \neq i_n}}\sum_{g=1}^{G}\beta_{i_\ell^c z_{i_n^c} g} z_{i_\ell^c g}\kappa(t_n^c - t_\ell^c) + \alpha_{i_n^c}\sum_{\substack{\ell<n \\ i_\ell^c = i_n}}\kappa(t_n^c - t_\ell^c).$$

For any positive values $\eta_n^c, \eta_{\ell n}^{gc}, \gamma_n^c$ such that $\eta_n^c + \sum_{\substack{\ell<n \\ i_\ell^c \neq i_n}}\sum_{g=1}^{G}\eta_{\ell n}^{gc} + \gamma_c^c = 1$, one could *lower-bound* $\mathbb{E}_q\left[\ln A_n^c\right]$ with

$$\eta_n^c\ln\frac{\mu_{i_n^c}}{\eta_n^c} + \sum_{\substack{\ell<n \\ i_\ell^c \neq i_n}}\sum_{g=1}^{G}\eta_{\ell n}^{gc}\mathbb{E}_q\left[\ln(\beta_{i_\ell^c z_{i_n^c} g} z_{i_\ell^c g}\kappa(t_n^c - t_\ell^c))\right] - \eta_{\ell n}^{gc}\ln\eta_{\ell n}^{gc} + \gamma_n^c\ln\left(\frac{\alpha_{i_n^c}\sum_{\substack{\ell<n \\ i_\ell^c = i_n}}\kappa(t_n^c - t_\ell^c)}{\gamma_n^c}\right)$$

where we apply *Jensen's inequality*, the concavity of natural logarithm and the additivity of expectation.

For the last term (i.e. normalization term), we have

$$\mathbb{E}_q\left[\sum_{c=1}^{C}\sum_{i=1}^{I}\int_0^{T_c}\lambda_i^c(t)dt\right] = \sum_{c=1}^{C}\sum_{i=1}^{I}\mu_i T_c + \sum_{c=1}^{C}\sum_{i=1}^{I}\left(\sum_{\substack{n=1\\ i_n^c\neq i}}^{N_c}\sum_{g=1}^{G}\beta_{i_n^c}\mathbb{E}_q\left[z_{ig}z_{i_n^c g}\right]K(T_c-t_n^c) + \sum_{\substack{n=1\\ i_n^c=i}}^{N_c}\alpha_i K(T_c-t_n^c)\right).$$

Adding the terms derived above, the theorem is proved.

## *A.3 Derivation of optimal distribution* $q_i^\star(\mathbf{Z}_i)$

The optimal distribution $q_i^\star(\mathbf{Z}_i)$ satisfies

$$\ln q_i^\star(\mathbf{Z}_i) = \mathrm{E}_{q_{-\mathbf{Z}_i}}[\mathcal{L}(\mathbf{Z},\boldsymbol{t})] + \text{const},$$

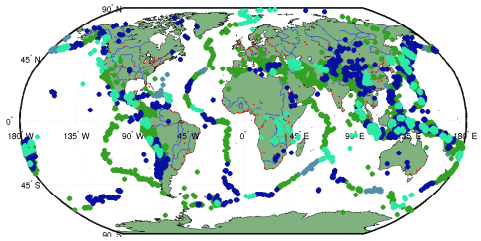
where the expectation is over all $\mathbf{Z}_j, j\neq i$. Replacing $\mathcal{L}(\mathbf{Z},\boldsymbol{t})$ with the lower-bound in Theorem 3.3.1, collecting all relevant terms to $\mathbf{Z}_i$, and absorbing everything else in the constant, one could find that $\ln q_i^\star(\mathbf{Z}_i)$ has the following form

$$\begin{aligned}\ln q_i^\star(\mathbf{Z}_i) = \text{const} &+ \sum_{g=1}^{G}(a_g^0-1)\ln z_{ig} - b_g^0 z_{ig} \\ &+ \sum_{c=1}^{C}\left[\sum_{n=1}^{N_c}\sum_{\ell<n}\sum_{g=1}^{G}\eta_{\ell n}^{gc}\delta_{\ell n}^{ic}\ln z_{ig} - \sum_{\substack{n=1\\ i_n^c\neq i}}^{N_c}\sum_{g=1}^{G} z_{ig}\beta_{i_n^c}\mathbb{E}_q\left[z_{i_n^c g}\right]K(T_c-t_n^c)\right. \\ &\left. - \sum_{j\neq i}\sum_{\substack{n=1\\ i_n^c=i}}^{N_c}\sum_{g=1}^{G} z_{ig}\beta_i\mathbb{E}_q\left[z_{jg}\right]K(T_c-t_n^c)\right], \qquad \text{(A.3.1)}\end{aligned}$$

