MODELING, PREDICTING, AND GUIDING USERS' TEMPORAL BEHAVIORS

A Dissertation
Presented to
The Academic Faculty

By

Yichen Wang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology

August 2018

MODELING, PREDICTING, AND GUIDING USERS' TEMPORAL BEHAVIORS

Approved by:

Dr. Le Song, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Hongyuan Zha
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Haomin Zhou
School of Mathematics
Georgia Institute of Technology

Dr. Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Xiaojing Ye
School of Mathematics
Georgia State University

Date Approved: May 02, 2018

To my parents:

Ping Chen and Qi Wang

and to my wife:

Hemochen An

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my advisor Dr. Le Song for providing me an opportunity to work in his lab and continuously supporting my study and related research. His inspiring thought, insightful guidance, and kind encouragement truly help me move on the right track these years. Without his his extraordinary supervision, I could not achieve what I have currently.

My research and growth in machine learning also benefit significantly from interacting with a marvelous group of coauthors and collaborators. I would like to express my sincere appreciation to each of them: Hanjun Dai, Nan Du, Mehrdad Farajtabar, Aditya Pal, Vasilis Syrgkanis, Evangelos Theodorou, Rakshit Trivedi, Apurv Verma, Grady Williams, Bo Xie, Xiaojing Ye, Hongyuan Zha, Haomin Zhou, etc. I am especially thankful to Hongyuan for his generous help and career advice. I would like to thank Nan, Bo, Hanjun, Mehrdad, and Rakshit for active discussions and close collaborations. I have learned a lot from Nan and Bo. Thank you for the kind help during our collaboration and your suggestions in both research topic and career path when I was a junior student. I also thank Hanjun and Rakshit for the timely and strong support whenever I need help. I have great memories and experiences at Microsoft Research and IBM Research. I would like to thank Vasilis and Aditya for brining me to a variety of interesting fields and help me broaden the scope of my research.

I would also like to express my gratitude to my committee members, Hongyuan Zha, Haomin Zhou, Mark Davenport, and Xiaojing Ye for their valuable suggestions and constructive feedbacks on my dissertation.

Last but not the least, I feel deeply indebted to the love and support of my parents. My wife, Hemochen, I always thank you for supporting me and for believing in me unconditionally. I love you very much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Millions of people around the world generate large-scale temporal behavior data from various domains, such as social platforms, economic systems, healthcare systems, and online service websites. This user generated data provides great opportunities and challenges to efficiently study the huge volumes of data, and design mathematical modeling frameworks to properly analyze the complex behavior of users, and to apply the resulting insights in the real world. In our work, we propose a systematic paradigm for designing machine learning models and algorithms to improve the understanding of users' temporal behaviors. With this novel framework, we contribute in the following three aspects:

Expressive machine learning models. We first propose a non-parametric point process model to capture the various user behavior patterns in service platforms and social networks. We develop a robust learning algorithms, despite the non-convexity of the learning problem, we are able to obtain theoretical guarantees on the generalization error. We then propose a co-evolutionary latent feature processes model to capture the evolving features of users and items in online service platforms. We propose an efficient non-negative matrix rank minimization algorithm, which elegantly inherits the advantages from both proximal methods and conditional gradient methods to solve the matrix rank minimization problem under non-negative constraints. We also build the link between recurrent neural network and the point processes to learn a nonlinear embedding of users and items latent features. Finally, we are the first to establish a previously unexplored connection between point processes and stochastic differential equations, which opens the gate to bring the techniques in stochastic optimal control theory to guiding users' temporal behaviors.

Scalable predictive algorithms. In addition to designing models to study the microscopic pattern of users' behaviors, we develop scalable predictive algorithms for

macroscopic and collective user activity levels, *e.g.*, expected total number of active users during a period) by exploiting these models. We develop a unified framework for predicting user activity levels in point processes. We derive a differential-difference equation to compute a conditional probability mass function for point processes, which is applicable to general point processes and prediction tasks. Our framework provides an unbiased estimator of the probability mass function of point processes and significantly reduces the required sample size compared with the Monte Carlo method.

Efficient optimal control algorithms. Besides understanding and predicting users' temporal behaviors, it is also important to design "closed loop" systems that can incorporate users' feedback adaptively and help users make better strategic decisions. However, existing works are "open loop": they design policies before the behavior process unfolds and do not consider users' feedback. We develop an efficient framework to incorporate users' feedback when designing optimal policies. Specifically, we extend the classic control theories, such as Hamilton Jacobian Bellman equation to point processes. We also propose a novel intensity optimal control problem, and further design a novel measure-theoretic framework to compute the optimal policies.

Keywords: Point processes, Hawkes processes, Survival analysis, Low-rank models, Mass transport, Fokker Planck equation, Stochastic optimal control, Reinforcement learning, Social network analysis, Information diffusion, Recommendation systems

CHAPTER 1
INTRODUCTION

Internet is transforming our lives: now people are relying increasingly on Internet to communicate with their friends, participate in discussions, place advertisements, make purchases, and stay in touch with the world. Millions of people around the world generate large-scale temporal behavior data from various domains, such as social platforms, economic systems, healthcare systems, and online service websites. As life becomes increasingly digital — behaviors, opinions, interactions, and decisions are represented as bits, sets, events, and time series — the key challenge to tackle is: **How to make these digital platforms more useful and engaging for online users and the entire society?**

To tackle this challenge, not only computational techniques are required to efficiently study the huge volumes of data, but mathematical modeling frameworks are necessary to properly analyze the complex behavior of users, and to apply the resulting insights in the real world. Despite their importance, rigorous principles to distill knowledge from these behavior data is still lacking.

For a long time, epoch based methods have been applied to analyze their behaviors. Users' temporal behaviors are *asynchronous*, *interdependent*, and contain high-dimensional information about location, time, and text. However, these models discretize time into equally spaced intervals, ignore the asynchronous nature of users' behaviors, and are not able to answer fine-grained time-sensitive queries. Hence it is urgent to develop new and realistic representations to model users' behaviors in a principled way. To fulfill this need, **I design machine learning models and algorithms to improve the understanding of users' temporal behaviors**.

My research strives to understand why users behaves in a particular way and pro-

vide better insights in decision making. I take a bottom-up approach which starts by considering the mechanism driving the stochastic behavior of each user to later produce macroscopic patterns. There are three fundamental problems in understanding users' behaviors. First, to understand "*what is the pattern of a user's temporal behavior?*", I design expressive point process models for users' behaviors [1, 2, 3, 4, 5]. Second, to understand "*what will users do by when and where?*", I develop scalable predictive algorithms to infer users' future behaviors by exploiting the learned models [6, 7]. Finally, to understand "*what is the optimal policy for a user when making decisions?*", I develop new reinforcement learning algorithms to help users make optimal decisions [8, 9]. In what follows, I will describe several challenging research problems and how my research develops rigorous principles for these research topics.

1.1 Research Problems

We investigate the following fundamental research problems.

Non-parametric learning of point processes. Hawkes processes are powerful tools for modeling the mutual-excitation phenomena commonly observed in event data from a variety of domains, such as social networks, quantitative finance, and healthcare records. The intensity function of a Hawkes process is typically assumed to be linear in the sum of triggering kernels, rendering it inadequate to capture nonlinear effects present in real-world data. For example, after purchasing a new album, users may be initially highly engaged, and play the album over and over again. However, the engagement will saturate at some point as they become bored of the same album. Such plateau pattern may not be captured by a simple linear relation. In another scenario, a recent hospital visit may trigger more future visits due to the progression of a disease into a more severe stage. Such cumulative influence from recent events may grow faster than a linear model can explain. Hence, the first question that we aim at addressing is:

Can we design an expressive point process model which can capture various users' temporal behaviors?

To address this problem, in Chapter 2 we propose a novel point process — the Isotonic-Hawkes Process, whose intensity function is modulated by an additional nonlinear link function. We also developed a novel iterative algorithm which learns both the nonlinear link function and other parameters provably. We showed that Isotonic-Hawkes processes can fit a variety of nonlinear patterns which cannot be captured by conventional Hawkes processes, and achieve superior empirical performance in real world applications.

Time sensitive recommendation and continuous-time evolving graphs. Matching users to the right items at the right time is a fundamental task in recommendation systems. As users interact with different items over time, users' and items' feature may evolve and co-evolve over time. Traditional models based on static latent features or discretizing time into epochs can become ineffective for capturing the fine-grained temporal dynamics in the user-item interactions. Both users' interests and items' semantic features are dynamic and can evolve over time [10, 11]. The interactions between users and service items play a critical role in driving the evolution of user interests and item features. Based on the user-item interaction data, we are interested in the following question:

Can we design an efficient framework that recommends the most desirable item to a user at the right time? Can we predict when a user would return to the existing service in the future?

In Chapter 3, we propose a coevolutionary latent feature process model that accurately captures the coevolving nature of users' and items' feature. In Chapter 4, we further extend our framework with recurrent neural networks, and develop a generic embedding framework for continuous-time evolving graphs.

Scalable prediction for macroscopic activity levels. In addition to designing

models to study the microscopic pattern of users' behaviors, I develop scalable predictive algorithms for macroscopic and collective user activity levels (e.g. expected total number of active users during a period) by exploiting these models. A framework for doing this is critically important. For example, for online merchants such as Amazon, an accurate estimate of the number of future purchases of a product helps optimizing future advertisement placements. However, existing approaches either use expensive Monte Carlo simulations requiring a large number of samples, or neglect part of the stochasticity of the stochastic processes, leading to severely biased estimators. Hence a central question is:

Can we design an efficient predictive algorithm that can estimate the probability mass function of general point processes?

In Chapter 5, we propose a framework that provides an efficient unbiased estimator for point processes. In particular, we design a key reformulation of the prediction problem, and further derive a differential-difference equation to compute a conditional probability mass function.

Optimal policies for guiding systems driven by point processes. We have developed point process models for learning and predicting users' temporal behaviors. However, these works focus on the "open loop" setting where learned models are used for predictive tasks. Typically, we are interested in the "closed loop" setting where a policy needs to be learned to incorporate user feedbacks and guide user activities to desirable states. Although point processes have good predictive performance, it is not clear how to use them for the challenging closed loop activity guiding task. Hence, a key question is:

Can we design efficient policy to control the drift part of the system that is driven by point processes, such that the system is steered to a target state?

In Chapter 6, we propose a generic framework to reformulate point processes into stochastic differential equations (SDEs), which allows us to extend methods from

stochastic optimal control to address the activity guiding problem. We also design an efficient algorithm that controls the drift part of the SDE system, and show that our method guides user activities to desired states more effectively than the state of the art.

Optimal policies for guiding point processes. Besides guiding users' behaviors by influencing drift part of the behavior system, we are also interested in controlling these point processes to influence user behaviors. A framework for doing this is critically important. For example, government agents may want to effectively suppress the spread of terrorist propaganda, which is important for understanding the vulnerabilities of social networks and increasing their resilience to rumor and false information; to gain more attention, a broadcaster on Twitter may want to design a smart tweeting strategy such that his posts always remain on top of his followers' feeds. Hence a we propose a novel question:

Can we design efficient policy to control the intensity of point processes, such that the stochastic system driven by the point process is steered to a target state?

In Chapter 7, we exploit the key insight to view this intensity stochastic optimal control problem from the perspective of finding the optimal measure and variational inference. We further propose a convex optimization framework and an efficient algorithm to update the policy adaptively to the current system state.

1.2 Contributions and Organization

To solve these research problems, throughout this dissertation, we mainly take the following three aspects into consideration:

Expressive models and efficient learning algorithms. We propose simple probabilistic models with rigorous mathematical specifications, and accurate modeling of event data in social platforms. We also develop efficient learning algorithms for fitting the proposed models to large scale datasets. In Chapter 2, we first propose a

non-parametric point process model to capture the various user behavior patterns in service platforms and social networks. We develop a robust learning algorithms, despite the non-convexity of the learning problem, we are able to obtain theoretical guarantees on the generalization error. In Chapter 3, we then propose a co-evolutionary latent feature processes model to capture the evolving features of users and items in online service platforms. We propose an efficient non-negative matrix rank minimization algorithm, which elegantly inherits the advantages from both proximal methods and conditional gradient methods to solve the matrix rank minimization problem under non-negative constraints. In Chapter 4, we also build the link between recurrent neural network and the point processes to learn a nonlinear embedding of users latent features in continuous-time evolving graphs.

Scalable predictive algorithms. Using the models that explain the data, we design scalable new inference algorithms to make future predictions of users's macroscopic activity levels. In Chapter 5, we develop a unified framework for predicting user activity levels in point processes. We derive a differential-difference equation to compute a conditional probability mass function for point processes. It is applicable to general point processes and prediction tasks. Our framework provides an unbiased estimator of the probability mass function of point processes and significantly reduces the sample size compared with the Monte Carlo method.

Efficient optimal control algorithms. We are the first to establish a previously unexplored connection between point processes and stochastic differential equations (SDEs), which significantly generalizes existing models, and opens the gate to bring the techniques in stochastic optimal control theory to guiding users' temporal behaviors. We develop an efficient framework to incorporate users' feedback when designing optimal policies to control the SDE system. We study two aspects of the control problem. In Chapter 6, we study the problem of controlling the drift part of the behavior system, and extend the classic theories such as Hamilton Jacobian Bellman equation

to point processes. In Chapter 7, we propose a novel intensity optimal control problem that directly controls the jump part of the system, and further design a novel measure-theoretic algorithm to compute the optimal policies.

1.3 Literature Survey

Time series models. Sequence data modeling is a classic topic in machine learning and data mining. Many time series models, such as Markov chain [12], Hidden Markov Models [13], Kalman Filters [14], Vector Auto Regressive models [15], are widely used to study sequential event data. However, one major limit of these models is that the time is discretized into equally spaced epochs. Since the system evolves in a synchronized step-by-step way, these models are not expressive enough to capture the asynchronous characteristics of users' temporal behaviors. Another limitation is that the epoch length, as a hyper-parameter, is typically difficult to tune. Moreover, we need to aggregate many samples in one epoch and deal with the case when there is no data point in one epoch. The Semi-Markov model [16] seeks to solve this issue by explicitly modeling the transition time between two states as continuous exponential random variables. However, such strong model assumption has limited the applicability of these models. Finally, most of the existing time series models consider low dimensional data, which are not applicable to model users' temporal behaviors that involve high dimensional features and complex interdependencies.

Epoch based behavior modeling. In the area of social network analysis, [17] studied the problem of predicting links between users using structural properties of the networks. [18] further used the rich contextual features in the networks to solve a binary classification problem. The limitation in these works is that the link prediction is independent among users and the global information of interdependency among users are not considered. Recently, there is a growing interest in studying the dynamic interaction among users. [19] studied the relational evens in the longitudinal

social networks, [20] proposed to model the group dynamics in networks, [21, 22] studied the macroscopic online user activities using tensor methods. [23] proposed generative methods to study information cascades. However, most of these models still reply on the epoch assumption and treat time as discrete indices. Recently, [24] proposed continuous time Markov processes to model friendship dynamics. However, the restricted assumption of these models limit their ability to capture the influence of historical events as well as users' interactions.

Point processes. Point processes treat time as a random variable and have been widely applied in modeling users' temporal behaviors in different applications, such as criminal analysis [25, 26, 27], information diffusion [28, 29], vision perception modeling [30], recommender systems [5], and social networks [31, 32, 33, 34]. However, these works typically do not model the interdependency among users and can only be applied to small dimensions. To solve these issues, we aim at developing a generic point process based framework that is applicable to model the asynchronous, interdependent, and high dimensional event data.

1.4 Background on Point Processes

A temporal point process is a random process whose realization consists of a list of discrete events localized in time, $\{t_i\}$ with $t_i \in \mathbb{R}$ and $i \in \mathbb{Z}$. Let the history $\mathcal{H}_{t^-}$ be the list of event time $\{t_1, t_2, \cdots, t_n\}$ up to but not including time t. The lengths of the time intervals between neighboring successive events are referred to as the inter-event times. Given the history of past events, we can explicitly specify the conditional density function that the next event will happen at time t as $f(t) := f(t|\mathcal{H}_{t^-})$. Then the joint density of observing the sequence of events is

$$f(\{t_i\}_{i=1}^n) = \prod_{i=1}^n f(t_i)$$

A temporal point process can be equivalent represented as a counting process, $N(t)$, which records the number of events before time t. Then in a small time window $\mathrm{d}t$ between $[0, t)$, the number of observed event is

$$\mathrm{d}N(t) = \sum_{t_i \in \mathcal{H}_{t^-}} \delta(t - t_i)\mathrm{d}t$$

and hence $N(t) = \int_0^t \mathrm{d}N(s)$. It is often assumed that only one event can happen in a small window of size $\mathrm{d}t$, and hence $\mathrm{d}N(t) \in \{0, 1\}$.

An important way to characterize temporal point processes is visa the conditional intensity function – the stochastic model for the next event time given all previous events. Within a small window $[t, t + \mathrm{d}t)$, $\lambda(t)\mathrm{d}t$ is the probability for the occurrence of new event given the history $\mathcal{H}_{t^-}$:

$$\lambda(t)\mathrm{d}t = \mathbb{P}[\text{event in } [t, t + \mathrm{d}t)|\mathcal{H}_{t^-}]$$

Intuitively, this intensity function denotes the hazard rate that an event can happen at time t.