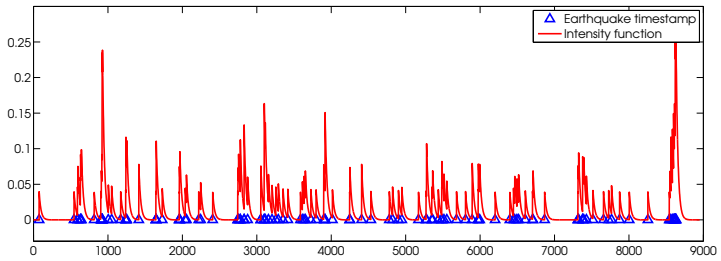
This expression shows that $z_{ig}, g=1,\ldots,G$ are independent Gamma random variables and $z_{ig}\sim\text{Gamma}(a_{ig},b_{ig})$ where $a_{ig},b_{ig}$ are defined in Eq. (3.3.1).

## *A.4 Earthquake experiments*

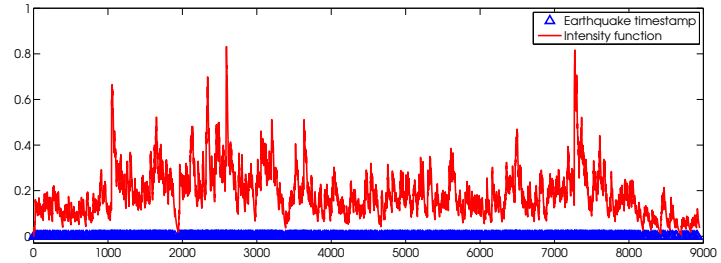
To supplement the experiment on Earthquake dataset, we draw the intensity function for the clusters of earthquakes around certain region on the global map (Figure A.1).



(a) Earthquake map




(b) Intensity function for the cluster on the Parcific region to the west of South America.




(c) Intensity function for the cluster around Japan.

Figure A.1: Earthquake experiments.

We choose two regions to demonstrate the working of our proposed algorithm. The first region is the Parcific ocean region close to the West of South America (Figure A.1b). One could see that the intensity function for this cluster form regions of spike indicating that events form clusters. On the other hand, the intensity function for the region around Japan (Figure A.1c) shows no clear cluster. It therefore leads to clustering result that does not match the geological clusterings. This shows that while time information could provide valuable information, in practice, one may want to incorporate as much information as possible to have better understanding of the data at hand. One possible direction is to use a triggering kernel $\kappa(dt, dx)$ where $dt$ is the

difference in time and $dx$ is the difference in location.

# APPENDIX B

# STOCHADMM PROOFS

## *B.1 Auxillary Lemma*

We first give a lemma presented in [67]. It will be used to prove our linear convergence rate.

**Lemma 2.** Let $l(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ be a convex differentiable function with gradient $\mathbf{g}$. Let scalar $s \geq 0$. For any vector $\mathbf{u}$ and $\mathbf{v}$, denote their Bregman divergence as $D(\mathbf{u}, \mathbf{v})$. If $\forall \mathbf{u} \in \mathcal{X}$, $\mathbf{x}^* \equiv \arg\min_{\mathbf{x}\in\mathcal{X}} l(\mathbf{x}) + sD(\mathbf{x}, \mathbf{u})$, then with $\Theta \equiv \langle \mathbf{g}(\mathbf{x}^*), \mathbf{x}^* - \mathbf{x} \rangle$, we have

$$\Theta \leq s\left[D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x}^*, \mathbf{u}) - D(\mathbf{x}, \mathbf{x}^*)\right]. \tag{B.1.1}$$

*Proof.* Invoking the optimality condition we have

$$\langle \mathbf{g}(\mathbf{x}^*) + s\nabla D(\mathbf{x}^*, \mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \geq 0, \ \forall \mathbf{x} \in \mathcal{X},$$

which is equivalent to

$$\begin{aligned}
\langle \mathbf{g}(\mathbf{x}^*), \mathbf{x}^* - \mathbf{x} \rangle &\leq s \langle \nabla D(\mathbf{x}^*, \mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \\
&= s \langle \nabla \omega(\mathbf{x}^*) - \nabla \omega(\mathbf{u}), \mathbf{x} - \mathbf{x}^* \rangle \\
&= s\left[D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x}, \mathbf{x}^*) - D(\mathbf{x}^*, \mathbf{u})\right].
\end{aligned}$$

□

## *B.2 Proof for Lemma 1*

*Proof.* By the strong convexity of $\theta_1$ and $\theta_2$ we have $\forall \mathbf{x} \in \mathcal{X}$ and $\forall \mathbf{y} \in \mathcal{Y}$:

$$\theta_1(\mathbf{x}^{k+1}) - \theta_1(\mathbf{x}) \leq \left\langle \theta_1'(\mathbf{x}^{k+1}), \mathbf{x}^{k+1} - \mathbf{x} \right\rangle - \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2. \tag{B.2.1}$$

$$\theta_2(\mathbf{y}^{k+1}) - \theta_2(\mathbf{y}) \leq \left\langle \theta_2'(\mathbf{y}^{k+1}), \mathbf{y}^{k+1} - \mathbf{y} \right\rangle - \frac{\mu_2}{2}\|\mathbf{y}^{k+1} - \mathbf{y}\|^2. \tag{B.2.2}$$

Invoking the optimality condition of Line 2 of Alg. 6.1 we have $\forall \mathbf{y} \in \mathcal{Y}$:

$$\langle \theta_2'(\mathbf{y}^{k+1}) + B^T \left[\beta(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}) - \boldsymbol{\lambda}^k\right], \mathbf{y}^{k+1} - \mathbf{y}\rangle \leq 0. \tag{B.2.3}$$

Using Lemma 2 by taking the Bregman divergence $D(\cdot,\cdot)$ as $\|\|_G^2$ ($G \succeq 0$) we have $\forall \mathbf{x} \in \mathcal{X}$:

$$\begin{aligned}
&\theta_1(\mathbf{x}^{k+1}) - \theta_1(\mathbf{x}) + \left\langle \mathbf{x}^{k+1} - \mathbf{x}, -A^T\boldsymbol{\lambda}^{k+1}\right\rangle \\
&\overset{(B.2.1)}{\leq} \left\langle \theta_1'(\mathbf{x}^{k+1}) - A^T\boldsymbol{\lambda}^{k+1}, \mathbf{x}^{k+1} - \mathbf{x}\right\rangle - \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 \\
&= \left\langle \theta_1'(\mathbf{x}^{k+1}) + A^T\left[\beta(A\mathbf{x}^{k+1} + B\mathbf{y}^k - \mathbf{b}) - \boldsymbol{\lambda}^k\right], \mathbf{x}^{k+1} - \mathbf{x}\right\rangle \\
&- \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 + \left\langle \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{x}\right\rangle \\
&\overset{(B.1.1)}{\leq} \frac{1}{2}\left(\|\mathbf{x} - \mathbf{x}^k\|_G^2 - \|\mathbf{x} - \mathbf{x}^{k+1}\|_G^2\right) - \frac{1}{2}\|\mathbf{x}^k - \mathbf{x}^{k+1}\|_G^2 \\
&- \frac{\mu_1}{2}\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 + \left\langle \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{x}\right\rangle
\end{aligned} \tag{B.2.4}$$

The last term can be further bounded as

$$\begin{aligned}
&\left\langle \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k), \mathbf{x}^{k+1} - \mathbf{x}\right\rangle \\
&= \frac{\beta}{2}\left(\|A\mathbf{x} + B\mathbf{y}^k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}^{k+1} - \mathbf{b}\|^2\right) \\
&+ \frac{\beta}{2}\left(\|A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}\|^2 - \|A\mathbf{x}^{k+1} + B\mathbf{y}^k - \mathbf{b}\|^2\right) \\
&= \frac{\beta}{2}\left(\|A\mathbf{x} + B\mathbf{y}^k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}^{k+1} - \mathbf{b}\|^2\right) \\
&- \frac{\beta}{2}\|\mathbf{y}^k - \mathbf{y}^{k+1}\|_{B^TB}^2 - (\mathbf{y}^k - \mathbf{y}^{k+1})^T B^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k+1}) \\
&\leq \frac{\beta}{2}\left(\|A\mathbf{x} + B\mathbf{y}^k - \mathbf{b}\|^2 - \|A\mathbf{x} + B\mathbf{y}^{k+1} - \mathbf{b}\|^2\right) \\
&- \frac{\beta}{2}\|\mathbf{y}^k - \mathbf{y}^{k+1}\|_{B^TB}^2,
\end{aligned} \tag{B.2.5}$$

where in the last step we used Lemma 3.1 of [42]. Combining (B.2.2) and (B.2.3) we have

$$\theta_2(\mathbf{y}^{k+1}) - \theta_2(\mathbf{y}) + \left\langle \mathbf{y}^{k+1} - \mathbf{y}, -B^T\boldsymbol{\lambda}^{k+1}\right\rangle \leq -\frac{\mu_2}{2}\|\mathbf{y}^{k+1} - \mathbf{y}\|^2. \tag{B.2.6}$$