Given a sequence of events $\mathcal{H}_{t^-} = \{t_1, \ldots, t_n\}$, for any $t > t_n$, we characterize the conditional probability (survival function) that no event happens during $[t_n, t)$ as

$$S(t|\mathcal{H}_{t^-}) = \exp\left(-\int_{t_n}^{t} \lambda(\tau)\, d\tau\right)$$

We can also characterize the conditional density $f(t|\mathcal{T})$ that an event occurs at time t as

$$f(t|\mathcal{H}_{t^-}) = \lambda(t)\, S(t|\mathcal{H}_{t^-})$$

The function forms of $\lambda(t)$ are often designed to capture the phenomena fo interests. Next, we discuss several classic models

- Poisson processes. It is the simplest point process model, whose intensity function is a constant, *i.e.*, $\lambda(t) = \mu$. Hence the generation of future event does not depend on the historical events. This assumption is too strong and the Poisson process model is not able to capture the influence of historical evnets.

- Hawkes processes. This model captures the mutual excitation phenomena between events, and its intensity function is defined as

$$\lambda(t) = \lambda_0 + \alpha \sum\nolimits_{t_i \in \mathcal{H}_{t^-}} \kappa(t - t_i) \tag{1.1}$$

  where λ_0 captures the long-term incentive to generate events. $\kappa(t) \geqslant 0$ models temporal dependencies, and $\alpha \geqslant 0$ quantifies how the influence from each past event evolves over time, making the intensity function depend on the history $\mathcal{H}_{t^-}$. This model has been applied successfully in modeling criminal retaliations [26], online users behaviors [35, 4, 5], and opinion dynamics [9]. In the Hawkes process, past events affect the occurrence of future events, which makes it particularly useful for modeling clustered event patterns. However, the linear link function of the intensity function may be insufficient to model many real world scenarios.

CHAPTER 2
NON-PARAMETRIC LEARNING OF POINT PROCESS MODELS

2.1 Introduction

Temporal point processes are powerful tools for modeling the complex dynamics of events occurrences. In particular, Hawkes processes [36] are well-suited to capture the phenomenon of mutual excitation between the occurrence of events, and have been applied successfully in modeling criminal retaliations [26], online users behaviors [35, 4, 5], and opinion dynamics [9].

A Hawkes process is characterized by a *linear* intensity function, *i.e.*, $\lambda(t) = w \cdot x_t$ where x_t denotes time-dependent features and w is the weight. The intensity function parametrizes the likelihood of observing an event in the time window $[t, t + \mathrm{d}t)$ given that it has not occurred before t. Such linearity may be insufficient to model many real world scenarios. For example, after purchasing a new album, users may be initially highly engaged, and play the album over and over again. However, the engagement will saturate at some point as they become bored of the same album. Such plateau pattern may not be captured by a simple linear relation. In another scenario, a recent hospital visit may trigger more future visits due to the progression of a disease into a more severe stage. Such cumulative influence from recent events may grow faster than a linear model can explain.

Nonlinear Hawkes process [37] has been introduced to provide more flexibility in explaining the real-world phenomena. It applies a fixed nonlinear link function g to the linear combination, *i.e.*, $\lambda(t) = g(w \cdot x_t)$. For computational considerations, $g(\cdot)$ is often assumed to be in some simple parametric forms, such as $\exp(u)$ and $\max(0, u)$ [38, 39]. Although these models are more flexible, they are still restricted

to a few nonlinear patterns with a fixed parametrization, which may not be correct for real world data. Ideally, both $g(\cdot)$ and w should be learned from data. Unfortunately, such desideratum leads to a *non-convex* optimization problem, where efficient algorithms with provable guarantees do not exist.

To address these challenges, we propose a novel model, referred to as the *Isotonic-Hawkes* process, where both $g(\cdot)$ and w can be directly learned from data. Rather than committing to a fixed parametric form, we instead use a non-parametric, monotonic nonlinear link function. Therefore, it is extremely flexible to capture different temporal dynamics without the need to select a fixed form in advance.

To solve the non-convex learning problem with guarantees, we propose a different loss function than the typical log-likelihood for point processes. Moreover, by exploiting the problem structure, we are still able to provide theoretical guarantees on the computational and statistical performance. Our work makes the following contributions:

- We propose a novel method for nonlinear Hawkes process that can learn both the link function and other parameters directly from data.
- Although the learning involves a *non-convex* problem, our algorithm can provably recover the true link function and the model parameters. This also requires a novel analysis for non *i.i.d.* observations.
- Our method achieves superior empirical performance, significantly outperforming alternatives on both synthetic and real-world datasets.

Related work. Prior work on nonlinear Hawkes process focuses on theoretical properties [37, 40, 39]. The link function is usually given, and the discretization of time is needed in order to evaluate the integral of the intensity function. Hence, efficient algorithms are available only for specific link functions [41, 42, 38]. In contrast, our method is the first algorithm that can learn both the link function and the model parameters non-parametrically.

Our work is also closely related to Isotonic regression and Single Index Model (SIM). The Isotonic regression [43, 44, 45] is a well studied technique to fit an arbitrary monotonic 1-D function. SIM generalizes the linear regression and estimates both the nonlinear link function and the feature weights. However, earlier work are usually heuristics, which are not guaranteed to converge to a global optimum [46, 47, 48, 49]. Only recently algorithms have been proposed with global convergence guarantees [50, 51, 52].

Unlike SIM, which only focuses on regression, our work is concerned with learning a temporal point process where the response variable is not directly observed. At the same time, the observations are non *i.i.d.* , a setting significantly different from previous works. The added complexity of temporal point processes requires us to develop a new efficient algorithm and its analysis.

2.2 Isotonic Hawkes Processes

We propose a new family of nonlinear Hawkes processes: Isotonic-Hawkes processes. We present the moment matching learning framework for the non-convex problem. To facilitate learning, we optimize the representation of the objective function by showing that the intensity integral in the objective function can be exactly computed. Then we present the overall algorithm, which applies an alternating minimization scheme to update the link function g and weights parameters w.

2.2.1 Model Formulation

In Isotonic-Hawkes processes, we model its intensity as the composition of a monotonic, non-parametric link function and a linear Hawkes intensity.

Definition 1. *A Isotonic-Hawkes process is a counting process $N(t)$, with associated*

history $\mathcal{H}_t = \{t_i | t_i < t\}$*, such that the intensity function* $\lambda(t)$ *can be written as:*

$$\lambda(t) = g\left(\lambda_0 + \alpha \sum_{t_i \in \mathcal{H}_t} \kappa(t - t_i)\right) = g(w \cdot x_t) \tag{2.1}$$

where $\kappa(t) : \mathbb{R}^+ \to \mathbb{R}^+$ *is a continuous monotonic decreasing triggering kernel capturing the influence of the history* $\mathcal{H}_t$*,* $\lambda_0 \geqslant 0$ *is a baseline intensity independent of the history, and* $g \in \mathcal{G} : \mathbb{R} \to \mathbb{R}^+$*, is a monotonic increasing and G-Lipschitz link function,* i.e.*,*

$$0 \leqslant g(b) - g(a) \leqslant G(b - a) \text{ for all } 0 \leqslant a \leqslant b \tag{2.2}$$

We set $w = (\lambda_0, \alpha)^\top$*, and* $x_t = \left(1, \sum_{t_i \in \mathcal{H}_t} \kappa(t - t_i)\right)^\top$*.*

We require $\kappa(t)$ to be monotonically decreasing, such as the exponential kernel $\exp(-t)\mathbb{I}[t > 0]$, Gaussian kernel and heavy tailed log-logistic kernel. This property is useful for computing the integral of the intensity discussed later.

The linear term in Hawkes process alone is not sufficient to capture the general trend in real-world applications. For instance, linearity leads to unbounded intensity, which is at odds with the saturation phenomenon. The nonlinear link function g enables the model to adapt to such nonlinearities in the data, hence achieving better performance. We assume g is nonparametric and monotonic increasing, which covers a wide range of functions, and also maintains the properties of the composed intensity function.

2.2.2 Moment Matching Objective

Maximum Likelihood Estimation (MLE) is often used to learn Hawkes processes, yielding a convex problem w.r.t. w. The estimator has good statistical rates and is consistent and asymptotically efficient [53]. However, if we want to learn $g(\cdot)$ and w jointly, MLE becomes non-convex, and we no longer have statistical guarantees. To solve this problem, we use a different learning criteria based on the moment matching

idea [54]. We can establish global convergence guarantees despite the non-convexity by establishing the connections to Isotonic regression and SIM.

Let $N_i = N(t_i)$. Since $N(t)$ is a counting process, the count increases by one at each time t_i. Hence for a list of events $\{t_i\}_{i=1}^n$, we have $N_i = i$ for $i \in [n]$. We have

$$\mathbb{E}[N_i|\mathcal{H}_{t_i}] = \int_0^{t_i} \lambda(t)\mathrm{d}t = \int_0^{t_i} g(w \cdot x_t)\mathrm{d}t. \tag{2.3}$$

Therefore, we can estimate the parameters g and w by matching the integral $\int_0^{t_i} \lambda(t)\mathrm{d}t$ with observations N_i, which leads to the following objective function:

$$\min_{g\in\mathcal{G},w} \frac{1}{n}\sum_{i=1}^{n}\left(N_i - \int_0^{t_i} g(w \cdot x_t)\mathrm{d}t\right)^2. \tag{2.4}$$

Note that we need to optimize an integral w.r.t. a function g, which is challenging in representation and computation. Instead of optimizing over $\mathcal{G}$, we replace it with the family of *piecewise constant non-decreasing functions, $\mathcal{F}$, and the jumps of g is defined only at the intensity of each observed event.* As shown in Theorem 2, the integral of such functions can be computed *exactly* as weighted combinations of $g(w \cdot x_{t_i})$ defined on the observed time points t_i. For notation simplicity, we set $x_i = x_{t_i}$. The piecewise-constant function will provide a good approximation to the original function as we increase the number of training samples.

2.2.3 Integral Computation

We assume g is a piecewise constant function defined on each time t_i. Then we have the following result:

Theorem 2. *Assume g is piecewise constant, then the integral on $[0, t_i]$ is a weighted sum of $y_j = g(w \cdot x_j)$ with weights a_{ij}. That is, $\int_0^{t_i} g(w \cdot x_t)\mathrm{d}t = \sum_{j\in\mathcal{S}_i} a_{ij}y_j$.*

To efficiently compute the a_{ij}'s, we can first compute the integral on intervals $[t_{i-1}, t_i]$, then use cumulative sum to arrive at the final results.

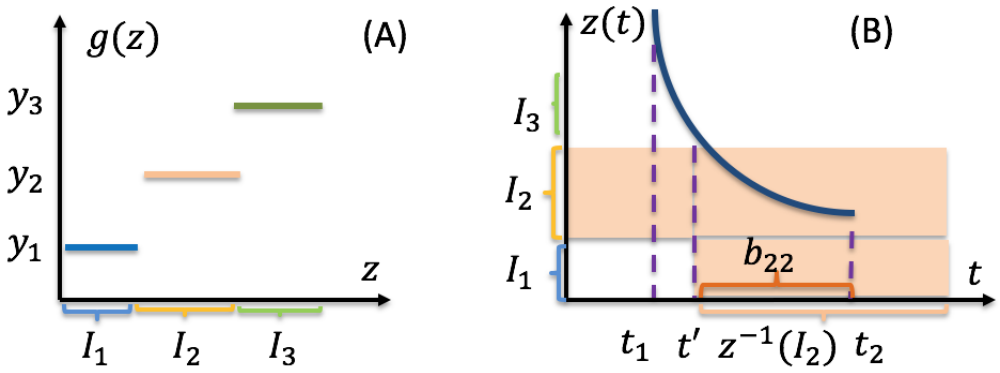


Figure 2.1: Illustration of integral computation. (A) the function g has 3 pieces and is constant on intervals I_1, I_2 and I_3. (B) The function $z(t) = w \cdot x_t$ is restricted on the interval $[t_1, t_2]$. It is continuous and monotonic decreasing due to the property of triggering kernel κ. The pre-image of I_2 is shown as the light yellow area on the t axis, and b_{22} is the intersection of $[t_1, t_2]$ and $z^{-1}(I_2)$. It is found by locating the pre-image of the endpoints, t' and another point outside the interval $[t_1, t_2]$ (not shown here).

Set $z(t) = w \cdot x_t$, since $g(\cdot)$ is a piecewise constant function, we have:

$$g(z(t)) = \sum_{j=1}^{n} y_j \mathbb{I}\left[z(t) \in I_j\right]$$

where $\mathbb{I}\left[\cdot\right]$ is the indicator function, and I_j denotes the j-th interval where $g(\cdot)$ is a constant. Therefore, we can write the integral on $[t_{i-1}, t_i]$ as:

$$\int_{t_{i-1}}^{t_i} g\big(z(t)\big)\mathrm{d}t = \sum_{j=1}^{n} y_j \int_{t_{i-1}}^{t_i} \mathbb{I}\left[t \in z^{-1}(I_j)\right] \mathrm{d}t$$

where $z^{-1}(I_j)$ denotes the pre-image of the interval I_j. Next, we need to compute $b_{ij} := \int_{t_{i-1}}^{t_i} \mathbb{I}\left[t \in z^{-1}(I_j)\right] \mathrm{d}t$. Since it is the length of the intersection of two intervals, we can compute b_{ij} by finding all the endpoints of the pre-images $z^{-1}(I_j)$.

To do this, we first state a property of $z^{-1}(I_j)$. Restricted on $[t_{i-1}, t_i]$, $z(t)$ is a continuous and monotonic decreasing function due to the monotonic decreasing triggering kernel κ (Figure 2.1(B)). Combined with the fact that I_j are disjoint and share endpoints (Figure 2.1(A)), the pre-images $z^{-1}(I_j)$ are also *disjoint and share endpoints*.

Algorithm 1 COMPUTE-COEFFICIENT

Input: t_i, for $i = 1, \cdots, n$
Output: a_{ij}
for $j = 1, \cdots, n$ **do**
    Compute t'_j that satisfies $x_{t'_j} = x_{t_j}$.
end for
for $j = 1, \cdots, n$ **do**
    Set $a_{0,j} = 0$
    for $i = 1, \cdots, n$ **do**
        Compute $b_{ij} = \min(t'_{j-1}, t_i) - \max(t'_j, t_{i-1})$
        Compute $a_{ij} = a_{(i-1),j} + b_{ij}$
    end for
end for

With this property, we can compute b_{ij} easily. According to the definition of I_j, one endpoint of $z^{-1}(I_j)$ is $w \cdot x_{t_j}$, so we just need to find another endpoint as $t'_j = z^{-1}(w \cdot x_{t_j})$, which is equivalent to solving the equation $w \cdot x_{t'_j} = w \cdot x_{t_j}$.

Note x_t only has two dimensions, and the first dimension is a constant. Hence, the above equation does not depend on w, and it suffices to solve $x_{t'_j} = x_{t_j}$, where the left-hand side is a function of the unknown t'_j, and the right-hand side is a function of the observed data. It can be easily solved by root finding algorithms. We can then compute b_{ij} as:

$$b_{ij} = \min(t'_{j-1}, t_i) - \max(t'_j, t_{i-1})$$

The min and max operator implement the interval intersection. Since $z(t)$ is monotonic decreasing, we have $t'_{j-1} \geq t'_j$. Figure 2.1 illustrates the algorithm.

After we have computed b_{ij}, a_{ij} is readily available by $a_{ij} = \sum_{i' <= i} b_{i'j}$. The corresponding index sets $\mathcal{S}_i$ contain nonzero a_{ij}'s. The detailed procedures are presented in Algorithm 1.

2.2.4 Overall Algorithm

With Theorem 2, we can replace the integral of an unknown function by the weighted summation of its values defined at the intensity of each observed event. Hence we

can represent the $g \in \mathcal{F}$ non-parametrically, and reformulate the objective function as:

$$\min_{g \in \mathcal{F}, w} \frac{1}{n} \sum_{i=1}^{n} \left(N_i - \sum_{j \in \mathcal{S}_i} a_{ij} g(w \cdot x_j) \right)^2. \tag{2.5}$$

We optimize g and w alternatively until convergence. The update rules for w and g are presented as follows.

Update $\hat{w}$. Given $\hat{g}^k$, the update rule for $\hat{w}^{k+1}$ is:

$$\hat{w}^{k+1} = \hat{w}^k + \frac{1}{n} \sum_{i=1}^{n} \left(N_i - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{g}^k(\hat{w}^k \cdot x_j) \right) \sum_{j \in \mathcal{S}_i} a_{ij} x_j \tag{2.6}$$

Similar to the Isotron algorithm [50], this update rule is parameter free and Perceptron-like.

Update $\hat{g}$. Note that $\hat{g}$ is a non-parametric function which is represented by its values $\hat{y}_i^k$ at $\hat{w}^k \cdot x_i$. Therefore, we only need to determine its values on existing data points.