We also have the following equality from the updating rule of $\boldsymbol{\lambda}$ in Line 3:

$$\begin{aligned}
&\left\langle \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}, A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b} \right\rangle \\
&= \frac{1}{\beta} \left\langle \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}, \boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{k+1} \right\rangle \\
&= \frac{1}{2\beta} \left( \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{k}\|^2 - \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{k+1}\|^2 \right) - \frac{1}{2\beta} \|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k}\|^2.
\end{aligned} \tag{B.2.7}$$

Summing (B.2.4), (B.2.5), (B.2.6) and (B.2.7), taking $\mathbf{x} = \mathbf{x}^*, \mathbf{y} = \mathbf{y}^*, \boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ and using the fact that $A\mathbf{x}^* + B\mathbf{y}^* - \mathbf{b} = 0$ we get

$$\begin{aligned}
&\frac{1}{2} \left( \|\mathbf{x}^k - \mathbf{x}^*\|_G^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^*\|_G^2 \right) \\
&+ \frac{\beta}{2} \left( \|\mathbf{y}^k - \mathbf{y}^*\|_{B^TB}^2 - \|\mathbf{y}^{k+1} - \mathbf{y}^*\|_{B^TB}^2 \right) \\
&+ \frac{1}{2\beta} \left( \|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*\|^2 - \|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 \right) \\
&\geq \frac{\mu_1}{2} \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \frac{\mu_2}{2} \|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 \\
&+ \frac{1}{2} \|\mathbf{x}^k - \mathbf{x}^{k+1}\|_G^2 + \frac{\beta}{2} \|\mathbf{y}^k - \mathbf{y}^{k+1}\|_{B^TB}^2 + \frac{1}{2\beta} \|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|^2 \\
&+ \left[ \theta_1(\mathbf{x}^{k+1}) - \theta_1(\mathbf{x}^*) + \theta_2(\mathbf{y}^{k+1}) - \theta_2(\mathbf{y}^*) \right] \\
&+ (\mathbf{x}^{k+1} - \mathbf{x}^*)^T (-A^T \boldsymbol{\lambda}^{k+1}) + (\mathbf{y}^{k+1} - \mathbf{y}^*)^T (-B^T \boldsymbol{\lambda}^{k+1}) \\
&+ (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)^T (A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - \mathbf{b}) \\
&\geq \mu_1 \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \mu_2 \|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 + \frac{1}{2} \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_M^2,
\end{aligned} \tag{B.2.8}$$

where the last inequality is due to the strong convexity of $\theta_1$ and $\theta_2$. □

## *B.3 Proof for Theorem 1*

*Proof.* Invoking the KKT optimality conditions for (6.2.2),

$$\theta_1'(\mathbf{x}^*) - A^T \boldsymbol{\lambda}^* = 0, \ \theta_2'(\mathbf{y}^*) - B^T \boldsymbol{\lambda}^* = 0. \tag{B.3.1}$$

Invoking the optimality conditions for Line 1 and 2 of Alg.6.1,

$$\theta_1'(\mathbf{x}^{k+1}) - A^T \boldsymbol{\lambda}^k + \beta A^T (A\mathbf{x}^{k+1} + B\mathbf{y}^k - b) = 0 \tag{B.3.2}$$

and

$$\theta_2'(\mathbf{y}^{k+1}) - B^T\boldsymbol{\lambda}^k + \beta B^T(A\mathbf{x}^{k+1} + B\mathbf{y}^{k+1} - b) = 0. \tag{B.3.3}$$

By the Lipshitz smoothness of $\theta_2$ and (B.3.1,B.3.3) we have

$$\|\theta_2'(\mathbf{y}^{k+1}) - \theta_2'(\mathbf{y}^*)\| = \|B^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*)\| \leq L_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|, \tag{B.3.4}$$

hence by the definition of $c$ (6.4.4) we have:

$$\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 \leq \frac{L_2^2}{c}\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2. \tag{B.3.5}$$

By (B.3.1) and (B.3.2) we have

$$\|\theta_1'(\mathbf{x}^{k+1}) - \theta_1'(\mathbf{x}^*)\| = \|A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k))\| \tag{B.3.6}$$

Combing (B.3.6) and the fact of strong-convexity

$$\|\theta_1'(\mathbf{x}^{k+1}) - \theta_1'(\mathbf{x}^*)\| \geq \mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|$$

we have

$$\begin{aligned}
\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 &\leq \frac{1}{\mu_1^2}\|A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \beta A^T B(\mathbf{y}^{k+1} - \mathbf{y}^k))\|^2 \\
&= \frac{1}{\mu_1^2}\|(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \beta B(\mathbf{y}^{k+1} - \mathbf{y}^k)\|_{AA^T}^2 \\
&\leq \frac{2\Lambda_{\max}(AA^T)}{\mu_1^2}\left[\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 + \beta^2\|\mathbf{y}^{k+1} - \mathbf{y}^k\|_{B^TB}^2\right]
\end{aligned} \tag{B.3.7}$$

Invoking Lemma 1 with $G = 0$ we have

$$\begin{aligned}
\frac{\mu_1^2}{2\beta\Lambda_{\max}(AA^T)}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 &\leq \frac{1}{\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*\|^2 + \beta\|\mathbf{y}^{k+1} - \mathbf{y}^k\|_{B^TB}^2 \\
&\overset{(6.4.1)}{\leq} \beta\|\mathbf{y}^k - \mathbf{y}^*\|_{B^TB}^2 + \frac{1}{\beta}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*\|^2 \\
&\quad - \beta\|\mathbf{y}^{k+1} - \mathbf{y}^*\|_{B^TB}^2 - 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 \\
&\quad - \frac{1}{\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|^2 - 2\mu_1\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2
\end{aligned} \tag{B.3.8}$$

Rearranging the items of the above inequality we have

$$\begin{aligned}
&\mu_1\left(2 + \frac{\mu_1}{2\beta\Lambda_{\max}(AA^T)}\right)\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \beta\|\mathbf{y}^{k+1} - \mathbf{y}^*\|_{B^TB}^2 + 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 + \frac{1}{\beta}\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|^2 \\
&\leq \beta\|\mathbf{y}^k - \mathbf{y}^*\|_{B^TB}^2 + \frac{1}{\beta}\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*\|^2 = \|\mathbf{w}^k - \mathbf{w}^*\|_M^2.
\end{aligned} \tag{B.3.9}$$

Next we bound the right hand side of the above inequality. Denote $\tau > 0$ such that

$$\tau\|\mathbf{w}^k - \mathbf{w}^*\|_M^2 \overset{(B.3.5)}{\leq} \tau\left[\frac{L_2^2}{\beta c} + \beta\Lambda_{\max}(B^T B)\right]\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 = 2\mu_2\|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2, \quad \text{(B.3.10)}$$

and the formula of $\tau$ (6.4.3) follows. Combing Lemma 1 and (B.3.10) we have

$$\|\mathbf{w}^k - \mathbf{w}^*\|_M^2 - \|\mathbf{w}^{k+1} - \mathbf{w}^*\|_M^2 \geq \tau\|\mathbf{w}^{k+1} - \mathbf{w}^*\|_M^2, \quad \text{(B.3.11)}$$

and together with (B.3.9) the linear rate follows. □

## *B.4 Proof for Theorem 2*

*Proof.* This result simply follows Lemma 1 and (B.3.10). □

# REFERENCES


[1] Agarwal, A. and Duchi, J., "Distributed delayed stochastic optimization," in *Advances in Neural Information Processing Systems 24*, pp. 873–881, 2011.

[2] Barbieri, N., Bonchi, F., and Manco, G., "Cascade-based community detection," in *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 33–42, ACM, 2013.

[3] Bertsekas, D. P., *Nonlinear Programming*. Athena Scientific, 2nd ed., 1999.

[4] Betensky, R. A. and Finkelstein, D. M., "A non-parametric maximum likelihood estimator for bivariate interval censored data," *Statistics in Medicine*, vol. 18, no. 22, pp. 3089–3100, 1999.

[5] Blundell, C., Heller, K. A., and Beck, J. M., "Modelling reciprocating relationships with hawkes processes.," in *NIPS*, pp. 2609–2617, 2012.

[6] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, 2010.

[7] Brown, E. N., Barbieri, R., Ventura, V., Kass, R. E., and Frank, L. M., "The time-rescaling theorem and its application to neural spike train data analysis," *Neural computation*, vol. 14, no. 2, pp. 325–346, 2002.