Given $\hat{w}^k$, we first sort $\left\{ \hat{w}^k \cdot x_i \right\}_{i=1}^n$ such that it is an increasing sequence. That is, we re-label the data points according to the sorted order. Then we solve the following Quadratic Programming problem for $\left\{ \hat{y}_i^{k+1} \right\}_{i=1}^n$:

$$\min \sum_{i=1}^{n} (N_i - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^{k+1})^2 \tag{2.7}$$

$$\text{s.t. } \hat{y}_i^{k+1} \leq \hat{y}_{i+1}^{k+1},\ 1 \leq i \leq n-1 \tag{2.8}$$

For simplicity we re-write the problem in matrix notations. Denote $\boldsymbol{N} = (N_1, \cdots, N_n)^\top$, $\hat{\boldsymbol{y}} = (\hat{y}_1, \cdots, \hat{y}_n)^\top$, $\boldsymbol{A}_{i,j} = a_{ij}$ if $j \in \mathcal{S}_i$ and $\boldsymbol{A}_{i,j} = 0$ otherwise. The monotonic constraint in (2.8) can be written as $\boldsymbol{B}\boldsymbol{y} \leq \boldsymbol{0}$ where $\boldsymbol{B}$ is the adjacent difference operator: $\boldsymbol{B}_{i,i} = 1$, $\boldsymbol{B}_{i,i+1} = -1$ and other entries are zero. Then we arrive at the following

Algorithm 2 LEARN-ISOTONIC-FUNC

Input: $\{N_i\}$, $\{a_{ij}\}$, η
Output: $\hat{\boldsymbol{y}}$
Initialize $\hat{\boldsymbol{y}}^0$ randomly
Construct matrices $\boldsymbol{N}$, $\boldsymbol{A}$ from input
$t = 0$
repeat
  $t = t + 1$
  $\hat{\boldsymbol{y}}^{t+1} = \Pi\left[\hat{\boldsymbol{y}}^t + \eta \boldsymbol{A}^\top \left(\boldsymbol{N} - \boldsymbol{A}\hat{\boldsymbol{y}}^t\right)\right]$
until convergence

formulation:

$$\min_{\hat{\boldsymbol{y}}} \|\boldsymbol{N} - \boldsymbol{A}\hat{\boldsymbol{y}}\|^2, \ s.t. \ \boldsymbol{B}\hat{\boldsymbol{y}} \leq 0$$

This is a convex problem and can be computed efficiently using projected gradient descent:

$$\hat{\boldsymbol{y}}^{t+1} := \Pi\left[\hat{\boldsymbol{y}}^t + \eta \boldsymbol{A}^\top \left(\boldsymbol{N} - \boldsymbol{A}\hat{\boldsymbol{y}}^t\right)\right]$$

where $\Pi[\boldsymbol{u}]$ is an operator that projects $\boldsymbol{u}$ into the feasible set:

$$\Pi[\boldsymbol{u}] = \operatorname*{argmin}_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{u}\|^2, \ s.t. \ \boldsymbol{B}\boldsymbol{x} \leq 0$$

The projection is exactly the Isotonic regression problem and can be solved by PAV [45] in $O(n \log n)$. In addition, the computation of the gradient is also efficient since $\boldsymbol{A}$ is a sparse matrix and it takes time $O(n + \text{nnz}(\boldsymbol{A}))$, where $\text{nnz}(\boldsymbol{A})$ is the number of nonzero elements. The number of iterations required to reach ϵ accuracy is $O(1/\epsilon)$, hence the overall complexity is $O((n \log n + \text{nnz}(\boldsymbol{A}))/\epsilon)$. This can also be accelerated to $O((n \log n + \text{nnz}(\boldsymbol{A}))/\sqrt{\epsilon})$ using Nesterov's acceleration scheme [55]. The algorithm is illustrated in Algorithm 2 and the whole alternating minimization procedure is summarized in Algorithm 8. Such procedure will efficiently find the near-optimal $\hat{g}$ and $\hat{w}$.

Algorithm 3 ISOTONIC-HAWKES

Input: Sequences of events $\{t_i\}_{i=1}^n$
Output: $\hat{g}, \hat{w}$
Compute $x_i = (1, \sum_{t_j \in \mathcal{H}_{t_i}} \kappa(t_i - t_j)^\top$ for $i \in [n]$
$\{a_{ij}\}$ = COMPUTE-COEFFICIENT($\{t_i\}$)
Compute $N_i = i$ for $i \in [n]$
Initialize w^0, g^0 randomly
$k = 0$
repeat
  $k = k + 1$
  Sort the data according to $\hat{w}^k \cdot x_i$
  Update $\hat{g}^k$ = LEARN-ISOTONIC-FUNC($\{N_i\}$, $\{a_{ij}\}$)
  Update $\hat{w}^{k+1}$ using (2.6)
until $\text{loss}(\hat{g}, \hat{w}) \leqslant \epsilon$

2.3 Theoretical Guarantees

We now provide the theoretical analysis of convergence property. First we define the error as:

$$\varepsilon(\hat{g}^k, \hat{w}^k) = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{g}^k(\hat{w}^k x_i) - g^*(w^* \cdot x_i) \right)^2 \tag{2.9}$$

where $g^*(\cdot)$ and w^* are the unknown true link function and model parameters, respectively. The goal is to analyze how quickly $\varepsilon(\hat{g}^k, \hat{w}^k)$ decreases with k.

Notations. Set $y_i^* = g^*(w^* \cdot x_i)$ to be the expected value of each y_i. Let $\bar{N}_i$ be the expected value of N_i. Then we have $\bar{N}_i = \sum_{j \in \mathcal{S}_i} a_{ij} y_j^*$. Clearly we do not have access to $\bar{N}_i$. However, consider a hypothetical call to the algorithm with input $\{(x_i, \bar{N}_i)\}_{i=1}^n$ and suppose it returns $\bar{g}^k$. In this case, we define $\bar{y}_i^k = \bar{g}^k(\bar{w}^k \cdot x_i)$.

We first bound the error using the squared distance $\|\hat{w}^k - w^*\|^2$ between two consecutive iterations:

Lemma 3. *Suppose that* $\|\hat{w}^k - w^*\| \leq W$, $\|x_i\| \leq 1$, $\sqrt{c} \leq \sum_{j \in \mathcal{S}_i} a_{ij} \leq \sqrt{C}$, $y_j \leq M$, *and*

$$\frac{1}{n} \sum_{i=1}^{n} \left| (N_i - \bar{N}_i) \right| \leq \eta_1, \quad \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \mathcal{S}_i} a_{ij} |\hat{y}_j^k - \bar{y}_j^k| \leq \eta_2$$

then we have:

$$\|\hat{w}^k - w^*\|^2 - \|\hat{w}^{k+1} - w^*\|^2 \geq C_2\varepsilon(\hat{g}^k, \hat{w}^k) - C_1(\eta_1 + \eta_2),$$

where $C_1 = \max\{5CW, 4M\sqrt{c} + 2CW\}$, $C_2 = 2c - C$.

The squared distance decreases at a rate depending on $\varepsilon(\hat{g}^k, \hat{w}^k)$ and the upper bounds η_1 and η_2. The following two lemmas provide the concrete upper bounds.

Lemma 4 (Martingale Concentration Inequality)**.** *Suppose* $dM(t) \leq K$, $V(t) \leq k$ *for all* $t > 0$ *and some* $K, k \geq 0$. *With probability at least* $1 - \delta$, *it holds that*

$$\frac{1}{n}\sum_{i=1}^{n} |N_i - \bar{N}_i| \leq O\left((K + \sqrt{4K^2 + 8k^2})\big(\log(1/\delta)\big)^{1/2}\right).$$

Note $N_i - \bar{N}_i = M_i$, which is the martingale at time t_i. A continuous martingale is a stochastic process such that $\mathbb{E}[M_t|\{M_\tau, \tau \leq s\}] = M_s$. It means the conditional expectation of an observation at time t is equal to the observation at time s, given all the observations up to time $s \leq t$. $V(t)$ is the variation process. The martingale serves as the noise term in point processes (similar to Gaussian noise in regression) and can be bounded using the Bernstein-type concentration inequality.

Lemma 5. *With probability at least* $1 - \delta$, *it holds for any* k *that*

$$\frac{1}{n}\sum_{j=1}^{n} |\hat{y}_j^k - \bar{y}_j^k| \leq O\left(\left(\frac{W^2 \log(Wn/\delta)}{n}\right)^{1/3}\right).$$

Lemma 5 relates $\hat{y}_j^k$ (the value we can actually compute) to $\bar{y}_j^k$ (the value we could compute if we had the conditional means of N_j). The proof of this lemma uses the covering number technique in [51].

We now state the main theorem:

Theorem 6. *Suppose* $\mathbb{E}[N_i|\mathcal{H}_{t_i}] = \int_0^{t_i} g^*(w^* \cdot x_t)\mathrm{d}t$, *where* g^* *is monotonic increasing,*

1-Lipschitz and $\|w^*\| \leq W$. *Then with probability at least* $1-\delta$, *there exist some iteration* $k < O\left(\left(\frac{Wn}{\log(Wn/\delta)}\right)^{1/3}\right)$ *such that*

$$\varepsilon(\hat{g}^k, \hat{w}^k) \leq O\left(\left(\frac{W^2 \log(Wn/\delta)}{n}\right)^{1/3}\right).$$

Theorem 6 implies that some iteration has $\varepsilon(\hat{g}^k, \hat{w}^k) = O(1/\sqrt[3]{n})$. It is plausible the rate is sharp based on the information-theoretic lower bounds in [56].

Proof sketch. We conduct a telescoping sum of Lemma 3 and show that there are at most $O\big(W/(\eta_1+\eta_2)\big)$ iterations before the error $\varepsilon(\hat{g}^k, \hat{w}^k)$ is less than $O(\eta_1+\eta_2)$. Set η_1, η_2 to be the right-hand sides in Lemma 4 and 5. Since η_2 is the dominant term compared with η_1, we replace η_1 by η_2 in the final results. This completes the proof.

2.4 Extensions

We provide several extensions of the Isotonic-Hawkes processes to more general cases.

General point processes. The idea and algorithm of Isotonic-Hawkes can be easily extended to other point processes. The time-dependent feature x_t in Isotonic-Hawkes is designed to capture the influence of history. However, one can also incorporate and extend other features in prior work [57, 58] or design it from users' experiences and the application domain.

Learning monotonic decreasing functions. Our model can be easily extended to learn a monotonically decreasing function. We just need to change the sign of the inequality in (2.8). Note that Theorem 6 still holds in this case.

Low-rank Isotonic-Hawkes processes. We can also use our model to learn low rank parameters. For example, in the time-sensitive recommendations for online services [5], we model user u's past consumption events on item i as an Isotonic-Hawkes process (Figure 2.2) and need to learn the parameters $\{\lambda_0^{ui}, \alpha^{ui}, g^{ui}\}$ for each

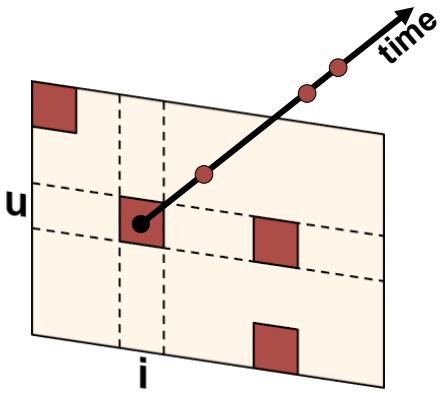


Figure 2.2: The sequence of events for each pair is modeled as an Isotonic-Hawkes process.

user-item pair (u, i). That is, we Since group structure often exists within users' preferences and items' attributes, we assume that both matrices $\boldsymbol{\lambda}_0 = (\lambda_0^{ui})$ and $\boldsymbol{\alpha} = (\alpha^{ui})$ have low-rank structures. We can then factorize them as product of two rank r matrices: $\boldsymbol{\lambda}_0 = \boldsymbol{X}_0\boldsymbol{Y}_0$ and $\boldsymbol{\alpha}_0 = \boldsymbol{X}\boldsymbol{Y}$. Then we formulate the learning problem by applying our objective function in (2.4) for each observed pair (u, i):

$$\min_{\boldsymbol{X}_0,\boldsymbol{Y}_0,\boldsymbol{X},\boldsymbol{Y},\boldsymbol{g}} \sum\nolimits_{\mathcal{H}^{ui}\in\mathcal{O}} \ell\left(\mathcal{H}^{ui}\right) \tag{2.10}$$

$$\ell\left(\mathcal{H}^{ui}\right) = \frac{1}{n^{ui}} \sum_{j=1}^{n^{ui}} \left(N_j^{ui} - \int_0^{t_i} g^{ui}(\boldsymbol{w}^{ui} \cdot x_t^{ui}) \mathrm{d}t \right)^2$$

where $w^{ui} = (\lambda^{ui}, \alpha^{ui})$. n^{ui} is total number of events and $\mathcal{H}^{ui}$ is the set of history events for user-item pair (u, i). $\mathcal{O} = \{\mathcal{H}^{ui}\}$ is the collection of all observed sequences.

We use the alternating minimization technique to update $\boldsymbol{X}_0, \boldsymbol{Y}_0, \boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{g}$. First keep $\boldsymbol{g}^k$ fixed and update the parameters to $\boldsymbol{X}_0^{k+1}, \boldsymbol{Y}_0^{k+1}, \boldsymbol{X}^{k+1}, \boldsymbol{Y}^{k+1}$, then keep them fixed and update g^{k+1}. For the unobserved user-item pairs, after the algorithm stops, we obtain g^{ui} by taking average of the user's link functions learned from data.

Multi-dimensional Isotonic-Hawkes processes. We extend the Isotonic-Hawkes process to multi-dimension, which is particular useful to model information

diffusion in social networks. It is defined by a U-dimensional point process $\boldsymbol{N}(t)$, with intensity for the u-th dimension as:

$$\lambda^u(t) = g^u\left(\lambda_0^u + \alpha^{uu_i} \sum_{i:t_i \in \mathcal{H}_t} \kappa(t - t_i)\right) = g^u(w^u \cdot x_t^u)$$

where $\alpha^{uu'}$ captures the degree of influence in the u'-th dimension to the u-th dimension. As for learning, the input data is a sequence of events observed in the form of $\{(t_i, u_i)\}$ and each pair represents an event occurring at the u_i-th dimension at time t_i. Hence, for each dimension u, set $N_i^u = N^u(t_i)$, and we solve the problem:

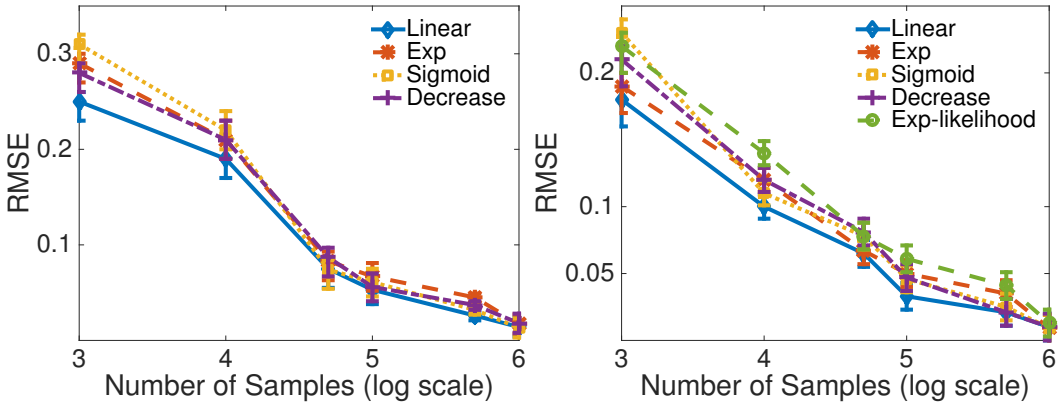
$$\min_{g^u, w^u} \frac{1}{n^u} \sum_{i=1}^{n^u} \left(N_i^u - \int_0^{t_i} g^u(w^u \cdot x_t^u) \mathrm{d}t\right)^2 \tag{2.11}$$

where the i-th entry of w^u and x_t^u is $w^u(i) = (\lambda_0^u, \alpha^{uu_i})$ and $x_t^u(i) = (1, \sum_{t_i \in \mathcal{H}_t} \kappa(t, t_i))$ respectively. Our goal is to learn $\boldsymbol{w} = (w^u)$ and $\boldsymbol{g} = (g^u)$. From (2.11) we can see that learning in each dimension u is independent of others. Hence under this framework, w^u and g^u can be learned using Algorithm 8 in parallel efficiently.

2.5 Experiments

We evaluate the performance of Isotonic-Hawkes on both synthetic and real-world datasets with respect to the following tasks :

- **Convergence**: investigate how well Isotonic-Hawkes can learn the true parameters as the number of training samples increases.
- **Fitting capability**: study how well Isotonic-Hawkes can explain real-world data by comparing it with the classic Hawkes process.
- **Time-sensitive recommendation**: demonstrate that Isotonic Hawkes can improve the predictive performance in item recommendation and time prediction.
- **Diffusion network modeling**: evaluate how well Isotonic-Hawkes can model the information diffusion from cascades of temporal events.



(a) RMSE for function g (b) RMSE for parameter w

Figure 2.3: Convergence by number of samples.

2.5.1 Experiments on Synthetic Data

Experimental setup. Table 2.1 lists the ground-truth setting with four typical link functions $g(\cdot)$ and the respective model parameters w. The first three link functions (Linear, Exp, Sigmoid) are monotonically increasing, while the last one is strictly decreasing. For the *Exp* link function, we explore the performance of learning self-inhibition by setting α to be negative. Without loss of generality, we use the unit-rate exponential decaying function as the triggering kernel. Then, based on the configuration of each row in Table 2.1, we simulate one million events using Ogata's Thinning algorithm [59].

Table 2.1: Model configurations.

Name	**link function** g	**Weights** w
Linear	$g(x) = x$	$w = (1.2, 0.6)$
Exp	$g(x) = e^x$	$w = (0.5, -0.1)$
Sigmoid	$g(x) = 1/(1 + e^{-4(x-2)})$	$w = (0.5, 1.2)$
Decrease	$g(x) = 1 - 1/(1 + e^{-4(x-3)})$	$w = (0.5, 1.2)$

Convergence analysis. We first evaluate the convergence property of our learning algorithm by increasing the number of samples from 1,000 to 1,000,000. For each dataset, we repeat the simulations ten times and report the averaged results. Figure 2.3 (a) shows the Root Mean Squared Error (RMSE) between the values of the

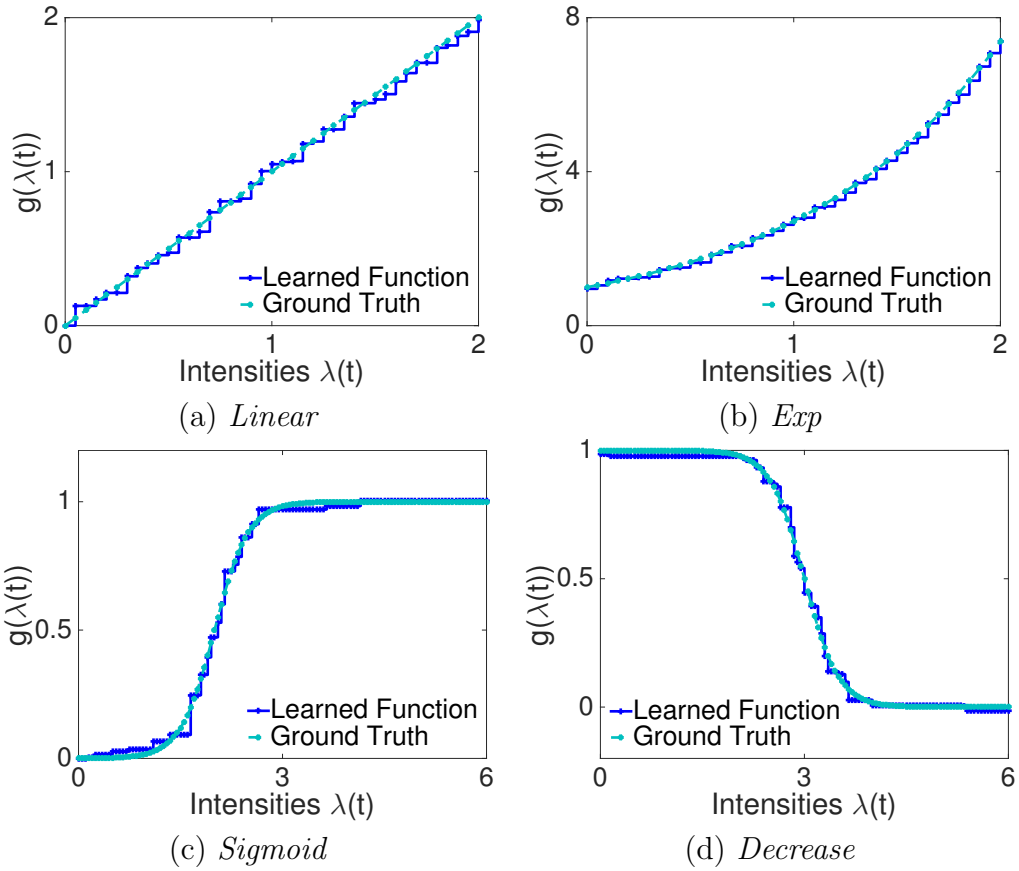


Figure 2.4: Comparison between learned link function and the ground truth on four synthetic datasets.

learned function and those given by the ground-truth link function as a function of training data size. Figure 2.3(b) shows the RMSE of learning the model parameters. The x-axis is in log scale. Since in all cases, the RMSE decreases in a consistent way, it demonstrates that Isotonic-Hawkes is very robust regardless of the latent ground-truth link functions. Furthermore, for the *Exp* link function, we compare the RMSE between our method and the likelihood based approach, Exp-likelihood [42], which has access to the link function and discretizes the time interval to compute the integral in the likelihood. Our method works better at estimating w. Finally, the ability to recover the linear link function verifies that Isotonic-Hawkes naturally includes the classic Hawkes process as a special case and is much more expressive to explain the data.

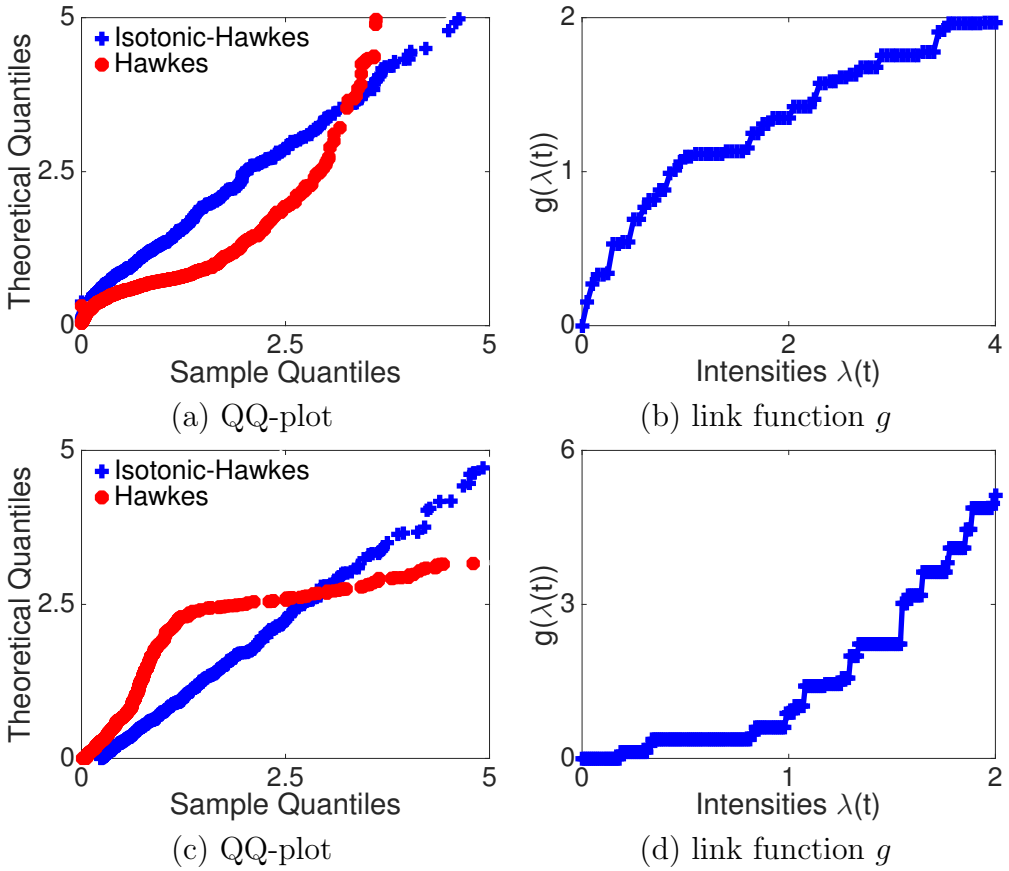
Visualization of recovered link functions. We also plot each learned link function against the respective ground-truth in Figure 2.4 trained with 1,000,000 events. In all the cases, the algorithm can achieve the global optimal to precisely recover the true functions.

2.5.2 Experiments on Time-sensitive Recommendation

Experimental setup. For the task of time-sensitive recommendation, we fit a low-rank Isotonic-Hawkes process with the alternating minimization technique from (2.10) to solve the following two related problems proposed from [5] : (1) how to recommend the most relevant item at the right moment; (2) how to accurately predict the next returning-time of users to existing services. We evaluate the predictive performance of our model on two real datasets. *last.fm*[1] consists of the music listening histories of around 1,000 users over 3,000 different albums. We use the events of the first three months for training and those of the next month for testing. There are around 20,000 user-album pairs with more than one million events. *tmall.com*[2] contains online shopping records. There are around 100K events between 26, 376 users and 2,563 stores. We use the events of the first four months for training and those of the last month for testing. The unit time is an hour.

Better data fitting capability. Since the true temporal dynamics governing the temporal point patterns are unknown, we first investigate whether our new model can better explain the data compared with the classic Hawkes process. According to the Time Changing Theorem [60], given a sequence $\mathcal{T} = \{t_i\}_{i=1}^n$ and a point process with intensity $\lambda(t)$, the set of samples $\{\int_{t_{i-1}}^{t_i} \lambda(t)\mathrm{d}t\}_{i=1}^n$ should conform to a unit-rate exponential distribution if $\mathcal{T}$ is truly sampled from the process. As a consequence, we compare the theoretical quantiles from the unit-rate exponential distribution with the empirical quantiles of different models. The closer the slope of QQ-plot goes to one,

[1] http://www.dtic.upf.edu/õcelma/MusicRecommendationDataset/lastfm-1K.html
[2] http://ijcai-15.org/index.php/repeat-buyers-prediction-competition



(a) QQ-plot (b) link function g

(c) QQ-plot (d) link function g

Figure 2.5: Experiment results on two randomly picked sequences from *last.fm* data. (a-b) and (c-d) correspond to two sequences.

the better a model matches the point patterns. [5] has shown that Hawkes process fits the data better compared to other simple processes.

In Figure 2.5 (a) and (c), we show that Isotonic-Hawkes achieves much better fitting capability. Furthermore, (b) and (d) visualize the learned link functions. In Figure 2.5(b), the function captures the phenomenon that the user's interests tend to saturate in the long-run despite that he may be excited about the item initially. Intuitively, we can also see this from (a), where Hawkes process has larger sample quantiles than the theoretical one, which means $\int_{t_{i-1}}^{t_i} \lambda(t)\mathrm{d}t$ is larger than the value it should be. Hence using a saturating function in (b) helps adjusting the Hawkes intensity $\lambda(t)$ and make it smaller. In contrast, (d) presents the opposite trend where the user was not quite interested in the given item initially, but later became addicted

to it. Since the Hawkes sample quantile is smaller than the theoretical one in (c), link function helps changing $\lambda(t)$ to be larger. Hence learning the link function is important.

Recommendation improvements. We evaluate the predictive performance on the two tasks following [5] : (1) Rank prediction. At each testing moment, we record the predicted rank of the target item based on the respective intensity function. We report the average rank over all test events. Smaller value indicates better performance. (2) Arrival-time prediction. We predict the arrival time of the next testing event and report the mean absolute error (MAE) between the predicted time and the true value. In addition, besides Hawkes process, we also compare with the commonly used Poisson process, which is a relaxation of the Hawkes model by assuming that each user-item pair has constant base intensity independent of the history, as well as the state-of-the-art Tensor factorization method [61] which applies Poisson factorization to fit the number of events in each discretized time slot and has better performance than methods based squared loss [62]. We use the parameters averaged over all time intervals to make predictions. The latent rank of the low-rank Isotonic-Hawkes process and the tensor method are tuned to give the best performance.

We summarize the results in Figure 2.6. First, Hawkes outperforms the Poisson process, which means that considering the effects of history is helpful. Second, Isotonic-Hawkes outperforms Hawkes process for a significant margin thanks to the better data fitting capability shown in Figure 2.5. For time prediction, since the MAE's unit is hour, we can see that the error difference between Isotonic-Hawkes and Hawkes is about three days. The online shopping services can benefit a lot from this improvement and make better demand predictions.

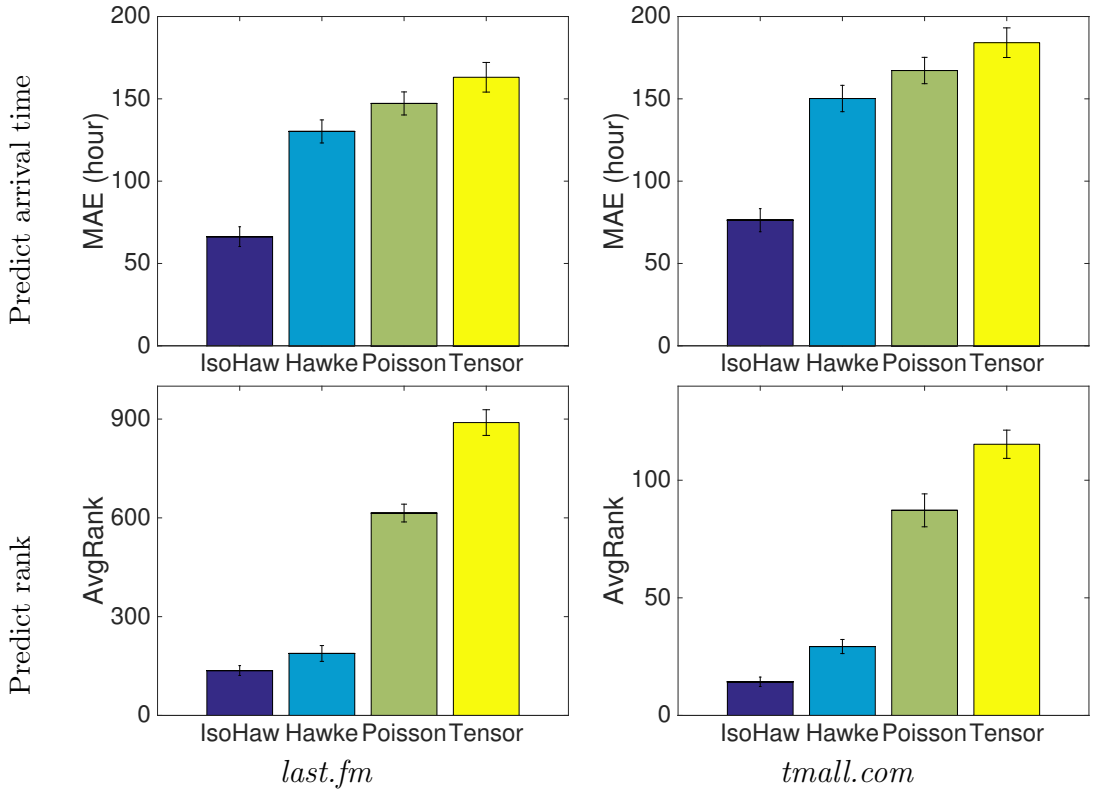
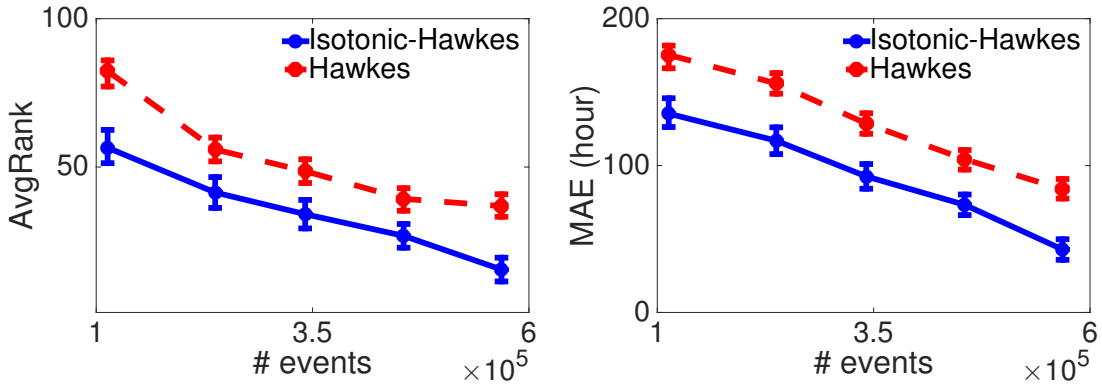


Figure 2.6: Time-sensitive recommendation results.

2.5.3 Experiments on Modeling Diffusion Networks

Finally, we apply the multi-dimension Isotonic-Hawkes process with the model estimation procedure in (2.11) to recover the latent information diffusion network reflected by the nonzero patterns of the mutual excitation matrix over the real *Network* dataset from [35]. This dataset comprises of all tweets posted by 2,241 users in eight month. The network has 4,901 edges. We split data into a training set (covering 85% of the total events) and a test set (covering the remaining 15%) according to time. Being similar to the time-sensitive recommendation task, we report the average rank of all testing events and MAE for the arrival time prediction with an increasing proportion of training events. Figure 2.7 verifies that Isotonic-Hawkes outperforms Hawkes process consistently.



(a) AvgRank (b) Arrival-time

Figure 2.7: Prediction results on the *Network* dataset.