[8] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C., "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[9] Candès, E. J., Li, X., Ma, Y., and Wright, J., "Robust principal component analysis?," *J. ACM*, vol. 58, pp. 11:1–11:37, June 2011.

[10] Casella, G. and George, E. I., "Explaining the gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. 167–174, 1992.

[11] Chapelle, O., "Modeling delayed feedback in display advertising," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1097–1105, ACM, 2014.

[12] Chapelle, O. and others, "A simple and scalable response prediction for display advertising," 2013.

[13] Chen, M.-H., *Statistical analysis of multivariate interval-censored failure time data*. PhD thesis, University of Missouri–Columbia, 2007.

[14] Cho, Y.-S., Galstyan, A., Brantingham, J., and Tita, G., "Latent point process models for spatial-temporal networks," *arXiv preprint arXiv:1302.2671*, 2013.

[15] Cox, D. R., "Regression models and life-tables," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 187–220, 1972.

[16] Crane, R. and Sornette, D., "Robust dynamic classes revealed by measuring the response function of a social system," *Proceedings of the National Academy of Sciences*, vol. 105, no. 41, pp. 15649–15653, 2008.

[17] Daley, D. J. and Vere-Jones, D., *An introduction to the theory of point processes*, vol. 2. Springer, 1988.

[18] Daley, D. J. and Vere-Jones, D., *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*, vol. 1. Springer, 2007.

[19] Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L., "Optimal distributed online prediction," in *Proceedings of the 28th International Conference on Machine Learning*, pp. 713–720, June 2011.

[20] Deng, W. and Yin, W., "On the global and linear convergence of the generalized alternating direction method of multipliers," Tech. Rep. TR12-14, Rice University CAAM Technical Report, 2012.

[21] Deschatres, F. and Sornette, D., "Dynamics of book sales: Endogenous versus exogenous shocks in complex networks," *Physical Review E*, vol. 72, no. 1, p. 016112, 2005.

[22] Diggle, P., "A kernel method for smoothing point process data," *Applied Statistics*, pp. 138–147, 1985.

[23] Du, N., Song, L., Gomez-Rodriguez, M., and Zha, H., "Scalable influence estimation in continuous-time diffusion networks," *Advances in Neural Information Processing Systems*, pp. 3147–3155, 2013.

[24] DuBois, C., Butts, C., and Smyth, P., "Stochastic blockmodeling of relational event dynamics," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 238–246, 2013.

[25] Duchi, J., Agarwal, A., and Wainwright, M., "Distributed dual averaging in networks," in *Advances in Neural Information Processing Systems 23*, pp. 550–558, 2010.

[26] Duchi, J., Agarwal, A., and Wainwright, M. J., "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[27] Easley, D. and Kleinberg, J., "Networks, crowds, and markets," *Cambridge Univ Press*, vol. 6, no. 1, pp. 6–1, 2010.

[28] Eckstein, J. and Bertsekas, D. P., "On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.

[29] Embrechts, P., Liniger, T., and Lin, L., "Multivariate hawkes processes: an application to financial data," *Journal of Applied Probability*, vol. 48, pp. 367–378, 2011.

[30] Erlang, A. K., "The theory of probabilities and telephone conversations," *Nyt Tidsskrift for Matematik B*, vol. 20, no. 33-39, p. 16, 1909.

[31] Fan, C.-P. S., *Local Likelihood for Interval-censored and Aggregated Point Process Data*. PhD thesis, University of Toronto, 2009.

[32] Farajtabar, M., Du, N., Valera, I., Song, L., Rodrigez, M. G., and Zha, H., "Shaping social activity by incentivizing users," *arXiv preprint arXiv:1408.0406*, 2014.

[33] Gabay, D., "Applications of the method of multipliers to variational inequalities," North-Holland: Amsterdam, 1983.

[34] Gabay, D. and Mercier, B., "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, 1976.

[35] Girvan, M. and Newman, M. E., "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[36] Glowinski, R. and Tallec, P. L., *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*. Studies in Applied and Numerical Mathematics, SIAM, 1989.

[37] Goldfarb, D., Ma, S., and Scheinberg, K., "Fast alternating linearization methods for minimizing the sum of two convex functions," 2010.

[38] Gomez Rodriguez, M., Leskovec, J., and Krause, A., "Inferring networks of diffusion and influence," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1019–1028, ACM, 2010.

[39] Hawkes, A. G., "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[40] Hawkes, A. G. and Oakes, D., "A cluster process representation of a self-exciting process," *Journal of Applied Probability*, pp. 493–503, 1974.

[41] He, B. S., Yang, H., and Wang, S. L., "Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities," *Journal of Optimization Theory and Applications*, vol. 106, no. 2, 2000.

[42] He, B. and Yuan, X., "On non-ergodic convergence rate of douglas-rachford alternating direction method of multipliers," 2012.

[43] He, B. and Yuan, X., "On the $O(1/n)$ Convergence Rate of the Douglas-Rachford Alternating Direction Method," *SIAM J. Numer. Anal.*, vol. 50, no. 2, pp. 700–709, 2012.

[44] Hong, M. and Luo, Z.-Q., "On the linear convergence of the alternating direction method of multipliers." http://arxiv.org/abs/1208.3922, 2012.

[45] Hunter, D., Smyth, P., Vu, D. Q., and Asuncion, A. U., "Dynamic egocentric models for citation networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 857–864, 2011.

[46] Iwata, T., Shah, A., and Ghahramani, Z., "Discovering latent influence in online social activities via shared cascade poisson processes," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 266–274, ACM, 2013.

[47] Jewell, N. P., Malani, H. M., and Vittinghoff, E., "Nonparametric estimation for a form of doubly censored data, with application to two problems in aids," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 7–18, 1994.

[48] Johansen, A. and Sornette, D., "Download relaxation dynamics on the www following newspaper publication of url," *Physica A: Statistical Mechanics and its Applications*, vol. 276, no. 1, pp. 338–345, 2000.

[49] Kim, M. and Leskovec, J., "Nonparametric multi-group membership model for dynamic networks," *Advances in Neural Information Processing Systems*, pp. 1385–1393, 2013.

[50] Kingman, J. F. C., *Poisson processes*, vol. 3. Oxford university press, 1992.

[51] Kongerud, J. and Samuelsen, S. O., "A longitudinal study of respiratory symptoms in aluminum potroom workers1-3," *The American review of respiratory disease*, vol. 144, p. 10, 1991.

[52] Lam, K., Xu, Y., and Cheung, T.-L., "A multiple imputation approach for clustered interval-censored survival data," *Statistics in medicine*, vol. 29, no. 6, pp. 680–693, 2010.

[53] Levine, R. A. and Casella, G., "Implementations of the monte carlo em algorithm," *Journal of Computational and Graphical Statistics*, vol. 10, no. 3, pp. 422–439, 2001.

[54] Lewis, E., Mohler, G., Brantingham, P. J., and Bertozzi, A. L., "Self-exciting point process models of civilian deaths in iraq," *Security Journal*, vol. 25, no. 3, pp. 244–264, 2011.

[55] Li, L. and Zha, H., "Dyadic event attribution in social networks with mixtures of hawkes processes," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 1667–1672, ACM, 2013.

[56] Li, X.-L., Tan, A., Philip, S. Y., and Ng, S.-K., "Ecode: event-based community detection from social networks," in *Database Systems for Advanced Applications*, pp. 22–37, Springer, 2011.

[57] Linderman, S. W. and Adams, R. P., "Discovering latent network structure in point process data," *arXiv preprint arXiv:1402.0914*, 2014.

[58] Liu, W., Pellegrini, M., and Wang, X., "Detecting communities based on network topology," *Scientific reports*, vol. 4, 2014.

[59] Marsan, D. and Lengline, O., "Extending earthquakes' reach through cascading," *Science*, vol. 319, no. 5866, pp. 1076–1079, 2008.

[60] Mateos, G., Bazerque, J. A., and Giannakis, G. B., "Distributed sparse linear regression," *IEEE Trans on Signal Processing*, vol. 58, pp. 5262 –5276, oct. 2010.

[61] Mitchell, L. and Cates, M. E., "Hawkes process as a model of social interactions: a view on video dynamics," *Journal of Physics A: Mathematical and Theoretical*, vol. 43, no. 4, p. 045101, 2010.

[62] Monteiro, R. D. C. and Svaiter, B. F., "Iteration-complexity of block-decomposition algorithms and the alternating minimization augmented lagrangian method," tech. rep., Georgia Institute of Technology, 2010.

[63] Mota, J. F. C., Xavier, J. M. F., Aguiar, P. M. Q., and Puschel, M., "D-admm: A communication-efficient distributed algorithm for separable optimization." arXiv:1202.2805, 2012.

[64] Nedic, A. and Ozdaglar, A., "Distributed subgradient methods for multi-agent optimization," *IEEE Trans on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[65] Ng, A. Y., Jordan, M. I., and Weiss, Y., "On spectral clustering1 analysis and an algorithm," *Proceedings of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press*, vol. 14, pp. 849–856, 2001.