2.6 Summary

In this chapter, we have proposed a novel nonlinear Hawkes process, the Isotonic-Hawkes process, with a flexible nonlinear link function. Along with the model, we have developed a computationally and statistically efficient algorithm to learn the link function and model parameters jointly, and rigorously show that under mild assumptions of the monotonicity, our algorithm is guaranteed to converge to the global optimal solution. Furthermore, our model is very general and can be extended to many different forms, including monotonically decreasing link functions, low-rank Isotonic-Hawkes processes model and multi-dimensional Isotonic-Hawkes processes. Experiments on both synthetic and real world datasets empirically verify the theoretical guarantees and demonstrate the superior predictive performance compared to other baselines.

Besides designing more expressive point process models, sometimes we also want to incorporate the rich contextual information of users, items, and interactions when designing models. In online service platforms, users interact with different items at different times. Their evolving interest in items is captured implicitly by the time of the interaction. By understand users' dynamic interests and serving them with potentially interesting items, these platforms can improve the satisfaction of users,

and boost the activities or revenue of the sites. In the next Chapter, we present an efficient framework that is able to model the dynamics of users' and items' embeddings and use these embeddings to design point process models.

CHAPTER 3
COEVOLUTIONARY FEATURE EMBEDDING FOR CONTINUOUS-TIME USER-ITEM INTERACTIONS

3.1 Introduction

Online social platforms and service websites, such as Reddit, Netflix and Amazon, are attracting thousands of users every minute. Effectively recommending the appropriate service items is a fundamentally important task for these online services. By understanding the needs of users and serving them with potentially interesting items, these online platforms can improve the satisfaction of users, and boost the activities or revenue of the sites due to increased user postings, product purchases, virtual transactions, and/or advertisement clicks [63, 5].

As the famous saying goes "You are what you eat and you think what you read", both users' interests and items' semantic features are dynamic and can *evolve* over time [10, 11]. The interactions between users and service items play a critical role in driving the evolution of user interests and item features. For example, for movie streaming services, a long-time fan of comedy watches an interesting science fiction movie one day, and starts to watch more science fiction movies in place of comedies. Likewise, a single movie may also serve different segment of audiences at different times. For example, a movie initially targeted for an older generation may become popular among the younger generation, and the features of this movie need to be redefined.

Another important aspect is that users' interests and items' features can *co-evolve* over time, that is, their evolutions are intertwined and can influence each other. For instance, in online discussion forums, such as Reddit, although a group (item) is

initially created for political topics, users with very different interest profiles can join this group (**user** $\rightarrow$ **item**). Therefore, the participants can shape the actual direction (or features) of the group through their postings and responses. It is not unlikely that this group can eventually become one about education simply because most users here concern about education (**item** $\rightarrow$ **user**). As the group is evolving towards topics on education, some users may become more attracted to education topics, and to the extent that they even participate in other dedicated groups on education. On the opposite side, some users may gradually gain interests in sports groups, lose interests in political topics and become inactive in this group. Such coevolutionary nature of user-item interactions raises very interesting questions on how to model them elegantly and how to learn them from observed interaction data.

Nowadays, user-item interaction data are archived in increasing temporal resolution and becoming increasingly available. Each individual user-item iteration is typically logged in the database with the precise time-stamp of the interaction, together with additional context of that interaction, such as tag, text, image, audio and video. Furthermore, the user-item interaction data are generated in an *asynchronous* fashion in a sense that any user can interact with any item at any time and there may not be any coordination or synchronization between two interaction events. These types of event data call for new representations, models, learning and inference algorithms.

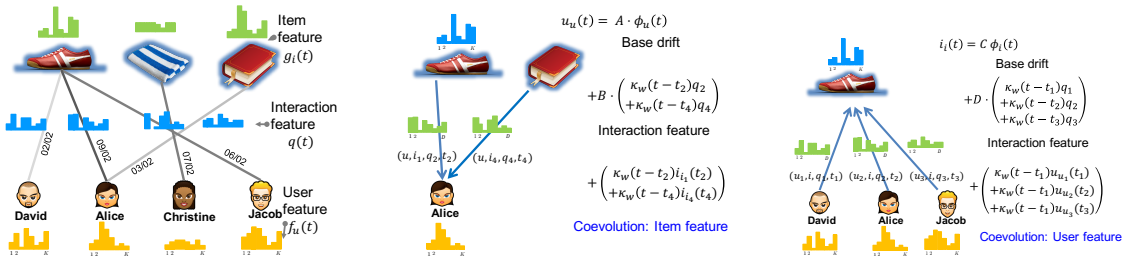
Despite the temporal and asynchronous nature of such event data, for a long-time, the data has been treated predominantly as a static graph, and fixed latent features have been assigned to each user and item [64, 65, 66, 67, 68, 63, 69]. In more sophisticated methods, the time is divided into epochs, and static latent feature models are applied to each epoch to capture some temporal aspects of the data [10, 61, 62, 70, 11, 71, 72, 73, 74, 75, 76]. For such epoch-based methods, it is not clear how to choose the epoch length parameter due to the asynchronous nature of the

user-item interactions. First, different users may have very different time-scale when they interact with those service items, making it very difficult to choose a unified epoch length. Second, it is not easy for the learned model to answer fine-grained time-sensitive queries such as when a user will come back for a particular service item. It can only make such predictions down to the resolution of the chosen epoch length. Most recently, [5] proposed an efficient low-rank point process model for time-sensitive recommendations from recurrent user activities. However, it still fails to capture the heterogeneous coevolutionary properties of user-item interactions with much more limited model flexibility. Furthermore, it is difficult for this approach to incorporate observed context features.

In this chapter, we propose a coevolutionary latent feature process for continuous-time user-item interactions, which is designed specifically to take into account the asynchronous nature of event data, and the co-evolution nature of users' and items' latent features. Our model assigns an evolving latent feature process for each user and item, and the co-evolution of these latent feature processes is considered using two parallel components:

- (**Item** $\rightarrow$ **User**) A user's latent feature is determined by the latent features of the items he interacted with. Furthermore, the contributions of these items' features are temporally discounted by an exponential decaying kernel function, which we call the Hawkes [36] feature averaging process.
- (**User** $\rightarrow$ **Item**) Conversely, an item's latent features are determined by the latent features of the users who interact with the item. Similarly, the contribution of these users' features is also modeled as a Hawkes feature averaging process.

Besides the two sets of intertwined latent feature processes, our model can also take into account the presence of potentially high dimensional observed context features and links the latent features to the observed context features using a low dimensional projection. Despite the sophistication of our model, we show that the model



(a) Data as a bipartite graph (b) User latent feature process (c) Item latent feature process

Figure 3.1: Model illustration. (a) User-item interaction events data. Each edge contains user, item, time, and interaction feature. (b) Alice's latent feature consists of three components: the drift of baseline feature, the time-weighted average of interaction feature, and the weighted average of item feature. (c) The symmetric item latent feature process. $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ are embedding matrices from high dimension feature space to latent space. $\kappa_\omega(t) = \exp(-\omega t)$ is an exponential decaying kernel.

parameter estimation, a seemingly non-convex problem, can be transformed into a convex optimization problem, which can be efficiently solved by the latest conditional gradient-like algorithm. Finally, the coevolutionary latent feature processes can be used for down-streaming inference tasks such as the next-item and the return-time prediction. We evaluate our method over a variety of datasets, verifying that our method can lead to significant improvements in user behavior prediction compared to the state-of-the-arts.

3.2 Coevolutionary Latent Feature Processes

In this section, we present the framework to model the temporal dynamics of user-item interactions. We first explicitly capture the co-evolving nature of users' and items' latent features. Then, based on the compatibility between a user' and item's latent feature, we model the user-item interaction by a temporal point process and parametrize the intensity function by the feature compatibility.

3.2.1 Event Representation

Given m users and n items, the input consists of all users' history events: $\mathcal{T} = \{e_k\}$, where $e_k = (u_k, i_k, t_k, \boldsymbol{q}_k)$ means that user u_k interacts with item i_k at time t_k and generates an interaction feature vector $\boldsymbol{q}_k \in \mathbb{R}^D$. For instance, the interaction feature can be a textual message delivered from the user to the chatting-group in Reddit or a review of the business in Yelp. It can also be unobservable if the data only contains the temporal information.

3.2.2 Latent Feature Processes

We associate a latent feature vector $\boldsymbol{u}_u(t) \in \mathbb{R}^K$ with a user u and $\boldsymbol{i}_i(t) \in \mathbb{R}^K$ with an item i. These features represent the subtle properties which cannot be directly observed, such as the interests of a user and the semantic topics of an item. Specifically, we model $\boldsymbol{u}_u(t)$ and $\boldsymbol{i}_i(t)$ as follows:

User latent feature process. For each user u, we formulate $\boldsymbol{u}_u(t)$ as:

$$\boldsymbol{u}_u(t) = \boldsymbol{A} \underbrace{\boldsymbol{\phi}_u(t)}_{\text{base drift}} + \boldsymbol{B} \underbrace{\sum_{\{e_k | u_k = u, t_k < t\}} \kappa_\omega(t - t_k) \boldsymbol{q}_k}_{\text{Hawkes interaction feature averaging}} + \underbrace{\sum_{\{e_k | u_k = u, t_k < t\}} \kappa_\omega(t - t_k) \boldsymbol{i}_{i_k}(t_k)}_{\text{co-evolution: Hawkes item feature averaging}}, \tag{3.1}$$

Item latent feature process. For each item i, we specify $\boldsymbol{i}_i(t)$ as:

$$\boldsymbol{i}_i(t) = \boldsymbol{C} \underbrace{\boldsymbol{\phi}_i(t)}_{\text{base drift}} + \boldsymbol{D} \underbrace{\sum_{\{e_k | i_k = i, t_k < t\}} \kappa_\omega(t - t_k) \boldsymbol{q}_k}_{\text{Hawkes interaction feature averaging}} + \underbrace{\sum_{\{e_k | i_k = i, t_k < t\}} \kappa_\omega(t - t_k) \boldsymbol{u}_{u_k}(t_k)}_{\text{co-evolution: Hawkes user feature averaging}}, \tag{3.2}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D} \in \mathbb{R}^{K \times D}$ are the embedding matrices mapping from the explicit high-dimensional feature space into the low-rank latent feature space. Figure 3.1 highlights the basic setting of our model. Next we discuss the rationale of each term in detail.

Drift of base features. $\boldsymbol{\phi}_u(t) \in \mathbb{R}^D$ and $\boldsymbol{\phi}_i(t) \in \mathbb{R}^D$ are the explicitly observed properties of user u and item i, which allows the basic features of users (*e.g.*, a user's self-crafted interests) and items (*e.g.*, textual categories and descriptions) to smoothly drift through time. Such changes of basic features normally are caused by external influences. One can parametrize $\boldsymbol{\phi}_u(t)$ and $\boldsymbol{\phi}_i(t)$ in many different ways, *e.g.*, the exponential decaying basis to interpolate these features observed at different times.

Evolution with interaction feature. Users' and items' features can evolve and be influenced by the characteristics of their interactions. For instance, the genre changes of movies indicate the changing tastes of users. The theme of a chatting-group can be easily shifted to certain topics of the involved discussions. In consequence, this term captures the cumulative influence of the past interaction features to the changes of the latent user (item) features. The triggering kernel $\kappa_\omega(t - t_k)$ associated with each past interaction at t_k quantifies how such influence can change through time. Its parametrization depends on the phenomena of interest. Without loss of generality, we choose the exponential kernel $\kappa_\omega(t) = \exp(-\omega t)$ to reduce the influence of each past event. In other words, only the most recent interaction events will have bigger influences. Finally, the embedding matrices $\boldsymbol{B}, \boldsymbol{D}$ map the observable high dimension interaction feature to the latent space.

Coevolution with Hawkes feature averaging processes. Users' and items' latent features can mutually influence each other. This term captures the two parallel processes:

- *Item* $\rightarrow$ *User*. A user's latent feature is determined by the latent features of the items he interacted with. At each time t_k, the latent item feature is $\boldsymbol{i}_{i_k}(t_k)$. Furthermore, the contributions of these items' features are temporally discounted by a kernel function $\kappa_\omega(t)$, which we call the Hawkes feature averaging process. The name comes from the fact that Hawkes process captures the temporal influence of history events in its intensity function. In our model, we capture both the temporal

influence and feature of each history item as a latent process.

- *User* $\rightarrow$ *Item.* Conversely, an item's latent features are determined by the latent features of all the users who interact with the item. At each time t_k, the latent feature is $\boldsymbol{u}_{u_k}(t_k)$. Similarly, the contribution of these users' features is also modeled as a Hawkes feature averaging process.

Note that to compute the third co-evolution term, we need to keep track of the user's and item's latent features after each interaction event, *i.e.*, at t_k, we need to compute $\boldsymbol{u}_{u_k}(t_k)$ and $\boldsymbol{i}_{i_k}(t_k)$ in (3.1) and (3.2), respectively. Set $\mathbb{I}(\cdot)$ to be the indicator function, we can show by induction that

$$\begin{aligned}
\boldsymbol{u}_{u_k}(t_k) = \boldsymbol{A}\Big[\sum\nolimits_{j=1}^{k} \mathbb{I}[u_j = u_k]\kappa_\omega(t_k - t_j)\boldsymbol{\phi}_{u_j}(t_j)\Big] + \boldsymbol{B}\Big[\sum\nolimits_{j=1}^{k} \mathbb{I}[u_j = u_k]\kappa_\omega(t_k - t_j)\boldsymbol{q}_j\Big] \\
+ \boldsymbol{C}\Big[\sum\nolimits_{j=1}^{k-1} \mathbb{I}[u_j = u_k]\kappa_\omega(t_k - t_j)\boldsymbol{\phi}_{i_j}(t_j)\Big] + \boldsymbol{D}\Big[\sum\nolimits_{j=1}^{k-1} \mathbb{I}[u_j = u_k]\kappa_\omega(t_k - t_j)\boldsymbol{q}_j\Big] \\
\boldsymbol{i}_{i_k}(t_k) = \boldsymbol{C}\Big[\sum\nolimits_{j=1}^{k} \mathbb{I}[i_j = i_k]\kappa_\omega(t_k - t_j)\boldsymbol{\phi}_{i_j}(t_j)\Big] + \boldsymbol{D}\Big[\sum\nolimits_{j=1}^{k} \mathbb{I}[i_j = i_k]\kappa_\omega(t_k - t_j)\boldsymbol{q}_j\Big] \\
+ \boldsymbol{A}\Big[\sum\nolimits_{j=1}^{k-1} \mathbb{I}[i_j = i_k]\kappa_\omega(t_k - t_j)\boldsymbol{\phi}_{u_j}(t_j)\Big] + \boldsymbol{B}\Big[\sum\nolimits_{j=1}^{k-1} \mathbb{I}[i_j = i_k]\kappa_\omega(t_k - t_j)\boldsymbol{q}_j\Big]
\end{aligned}$$

In summary, we have incorporated both of the exogenous and endogenous influences into a single model. First, each process evolves according to the respective exogenous base temporal user (item) features $\phi_u(t)$ ($\phi_i(t)$). Second, the two processes also inter-depend on each other due to the endogenous influences from the interaction features and the entangled latent features. We present our model in the most general form and the specific choices of $\boldsymbol{u}_u(t), \boldsymbol{i}_i(t)$ are dependent on applications. For example, if no interaction feature is observed, we drop the second term in (3.1) and (3.2).

3.2.3 User-item Interactions as Temporal Point Processes

For each user, we model the recurrent occurrences of user u's interaction with all items as a multi-dimensional temporal point process. In particular, the intensity in

the i-th dimension (item i) is:

$$\lambda^{u,i}(t) = \underbrace{\eta^{u,i}}_{\text{long-term preference}} + \underbrace{\boldsymbol{u}_u(t)^\top \boldsymbol{i}_i(t)}_{\text{short-term preference}} , \tag{3.3}$$

where $\boldsymbol{\eta} = (\eta^{u,i})$ is a baseline preference matrix. The rationale of this formulation is threefold. First, instead of discretizing the time, we explicitly model the timing of each event occurrence as a continuous random variable, which naturally captures the heterogeneity of the temporal interactions between users and items. Second, the base intensity $\eta^{u,i}$ represents the long-term preference of user u to item i, independent of the history. Third, the tendency for user u to interact with item i at time t depends on the compatibility of their instantaneous latent features. Such compatibility is evaluated through the inner product of their time-varying latent features.

Our model inherits the merits from classic content filtering, collaborative filtering, and the most recent temporal models. For the cold-start users having few interactions with the items, the model adaptively utilizes the purely observed user (item) base properties and interaction features to adjust its predictions, which incorporates the key idea of feature-based algorithms. When the observed features are missing and non-informative, the model makes use of the user-item interaction patterns to make predictions, which is the strength of collaborative filtering algorithms. However, being different from the conventional matrix-factorization models, the latent user and item features in our model are entangled and able to co-evolve over time. Finally, the general temporal point process ingredient of the model makes it possible to capture the dynamic preferences of users to items and their recurrent interactions, which is more flexible and expressive.

3.3 Parameter Estimation

In this section, we propose an efficient framework to learn the parameters. A key challenge is that the objective function is non-convex in the parameters. However, we reformulate it as a convex optimization by creating new parameters. Finally, we present the generalized conditional gradient algorithm to efficiently solve the objective function.

Given a collection of events $\mathcal{T}$ recorded within a time window $[0, T)$, we estimate the parameters using maximum likelihood estimation of all events. The joint negative log-likelihood [54] is:

$$\ell = -\sum_{e_k} \log\left(\lambda^{u_k, i_k}(t_k)\right) + \sum_{u=1}^{m}\sum_{i=1}^{n}\int_0^T \lambda^{u,i}(\tau)\, d\tau \tag{3.4}$$

The objective function is non-convex in variables $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}\}$ due to the inner product term in (3.3). To learn these parameters, one way is to fix the matrix rank and update each matrix using gradient based methods. However, it is easily trapped in local optima and one needs to tune the rank for the best performance. However, with the observation that the product of two low rank matrices yields a low rank matrix, we will optimize over the new matrices and obtain a convex objective function.

3.3.1 Convex Objective Function

We will create new parameters such that the intensity function is convex. Since $\boldsymbol{u}_u(t)$ contains the averaging of $\boldsymbol{i}_{i_k}(t_k)$ in (3.1), $\boldsymbol{C}, \boldsymbol{D}$ will appear in $\boldsymbol{u}_u(t)$. Similarly, $\boldsymbol{A}, \boldsymbol{B}$ will appear in $\boldsymbol{i}_i(t)$. Hence we have the inner product of these matrices as follows:

$$\mathcal{X} = \left\{\boldsymbol{A}^\top\boldsymbol{A}, \boldsymbol{B}^\top\boldsymbol{B}, \boldsymbol{C}^\top\boldsymbol{C}, \boldsymbol{D}^\top\boldsymbol{D}, \boldsymbol{A}^\top\boldsymbol{B}, \boldsymbol{A}^\top\boldsymbol{C}, \boldsymbol{A}^\top\boldsymbol{D}, \boldsymbol{B}^\top\boldsymbol{C}, \boldsymbol{B}^\top\boldsymbol{D}, \boldsymbol{C}^\top\boldsymbol{D}\right\}$$

These matrices will appear in (3.3) after expansion, due to the inner product $\boldsymbol{i}_i(t)^\top \boldsymbol{u}_u(t)$. For each matrix product in $\mathcal{X}$, we denote it as a new variable $\boldsymbol{X}_i$ and optimize the objective function over the these variables. We denote the corresponding coefficient of $\boldsymbol{X}_i$ as $x_i(t)$, which can be exactly computed. Denote $\boldsymbol{\Lambda}(t) = (\lambda^{u,i}(t))$, we can rewrite the intensity in (3.3) in the matrix form as:

$$\boldsymbol{\Lambda}(t) = \boldsymbol{\eta} + \sum\nolimits_{i=1}^{10} x_i(t) \boldsymbol{X}_i \tag{3.5}$$

The intensity is convex in each new variable $\boldsymbol{X}_i$, hence the objective function. We will optimize over the new set of variables $\mathcal{X}$ subject to the constraints that i) some of them share the same low rank space, *e.g.*, $\boldsymbol{A}^\top$ is shared as the column space in $\left\{\boldsymbol{A}^\top\boldsymbol{A}, \boldsymbol{A}^\top\boldsymbol{B}, \boldsymbol{A}^\top\boldsymbol{C}, \boldsymbol{A}^\top\boldsymbol{D}\right\}$ and ii) new variables are low rank (the product of low rank matrices is low rank). Next, we show how to incorporate the space sharing constraint for general objective function with an efficient algorithm.

First, we create a *symmetric* block matrix $\boldsymbol{X} \in \mathbb{R}^{4D \times 4D}$ and place each $\boldsymbol{X}_i$ as follows:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \boldsymbol{X}_3 & \boldsymbol{X}_4 \\ \boldsymbol{X}_2^\top & \boldsymbol{X}_5 & \boldsymbol{X}_6 & \boldsymbol{X}_7 \\ \boldsymbol{X}_3^\top & \boldsymbol{X}_6^\top & \boldsymbol{X}_8 & \boldsymbol{X}_9 \\ \boldsymbol{X}_4^\top & \boldsymbol{X}_7^\top & \boldsymbol{X}_9^\top & \boldsymbol{X}_{10} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}^\top\boldsymbol{A} & \boldsymbol{A}^\top\boldsymbol{B} & \boldsymbol{A}^\top\boldsymbol{C} & \boldsymbol{A}^\top\boldsymbol{D} \\ \boldsymbol{B}^\top\boldsymbol{A} & \boldsymbol{B}^\top\boldsymbol{B} & \boldsymbol{B}^\top\boldsymbol{C} & \boldsymbol{B}^\top\boldsymbol{D} \\ \boldsymbol{C}^\top\boldsymbol{A} & \boldsymbol{C}^\top\boldsymbol{B} & \boldsymbol{C}^\top\boldsymbol{C} & \boldsymbol{C}^\top\boldsymbol{D} \\ \boldsymbol{D}^\top\boldsymbol{A} & \boldsymbol{D}^\top\boldsymbol{B} & \boldsymbol{D}^\top\boldsymbol{C} & \boldsymbol{D}^\top\boldsymbol{D} \end{pmatrix} \tag{3.6}$$

Intuitively, minimizing the nuclear norm of $\boldsymbol{X}$ ensures all the low rank space sharing constraints. First, nuclear norm $\|\cdot\|_*$ is a summation of all singular values, and is commonly used as a convex surrogate for the matrix rank function [77], hence minimizing $\|\boldsymbol{X}\|_*$ ensures it to be low rank and gives the unique low rank factorization of $\boldsymbol{X}$. Second, since $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4$ are in the same row and share $\boldsymbol{A}^\top$, the space sharing constraints are naturally satisfied.

Finally, since it is typically believed that users long-time preference to items can

be categorized into a limited number of prototypical types, we set $\boldsymbol{\eta}$ to be low rank. Hence the objective is:

$$\min_{\boldsymbol{\eta}\geqslant 0, \boldsymbol{X}\geqslant 0} \ \ell(\boldsymbol{X}, \boldsymbol{\eta}) + \alpha\|\boldsymbol{\eta}\|_* + \beta\|\boldsymbol{X}\|_* + \gamma\|\boldsymbol{X} - \boldsymbol{X}^\top\|_F^2 \tag{3.7}$$

where ℓ is defined in (3.4) and $\|\cdot\|_F$ is the Frobenius norm and the associated constraint ensures $\boldsymbol{X}$ to be symmetric. $\{\alpha, \beta, \gamma\}$ control the trade-off between the constraints. After obtaining $\boldsymbol{X}$, one can directly apply (3.5) to compute the intensity and make predictions.

3.3.2 Generalized Conditional Gradient Algorithm

We use the latest generalized conditional gradient algorithm [5] to solve the optimization problem (3.7). It has an alternating updates scheme and efficiently handles the nonnegative constraint using the proximal gradient descent and the the nuclear norm constraint using conditional gradient descent. It is guaranteed to converge in $O(\frac{1}{t} + \frac{1}{t^2})$, where t is the number of iterations. For both the proximal and the conditional gradient parts, the algorithm achieves the corresponding *optimal* convergence rates. If there is no nuclear norm constraint, the results recover the well-known optimal $O(\frac{1}{t^2})$ rate achieved by proximal gradient method for smooth convex optimization. If there is no nonnegative constraints, the results recover the well-known $O(\frac{1}{t})$ rate attained by conditional gradient method for smooth convex minimization. Moreover, the per-iteration complexity is linear in the total number of events with $O(mnk)$, where m is the number of users, n is the number of items and k is the number of events per user-item pair.

3.4 Experiments

We evaluate our framework on synthetic and real-world datasets. We use all the events up to time $T \cdot p$ as the training data, and the rest as testing data, where T is the length of the observation window. We tune hyper-parameters and the latent rank of other baselines using 10-fold cross validation with grid search. We vary the proportion $p \in \{0.7, 0.72, 0.74, 0.76, 0.78\}$ and report the averaged results over five runs on two tasks:

- Item recommendation: for each user u, at every testing time t, we compute the survival probability $S^{u,i}(t) = \exp\big(-\int_{t_n^{u,i}}^{t} \lambda^{u,i}(\tau)d\tau\big)$ of each item i up to time t, where $t_n^{u,i}$ is the last training event time of (u, i). We then rank all the items in the ascending order of $S^{u,i}(t)$ to produce a recommendation list. Ideally, the item associated with the testing time t should rank one, hence smaller value indicates better predictive performance. We repeat the evaluation on each testing moment and report the Mean Average Rank (MAR) of the respective testing items across all users.
- Time prediction: we predict the time when a testing event will occur between a given user-item pair (u, i) by calculating the density of the next event time as $f(t) = \lambda^{u,i}(t)S^{u,i}(t)$. With the density, we compute the expected time of next event by sampling future events as in [5]. We report the Mean Absolute Error (MAE) between the predicted and true time. Furthermore, we also report the relative percentage of the prediction error with respect to the entire testing time window.

3.4.1 Competitors

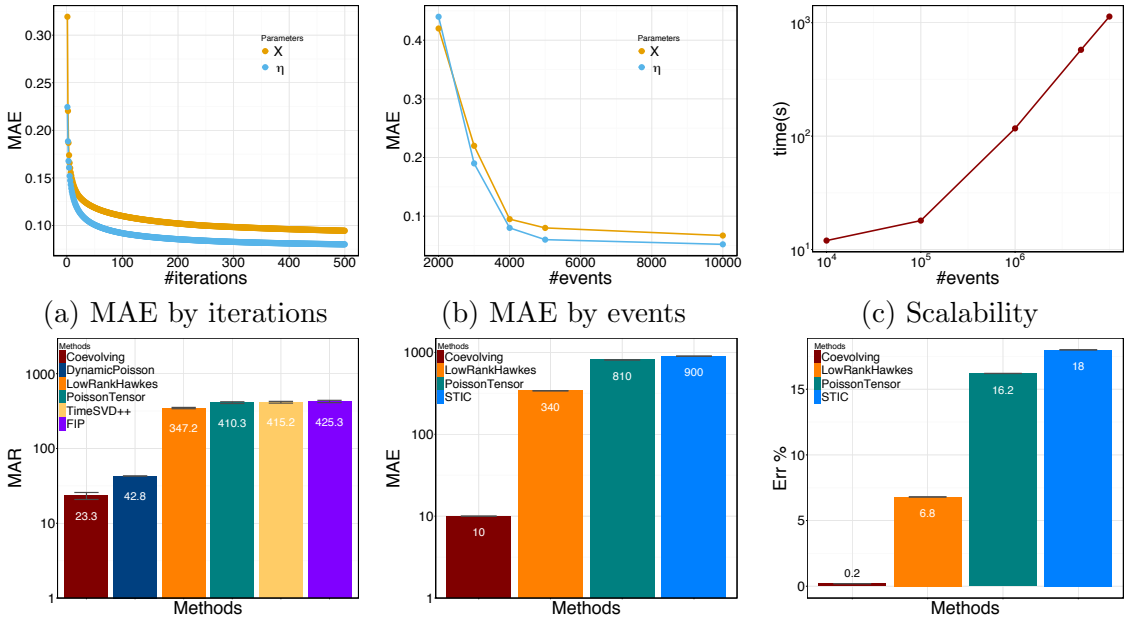
- TimeSVD++ is the classic matrix factorization method [10]. The latent factors of users and items are designed as decay functions of time and also linked to each other based on time.

- FIP is a static low rank latent factor model to uncover the compatibility between user and item features [68]. TSVD++ and FIP are only designed for data with explicit ratings. We convert the series of user-item interaction events into an explicit rating using the frequency of a users item consumptions [78].
- STIC fits a semi-hidden markov model to each observed user-item pair [79] and is only designed for time prediction.
- PoissonTensor uses Poisson regression as the loss function [61] and has been shown to outperform factorization methods based on squared loss [72, 73] on recommendation tasks. There are two choices of reporting performance: i) use the parameters fitted only in the last time interval and ii) use the average parameters over all intervals. We report the best performance between these two choices.
- LowRankHawkes is a Hawkes process based model and it assumes user-item interactions are independent [5].

3.4.2 Experiments on Synthetic Data

We simulate 1,000 users and 1,000 items. For each user, we further generate 10,000 events by Ogata's thinning algorithm. We compute the MAE by comparing estimated $\boldsymbol{\eta}, \boldsymbol{X}$ with the ground-truth. The baseline drift feature is set to be constant. Figure 3.2 (a) shows that it only requires a few hundred iterations to descend to a decent error, and (b) indicates that it only requires a modest number of events to achieve a good estimation. Finally, (c) demonstrates that our method scales linearly as the total number of training events grows.

Figure 3.2 (d-f) show that COEVOLVE achieves the best predictive performance. Because POISSONTENSOR applies an extra time dimension and fits each time interval as a Poisson regression, it outperforms TIMESVD++ by capturing the fine-grained temporal dynamics. Finally, our method automatically adapts different contributions of each past item factors to better capture the users' current latent features, hence it



(a) MAE by iterations (b) MAE by events (c) Scalability

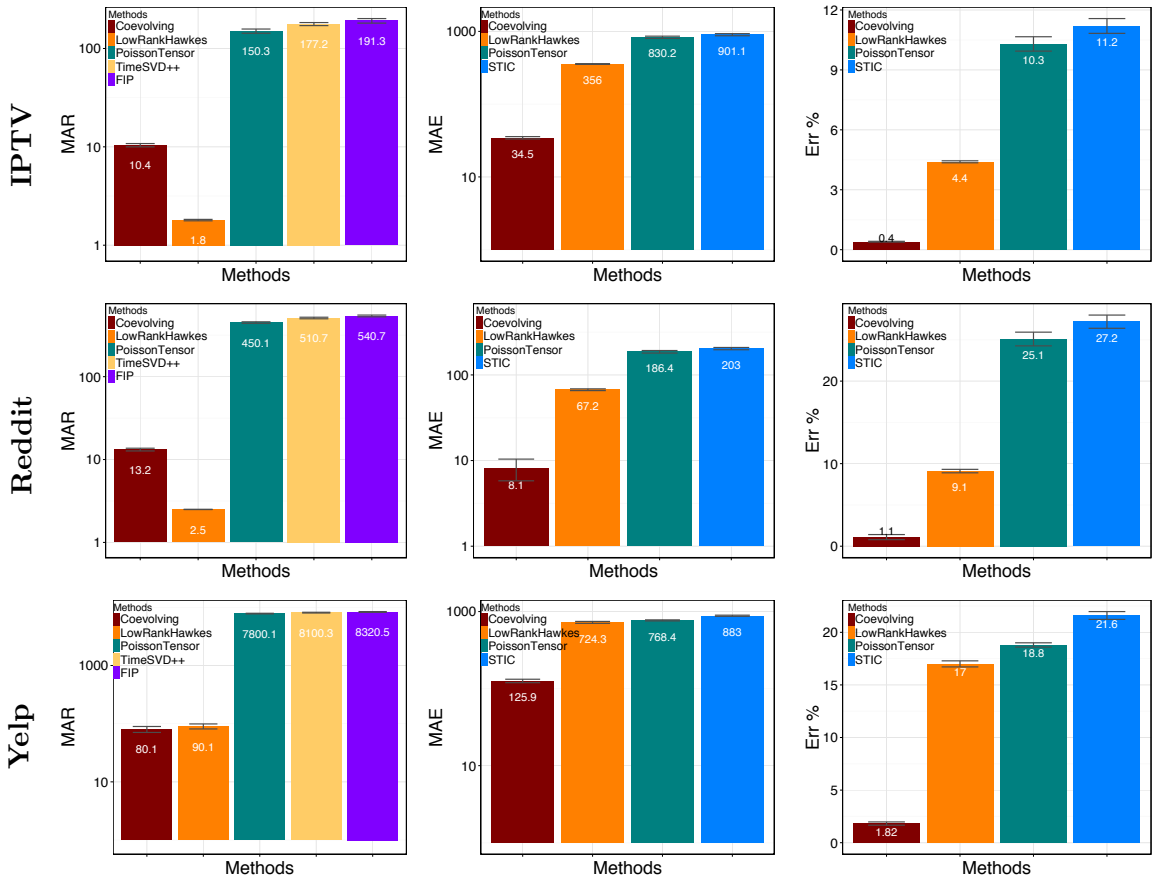
(d) Item recommendation (e) Time prediction (MAE) (f) Time prediction (relative)

Figure 3.2: Estimation error (a) vs. #iterations and (b) vs. #events per user; (c) scalability vs. #events per user; (d) average rank of the recommended items; (e) and (f) time prediction error.

can achieve the best prediction performance overall.

3.4.3 Experiments on Real-world Data

Datasets. Our datasets are obtained from three different domains from the TV streaming services (IPTV), the commercial review website (Yelp) and the online media services (Reddit). IPTV contains 7,100 users' watching history of 436 TV programs in 11 months, with 2,392,010 events, and 1,420 movie features, including 1,073 actors, 312 directors, 22 genres, 8 countries and 5 years. Yelp is available from Yelp Dataset challenge Round 7. It contains reviews for various businesses from October, 2004 to December, 2015. We filter users with more than 100 posts and it contains 100 users and 17,213 businesses with around 35,093 reviews. Reddit contains the discussions events in January 2014. Furthermore, we randomly selected 1,000 users and collect 1,403 groups that these users have discussion in, with a total of 10,000 discussion



(a) Item recommendation (b) Time prediction (MAE) (c) Time prediction (relative)

Figure 3.3: Prediction results on IPTV, Reddit and Yelp. Results are averaged over five runs with different portions of training data and error bar represents the variance.

events. For item base feature, IPTV has movie feature, Yelp has business description, and Reddit does not have it. In experiments we fix the baseline features. There is no base feature for user. For interaction feature, Reddit and Yelp have reviews in bag-of-words, and no such feature in IPTV.