[66] Ogata, Y., "Statistical models for earthquake occurrences and residual analysis for point processes," *Journal of the American Statistical Association*, vol. 83, no. 401, pp. 9–27, 1988.

[67] Ouyang, H., He, N., Tran, L. Q., and Gray, A., "Stochastic alternating direction method of multipliers," in *Proceedings of the 30th Intl. Conference on Machine Learning. JMLR: W&CP volume 28.*, 2013.

[68] Perry, P. O. and Wolfe, P. J., "Point process modelling for directed interaction networks," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 5, pp. 821–849, 2013.

[69] Rubin, D. B., *Multiple imputation for nonresponse in surveys*, vol. 307. Wiley. com, 2009.

[70] Stomakhin, A., Short, M. B., and Bertozzi, A. L., "Reconstruction of missing data in social networks based on temporal patterns of interactions," *Inverse Problems*, vol. 27, no. 11, p. 115013, 2011.

[71] Streit, R. L., *Poisson Point Processes: Imaging, Tracking, and Sensing*. Springer, 2010.

[72] Sun, J. and Kalbfleisch, J., "Estimation of the mean function of point processes based on panel count data," *Statistica Sinica*, vol. 5, pp. 279–290, 1995.

[73] Tanner, M. A. and Wong, W. H., "An application of imputation to an estimation problem in grouped lifetime analysis," *Technometrics*, vol. 29, no. 1, pp. 23–32, 1987.

[74] Wang, S. L. and Liao, L. Z., "Decomposition method with a variable parameter for a class of monotone variational inequality problems," *Journal of Optimization Theory and Applications*, vol. 109, no. 2, pp. 415–429, 2001.

[75] Wei, G. C. and Tanner, M. A., "A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 699–704, 1990.

[76] Wellner, J. A. and Zhang, Y., "Two estimators of the mean of a counting process with panel count data," *Annals of Statistics*, pp. 779–814, 2000.

[77] Winn, J., Bishop, C. M., and Jaakkola, T., "Variational message passing.," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[78] Wong, G. Y. and Yu, Q., "Generalized mle of a joint distribution function with multivariate interval-censored data," *Journal of Multivariate Analysis*, vol. 69, no. 2, pp. 155–166, 1999.

[79] Wu, C. J., "On the convergence properties of the em algorithm," *The Annals of statistics*, pp. 95–103, 1983.

[80] Xun, G., Yang, Y., Wang, L., and Liu, W., "Latent community discovery with network regularization for core actors clustering.," *COLING (Posters)*, pp. 1351–1360, 2012.

[81] Yang, J., McAuley, J., and Leskovec, J., "Community detection in networks with node attributes," *arXiv preprint arXiv:1401.7267*, 2014.

[82] Yang, S.-H. and Zha, H., "Mixture of mutually exciting processes for viral diffusion," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1–9, 2013.

[83] Zachary, W. W., "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, pp. 452–473, 1977.

[84] Zhang, X., Burger, M., and Osher, S., "A unified primal-dual algorithm framework based on bregman iteration," *J. of Scientific Computing*, vol. 46, no. 1, pp. 20–46, 2011.

[85] Zhou, D., Councill, I., Zha, H., and Giles, C. L., "Discovering temporal communities from social network documents," in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 745–750, IEEE, 2007.

[86] Zhou, D., Manavoglu, E., Li, J., Giles, C. L., and Zha, H., "Probabilistic models for discovering e-communities," in *Proceedings of the 15th international conference on World Wide Web*, pp. 173–182, ACM, 2006.

[87] Zhou, K., Zha, H., and Song, L., "Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 641–649, 2013.

[88] Zhu, H., Giannakis, G. B., and Cano, A., "Distributed in-network channel decoding," *IEEE Trans on Signal Processing*, vol. 57, no. 10, pp. 3970 –3983, 2009.

# VITA

Long Quoc Tran was born in Hanoi, Vietnam. After completing his schoolwork at HUS High School for Gifted Students in 1998, Long entered Vietnam National University at Hanoi. He received a Bachelor of Science with major in Communication Technology in June 2002. From 2003 to 2005, he studied at National University of Singapore and received a Master of Engineering with major in Computer Engineering in May 2005. During the following two years, he was employed as a lecturer in the Vietnam National University at Hanoi. In August 2007, Long entered the PhD in Computer Science program in the College of Computing, Georgia Institute of Technology in Atlanta, Georgia.