Figure 6.5 shows the predictive performance. For time prediction, COEVOLVE outperforms the baselines *significantly*, since we explicitly reason and model the effect that past consumption behaviors change users' interests and items' features. In particular, compared with LowRankHawkes, our model captures the interactions of each user-item pair with a multi-dimensional temporal point processes. It is more expressive than the respective one-dimensional Hawkes process used by LOWRANKHAWKES,

which ignores the mutual influence among items. Furthermore, since the unit time is hour, the improvement over the state-of-art on IPTV is around two weeks and on Reddit is around two days. Hence our method significantly helps online services make better demand predictions.

For item recommendation, COEVOLVE also achieves competitive performance comparable with LowRankHawkes on IPTV and Reddit. The reason behind the phenomena is that one needs to compute the rank of the intensity function for the item prediction task, and the value of intensity function for time prediction. LowRankHawkes might be good at differentiating the rank of intensity better than COEVOLVE. However, it may not be able to learn the actual value of the intensity accurately. Hence our method has the order of magnitude improvement in the time prediction task.

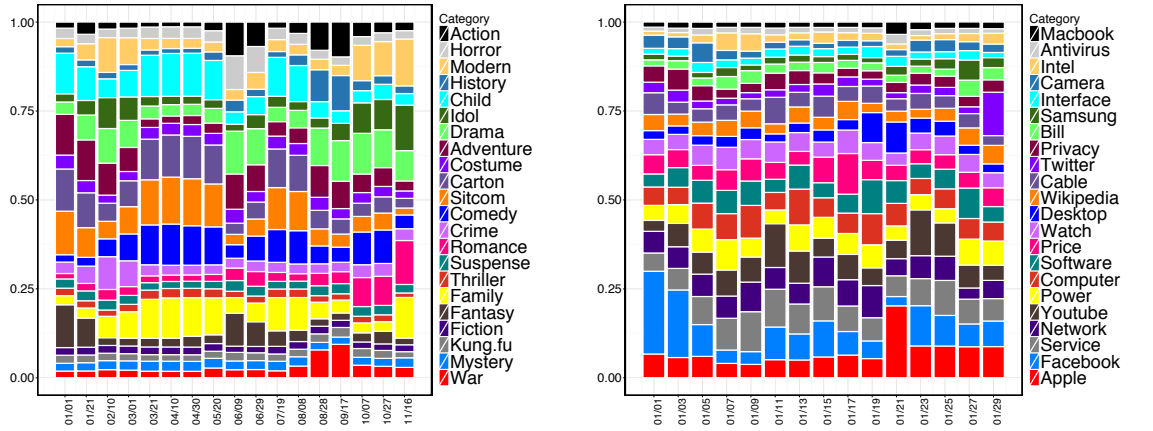
In addition to the superb predictive performance, COEVOLVE also learns the time-varying latent features of users and items. Figure 3.4 (a) shows that the user is initially interested in TV programs of adventures, but then the interest changes to Sitcom, Family and Comedy and finally switches to the Romance TV programs. Figure 3.4 (b) shows that Facebook and Apple are the two hot topics in the month of January 2014. The discussions about Apple suddenly increased on 01/21/2014, which can be traced to the news that Apple won lawsuit against Samsung[1]. It further demonstrates that our model can better explain and capture the user behavior in the real world.

3.5 Summary

In this chapter, we have proposed an efficient framework for modeling the co-evolution nature of users' and items' latent features. Empirical evaluations on large synthetic and real-world datasets demonstrate its scalability and superior predictive performance.

Our COEVOLVE framework is based on parametric embeddings for users and items

[1] http://techcrunch.com/2014/01/22/apple-wins-big-against-samsung-in-court/



(a) A user's feature in IPTV (b) The Technology group's feature in Reddit

Figure 3.4: Learned time-varying features of a user in IPTV and a group in Reddit.

latent features. A natural extension is to combine our framework with the power of recurrent neural networks, and learn nonlinear embeddings for users and items. In the next chapter, we extend our framework and develop a generic embedding framework which is applicable to many applications that involve the continuous-time evolving graphs, such as time-sensitive recommendation and knowledge graph reasoning.

CHAPTER 4
A GENERIC EMBEDDING FRAMEWORK FOR CONTINUOUS-TIME EVOLVING GRAPHS

4.1 Introduction

Many real world applications involves continuous-time evolving graphs, where the dynamic edges of the graphs are relational events between entities, and the precise time stamps of these events are recorded in the datasets. For instance, in online discussion forums such as reddit, users post different topics in different groups at difference times, and each post is a dynamic edge between a user and a group. In online e-commerce sites such as Amazon and Alibaba, users purchases different items and may leave comments for the item, and each shopping event is a dynamic edge between a user and an item. In temporal knowledge graphs, different celebrities, organizations and countries form different relations at different times, such as visiting each other and signing economic agreement, which are the dynamic edges in the graphs. In crime behavior analysis, criminal agents commit burglaries at different locations, and each burglary event is a dynamic edge between the criminal agent and the location.

Common to these applications are the evolutionary and coevolutionary nature of the underlying entities:

- In recommendation systems, as users interact with different items, their interactions further drives the evolution of user interests and item features. The users' interests and items' features can also coevolve over time, *i.e.*, their features are intertwined and can influence each other. In forums such as Reddit, users with very different interest profiles can join the same group and contribute discussions. Thus the

topic of this group may drift over time. On the other hand, users' interests will also change due to the evolution of groups' topics.

- In a temporal knowledge graph, when two entities forge a relationship, the newly formed edge drives their preferences and behaviors. For instance, two countries forging an alliance are most likely to take confrontational stands against enemies of each other. As the saying goes that *"people (entities) change over time"* and *"relationships change over time"*, such change is effected by combination of historical factors (self-evolving) and the factors of the other entities (coevolving).

The complexity of such interaction patterns and fine temporal resolution of the relational events also raises interesting and challenging questions: How to represent each entity in the graph? How to deal with multi-edges between a pair of entities? How to take into account precise timing information?

Recently, point process based models have also been widely applied to capture the evolving nature of continuous-time event data [1, 2, 4, 5, 8, 6, 7, 9]. However, these works make strong parametric assumptions about the functional form of the generative processes, which may not be expressive enough for real world complex scenarios; hence they are not accurate enough to capture the complex and *nonlinear* co-evolution of entity features in real world. Our framework does not make these restricted assumptions and can adapt to the complexity of coevolutionary dynamics.

In this chapter, we treat the data as a continuous-time evolving graph, and propose a novel deep coevolutionary network model (DeepCoevolve) which defines deep learning architectures based on the unrolled evolving interaction graphs. Our model provide an effective and expressive representation/embedding of the underlying entity latent features, without assuming a fixed parametric form. In particular, our work makes the following contributions:

- **Novel framework.** We propose a novel deep learning framework that captures the *nonlinear* coevolution nature of entities' embeddings in a temporal dynamic

graph using a nonparametric way. It assigns an evolving feature embedding process for each entity, and the coevolution of these latent feature processes is naturally modeled according to the pairwise interactions between entities. Furthermore, with the fine grained model for event at each individual time point, we are able to predict the time of events.

- **Efficient training.** We use RNN to parametrize the interdependent and intertwined entity embeddings. The increased flexibility and generality further introduces technical challenges on how to train RNN on the *evolving networks*. The coevolution nature of the model makes the samples inter-dependent and not identically distributed, which is significantly more challenging than the typical assumptions in training deep models. We propose an efficient stochastic training algorithm that makes the BTPP tractable in the co-evolving network.
- **Strong performance.** We show that our framework has superior performance in diverse applications, including time-sensitive recommendation systems and temporal knowledge graph reasoning. Our method leads to significant improvements in user preference prediction on recommendation tasks, and in knowledge graph link prediction. The precision of time prediction is improved significantly. It verifies the importance of using nonparametric point process in temporal graphs. Time prediction is especially novel and not possible by most prior works.

4.2 Deep Coevolutionary Feature Embedding

In this section, we present a generative process for continuous-time evolving graphs based on a novel Deep Coevolutionary Network (DeepCoevolve). The high level idea of our method is to define deep learning architectures based on an unrolled evolving interaction graph, and this deep learning architecture computes a coevolutionary embedding representation for each entity in the graph. Then, based on the compatibility between the entity latent embeddings, we model the interaction/relational events by

a multi-dimensional temporal point process.

4.2.1 Continuous-time Evolving Graphs

We first describe how continuous-time evolving graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{Y})$ can be represented as a sequence of continuous-time events. Here $\mathcal{V}$ is the set of entities indexed from 1 to M, where $|\mathcal{V}| = M$. The notation $\mathcal{Y}$ is the set of all possible edge types indexed from 1 to Y, where $|\mathcal{Y}| = Y$. A directed edge $\boldsymbol{e}_n = (s_n, o_n, r_n, t_n, \boldsymbol{q}_n)$ represents the interaction with event type $r_n \in \mathcal{Y}$ between subject entity s_n and object entity o_n at time t_n. Here the optional component $\boldsymbol{q}_n \in \mathbb{R}^q$ represents the interaction context, *e.g.*, a text review in restaurant recommendation. Then the set of edges $\mathcal{E}$ contains N dynamic edges that occur between entities. Without loss of generality, $\mathcal{E} = \{\boldsymbol{e}_n = (s_n, o_n, r_n, t_n, \boldsymbol{q}_n)\}_{n=1}^N$ is ordered by time in the observation window $[0, T]$. Here $s_n, o_n \in \{1, \ldots, M\}$, $t_j \in \mathbb{R}^+$ and $0 \leqslant t_1 \leqslant t_2 \ldots \leqslant T$.

In the above event representation of continuous-time coevolving graphs, the temporal information is preserved precisely. Furthermore, this event representation also preserves the ordering of the occurrence of dynamic edges, and it allows multiple edges with potentially different types to be formed between a pair of entities.

4.2.2 Unrolling Continuous-time Coevolving Graphs

In order to define the deep coevolutionary network, we will design a key network transformation in this section (Figure 4.1). We call this transformation the unrolling of a continuous-time coevolving graph. Since the dynamic edges in our continuous-time evolving graph is ordered, we will unroll the graph by operating on one dynamic edge event at a time. Every time when a new event happens, the node representations of each entity involved in the event will be duplicated to reflect the evolution. Specifically, in a high level, each such edge event will lead to four directed edges in the unrolled graph, and two duplicated nodes in the unrolled graph. Two edges will

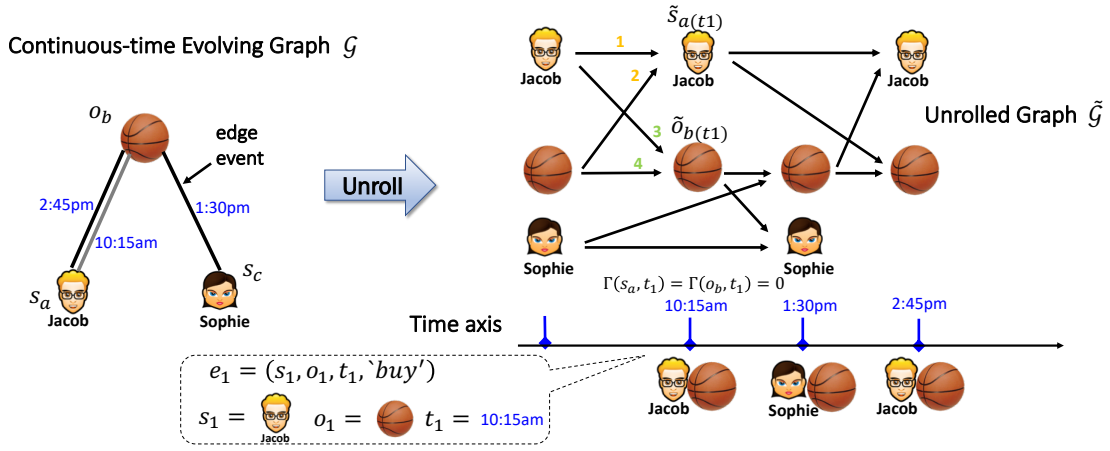


Figure 4.1: Unrolling transformation for a continuous-time evolving graph. Each dynamic edge events in the time-evolving graph will lead to 4 directed edges in the unrolled graph, and 2 duplicated nodes in the unrolled graph. For instance, edge event (Jacob, ball, 10.15am, purchase) will be unrolled into edge 1, 2, 3, 4 in the figure and a duplicated Jacob node and a ball node.

point from the latest nodes to the corresponding duplicated nodes, and the other two edges will point across. More specifically, the algorithm to produce the unrolled graph is as follows:

1. Given a continuous-time evolving graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{Y})$.
2. Initialize the unrolled graph as $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}} = \emptyset, \widetilde{\mathcal{E}} = \emptyset)$.
3. $t_0 = 0$.
4. For each $v \in \mathcal{V}$,

   Create its duplication $\widetilde{v_{(0)}}$ and let $\widetilde{\mathcal{V}} = \widetilde{\mathcal{V}} \cup \widetilde{v_{(0)}}$.

5. For $n = 1 \ldots, N$,

   Retrieve the corresponding dynamic edge event $e_n = (s_n, o_n, r_n, t_n, q_n) \in \mathcal{E}$;

   Relabel the nodes with time stamp as $s_n \mapsto \widetilde{s}_{n(t_n)}$ and $o_n \mapsto \widetilde{o}_{n(t_n)}$ respectively;

   Add node $\widetilde{\mathcal{V}} = \widetilde{\mathcal{V}} \cup \{\widetilde{s}_{n(t_n)}, \widetilde{o}_{n(t_n)}\}$;

Add edge

$$\widetilde{\mathcal{E}} = \widetilde{\mathcal{E}} \cup \{ \quad < \widetilde{s}_{n(t_n-)}, \quad \widetilde{s}_{n(t_n)} >, \qquad < \widetilde{o}_{n(t_n-)}, \quad \widetilde{o}_{n(t_n)} >,$$
$$< \widetilde{s}_{n(t_n-)}, \quad \widetilde{o}_{n(t_n)} >, \qquad < \widetilde{o}_{n(t_n-)}, \quad \widetilde{s}_{n(t_n)} > \}$$

* here the notation $\widetilde{v}_{(t-)} = \widetilde{v}_{(\Gamma(v,t))}$, where $\Gamma(v,t) = \arg\max\{\tau : \tau < t, \widetilde{v_{(\tau)}} \in \tilde{\mathcal{V}}\}$. In other words, it refers to the latest duplication of node v before time t, and $\Gamma(v,t)$ is the time stamp of the corresponding event.

The benefit of such unrolling operation is that the unrolled graph still preserve order of the dynamic edge events. Furthermore, the horizontal edges capture self-evolution of entities, while cross edges capture the coevolution/mutual influence between entities, allowing information to pass along the unrolled network. One can also make analogue between the unrolled graph to dynamic Bayes Networks from graphical model. Instead of defining dynamic Bayes Networks, we will define our deep coevolutionary network architecture based the unrolled graph.

4.2.3 Deep Coevolutionary Networks

In our framework, we user the low-dimension vector $\boldsymbol{f}_v(t) \in \mathbb{R}^d$ to represent the latent feature embedding of entity $v \in \mathcal{V}$ at time t. As we described in previous sections, these embedding vectors will change over time. Specifically, we model the evolution of embeddings using two symmetric update functions, $g^{(s)}$ and $g^{(o)}$, for subject and object entities, respectively. These embeddings are initialized as 0, and then the updates are carried out whenever an event happens between pairs of entities. The amount of the updates are determined by neural networks which aggregates historical information of entity embeddings, interaction time and context.

With the help of unrolled coevolving graph, the continuous time evolution of each node embeddings can be characterized as the embedding of each duplication node in

the unrolled graph $\widetilde{\mathcal{G}}$. Each time when an event happens, the embeddings of those involved entities will change. So it suffices to define the evolution after each event. The abstract form of the two parallel feature embedding updates are as follows.

DeepCoevolve:

When an event $e_n = (s_n, o_n, r_n, t_n, \boldsymbol{q}_n)$ happens, the newly created duplication $\widetilde{s}_{n(t_n)}$ and $\widetilde{o}_{n(t_n)}$ carries the evolution embedding of node s_n and o_n in the following way:

Subjects' embedding update. Embedding of $\widetilde{s}_{n(t_n)} \in \widetilde{\mathcal{V}}$ corresponds to the embedding of entity s_n at time t_n, thus we have:

$$f(\widetilde{s}_{n(t_n)}) = f_{s_n}(t_n) = g^{(s)}\Bigg(\underbrace{(t_n - \Gamma(s_n, t_n))}_{\text{drift}}, \quad \underbrace{\boldsymbol{f}(\widetilde{s}_{n(t_n-)})}_{\text{self evolution}}, \quad \underbrace{\boldsymbol{f}(\widetilde{o}_{n(t_n-)})}_{\text{object feature}}, \quad \underbrace{r_n, \boldsymbol{q}_n}_{\text{interaction}} \Bigg) \tag{4.1}$$

Objects' embedding update. Embedding of $\widetilde{o}_{n(t_n)} \in \widetilde{\mathcal{V}}$ corresponds to the embedding of entity o_n at time t_n, thus we have:

$$f(\widetilde{o}_{n(t_n)}) = f_{o_n}(t_n) = g^{(o)}\Bigg(\underbrace{(t_n - \Gamma(o_n, t_n))}_{\text{drift}}, \quad \underbrace{\boldsymbol{f}(\widetilde{o}_{n(t_n-)})}_{\text{self evolution}}, \quad \underbrace{\boldsymbol{f}(\widetilde{s}_{n(t_n-)})}_{\text{subject feature}}, \quad \underbrace{r_n, \boldsymbol{q}_n}_{\text{interaction}} \Bigg) \tag{4.2}$$

Each of the two equation incorporates four terms: temporal drift, self evolution, coevolution and interaction features. The rationale for designing these terms is explained below:

Temporal drift. The first term is defined based on the time difference between consecutive events of specific entity. It allows the entities' basic feature (*e.g.*, user profile) to smoothly change over time. Such changes of basic features normally are caused by external influences.

Self evolution. The current entity feature should also be influenced by its own feature at the earlier time. This captures the intrinsic evolution of entity features.

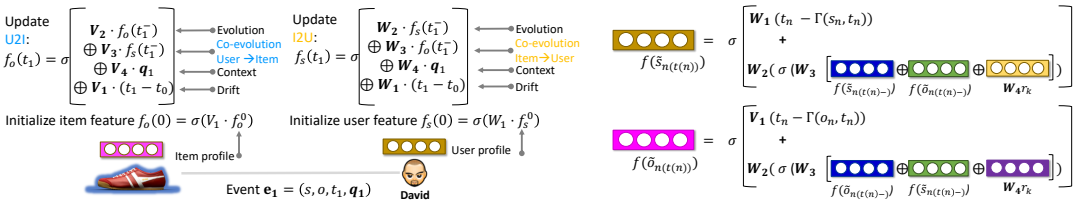


Figure 4.2: Parameterization adapted for different applications. (a) Coevolutionary process for user/item embeddings. (b) Coevolutionary process for subject and object embeddings in temporal knowledge graph.

For example, in recommendation systems, a user's current interest should be related to his/her interest two days ago; in knowledge graphs, a country's status is highly dependent to its diplomatic history.

Entity coevolution. This term captures the phenomenon that subject and object entity embeddings are mutually dependent on each other. The coevolving examples in many applications including recommendation and knowledge completion is explained in Section 4.1.

Interaction features. The interaction feature is the additional information happened in the entity interactions. For example, in online discussion forums such as Reddit, the interaction features are the posts and comments. In temporal knowledge graphs, the features are the actual relation type. This term models influence of interaction context. For example, if two countries have tense relationships, they are more likely to engage in conflicts.

Examples instantiations. We will elaborate how to apply the general DeepCoevolve framework to different applications, including recommendation systems and temporal knowledge graph reasoning. We show that by instantiating (4.1) and (4.2), we can easily apply the general framework to these applications. Figure 4.2 summarizes the adaptation to each application.

Time sensitive recommendation

In the temporal recommendation system, each event $\boldsymbol{e}_n = (s_n, o_n, r_n, t_n, \boldsymbol{q}_n)$ represents an interaction with type r_n between user s_n and item o_n at time t_n. The rich feature $\boldsymbol{q}_n$ can be the review text posted by the user, or simply some background information of corresponding user/item. In this setting we have $|\mathcal{Y}| = 1$, *i.e.*, only one event type. Thus we omit r_n here. Since it is essentially a bipartite graph, the users will only appear as subject, while items will only appear as object in this event.

Specifically, we use the following form to parameterize the users' and items' embedding process:

Users' embedding process:

$$f(\widetilde{s}_{n(t_n)}) = f_{s_n}(t_n) = \sigma\Big(\boldsymbol{W}_1(t_n - \Gamma(s_n, t_n)) + \boldsymbol{W}_2 f(\widetilde{s}_{n(t_n-)}) + \boldsymbol{W}_3 f(\widetilde{o}_{n(t_n-)}) + \boldsymbol{W}_4 \boldsymbol{q}_n\Big) \tag{4.3}$$

Items' embedding process:

$$f(\widetilde{o}_{n(t_n)}) = f_{o_n}(t_n) = \sigma\Big(\boldsymbol{V}_1(t_n - \Gamma(o_n, t_n)) + \boldsymbol{V}2 f(\widetilde{o}_{n(t_n-)}) + \boldsymbol{V}_3 f(\widetilde{s}_{n(t_n-)}) + \boldsymbol{V}_4 \boldsymbol{q}_n\Big) \tag{4.4}$$

Here the parameter set is $\theta = \{\boldsymbol{W}_{1-4}, \boldsymbol{V}_{1-4}\}$, where $\boldsymbol{W}_4, \boldsymbol{V}_4 \in \boldsymbol{R}^{d \times q}$ are the embedding matrices mapping from the explicit high-dimensional feature space into the low-rank latent feature space and $\boldsymbol{W}_1, \boldsymbol{V}_1 \in \mathbb{R}^d$, $\boldsymbol{W}_2, \boldsymbol{V}_2, \boldsymbol{W}_3, \boldsymbol{V}_3 \in \mathbb{R}^{d \times d}$ are weights parameters. $\sigma(\cdot)$ is activation function, such as commonly used Tanh or Sigmoid function. For simplicity, we use basic recurrent neural network to formulate the recurrence structure, but it is also straightforward to design more sophisticated structured using GRU or LSTM to gain more expressive power.

In this application, the edge type r_n in event $\boldsymbol{e}_n$ encodes the actual relation type between subject and object. Each entity $v \in \mathcal{V}$ can either be subject or object. The feature update of entity v depends on whether it appears as subject or object which helps to capture the direction:

Subjects' embedding process: if entity appears as subject,

$$f(\widetilde{s}_{n(t_n)}) = f_{s_n}(t_n) = \sigma\Big(\mathbf{W_t^s}(t_n - \Gamma(s_n, t_n)) + \mathbf{W^{hh}}\sigma\big(\mathbf{W^h}[f(\widetilde{s}_{n(t_n-)}) \oplus f(\widetilde{o}_{n(t_n-)}) \oplus \boldsymbol{W}^r r_n]\big)\Big) \tag{4.5}$$

where $\boldsymbol{A} \oplus \boldsymbol{B} := [\boldsymbol{A}, \boldsymbol{B}]$ represents the matrix concatenation operator.

Objects' embedding process: if entity appears as object,

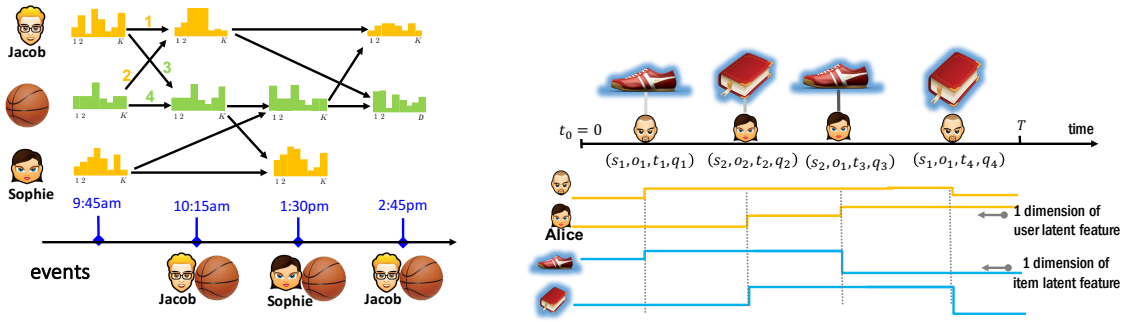
$$f(\widetilde{o}_{n(t_n)}) = f_{o_n}(t_n) = \sigma\Big(\mathbf{W_t^o}(t_n - \Gamma(o_n, t_n)) + \mathbf{W^{hh}}\sigma\big(\mathbf{W^h}[f(\widetilde{o}_{n(t_n-)}) \oplus f(\widetilde{s}_{n(t_n-)}) \oplus \boldsymbol{W}^r r_n]\big)\Big) \tag{4.6}$$

The parameter set is $\theta = \{\boldsymbol{W}_t^s, \boldsymbol{W}_t^o, \boldsymbol{W}^{hh}, \boldsymbol{W}^h\}$, where $\mathbf{W_t^s}, \mathbf{W_t^o} \in \mathbb{R}^d$, $\mathbf{W^{hh}} \in \mathbb{R}^{d \times l}$ and $\mathbf{W^h} \in \mathbb{R}^{l \times (2d+c)}$ are weight parameters in network learned during training. $\mathbf{W_t^s}, \mathbf{W_t^o}$ captures variation in temporal drift for subject and object respectively. $\mathbf{W^{hh}}$ is shared parameter that captures recurrent participation effect for each entity. $\mathbf{W^h}$ is a shared projection matrix applied to consider the compatibility of entities in their previous relationships. $\mathbf{W^r} \in \mathbb{R}^{c \times r}$ is the relationship embedding matrix (r is # of relationships).

4.2.4 Understanding Coevolutionary Embeddings

Although the recurrent updates in (4.1) and (4.2) involve only the subject and object entity pairs directly participating in that specific interaction, the influence of a particular entity can propagate very far into the entire graph.

For example, in recommendation system, a user's feature embedding can influence



(a) Graph of embedding computation (b) Piecewise-constant embedding process

Figure 4.3: (a) The arrows indicate the dependency structures in the embedding updates, *e.g.*, Jacob interacts with basketball at 10:15am. Then the feature embeddings are updated: the new feature embedding at 10:15 am is influenced by his previous feature embedding and the basketball's previous feature embedding at 9:45am (arrow 1 and 2); the basketball's feature embedding is also influenced by Jacob's previous feature embedding and its previous embedding feature (arrow 3 and 4). (b) A user or item's feature embedding is piecewise constant over time and will change *only* after an interaction event happens. Only one dimension of the feature embedding is shown.

the item feature embedding he directly interacts with, then modified item feature embedding can influence a different user who purchases that item in a future interaction event, and so on and so forth through the entire network. Such cascading effect is illustrated in Figure 4.3(a).

Since the feature embedding updates are event driven, the entities' feature embedding processes are piecewise constant functions of time. These embeddings are *changed only if an interaction event happens*. In Figure 4.3(b), a user's attribute changes only when he has a new interaction with some item. This is reasonable since a user's taste for music changes only when he listens to some new or old musics. Similarly, an item's attribute changes only when some user interacts with it.

4.2.5 Intensity Function as the Compatibility between Embeddings

In our paper, we use the generative multidimensional point process to capture the underlying properties in temporal graphs. Specifically, we build $M \times M \times Y$ dimensional process. Each dimension can be represented by a triplet $(s, o, r) \in \mathcal{V} \times \mathcal{V} \times \mathcal{Y}$.

Mathematically, we model the intensity function in the (s, o, r)-th dimension (subject s and object o with type r) as a Rayleigh process:

$$\lambda^{s,o,r}(t|t') = \underbrace{\exp\left(\boldsymbol{f}_s(t')^\top \boldsymbol{R}^r \boldsymbol{f}_o(t')\right)}_{\text{subject-object-edge\_type compatibility}} \cdot \underbrace{(t - t')}_{\text{time lapse}} \tag{4.7}$$

where $t > t'$, and t' is the last time point where any entity's embedding changes before time t. $\boldsymbol{R}^r \in \mathbb{R}^{d \times d}, \forall r \in \mathcal{Y}$ is used for bilinear computation of similarity. The rationale behind this formulation is as follows:

Time as a random variable. Instead of discretizing the time into epochs as traditional methods [11, 71, 74, 75, 76], we explicitly model the timing of each interaction event as a random variable, which naturally captures the heterogeneity of the temporal interactions between entities.

Short term preference. The probability for subject s to interact with object o depends on the compatibility of their instantaneous embeddings and the event type r, which is evaluated through the bilinear transformation at the last event time t'. Because $\boldsymbol{f}_s(t)$ and $\boldsymbol{f}_o(t)$ co-evolve through time, their inner-product measures a general representation of the cumulative influence from the past interactions to the occurrence of the current event. The $\exp(\cdot)$ function ensures the intensity is positive and well defined.

Rayleigh time distribution. The entity embeddings are *piecewise constant*, and we use the time lapse term to make the intensity *piecewise linear*. This form leads to a Rayleigh distribution for the time intervals between consecutive events in each dimension. It is well-adapted to modeling fads [54], where the likelihood of generating an event rises to a peak and then drops extremely rapidly. Furthermore, it is computationally easy to obtain an analytic form of this likelihood. One can then use it to make item recommendation by finding the dimension that the likelihood function reaches the peak.

4.3 Efficient Learning for Deep Coevolutionary Network

In this section, we will first introduce the objective function, and then propose an efficient learning algorithm for the generative multidimensional point process model.

4.3.1 Objective Function

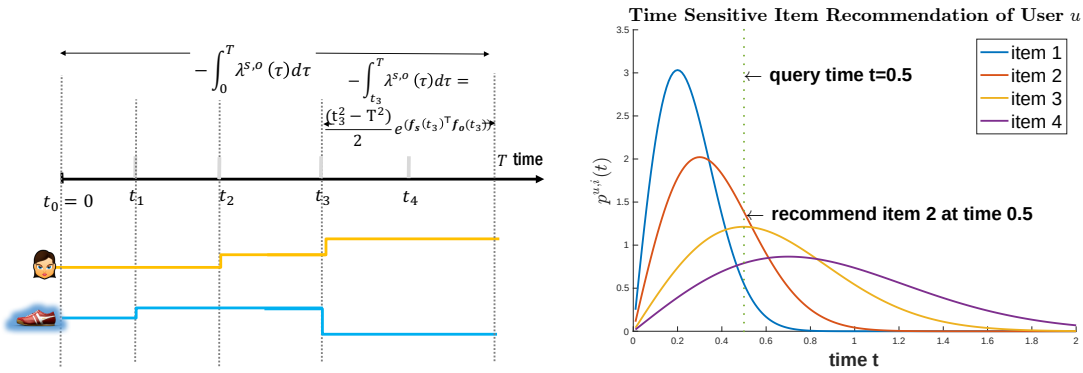
With the parameterized intensity function in (4.7), we can sample events according to it. Due to the interdependency between the feature embeddings and the propagation of influence over the interaction network, the different dimensions of the point process can intricate dependency structure. Such dependency allows sophisticated feature diffusion process to be modeled.

Given a sequence of events observed in real world, we can further estimate the parameters of the model by maximizing the likelihood of these observed events. Given a set of N events, the joint negative log-likelihood can be written as [60]:

$$\ell = \underbrace{-\sum_{j=1}^{N} \log\left(\lambda^{s_n,o_n,r_n}(t_n|t'_n)\right)}_{\text{happened events}} + \underbrace{\sum_{s=1}^{M}\sum_{o=1}^{M}\sum_{r=1}^{Y}\int_0^T \lambda^{s,o,r}(\tau|\tau')\,\mathrm{d}\tau}_{\text{survival of not-happened events}} \tag{4.8}$$

We can interpret it as follows: (i) the negative intensity summation term ensures the probability of all interaction events is maximized; (ii) the second survival probability term penalizes the *non-presence* of an interaction between all possible triplets on the observation window. Hence, our framework not only explains why an event happens, but also why an event did not happen.

Due to the co-evolution nature of our model, it is a very challenging task to learn the model parameters since the temporal dynamic graph is time-varying. Next, we will design an efficient learning algorithm for our model.



(a) Survival probability computation (b) Item recommendation

Figure 4.4: (a) Survival probability for a user s and item o. The integral $\int_0^T \lambda^{s,o}(\tau)\mathrm{d}\tau$ is decomposed into four inter-event intervals separated by $\{t_0, \cdots, t_3\}$. (b) Item recommendation using the score of likelihood between user and item pairs (s, o) at specific time t.

4.3.2 Efficient Learning Algorithm

We propose an efficient algorithm to learn the set of model parameter θ and $\{\boldsymbol{R}_i\}_{i=1}^Y$. The Back Propagation Through Time (BPTT) is the standard way to train a RNN. To make the back propagation tractable, one typically needs to do truncation during training. However, due to the novel co-evolutionary nature of our model, all the events are related to each other by the temporal dynamic graph, which makes it hard to decompose.

Hence, in sharp contrast to works [80, 81] in sequential data where one can easily break the sequences into multiple segments to make the BPTT trackable, it is a *challenging* task to design BPTT in our case. To efficiently solve this problem, we first order all the events globally and then do mini-batch training in a sliding window fashion. Each time when conducting feed forward and back propagation, we take the consecutive events within current sliding window to build the dynamic computational graph. Thus in our case the truncation is on the global timeline, compared with the trucation on individual independent sequences in prior works.

Next, we explain our procedure in detail. Given a mini-batch of K ordered events

$\tilde{\mathcal{O}} = \{e_n\}_{n=1}^K$, we set the time span to be $[T_0 = t_1, T = t_K]$. Below we show how to compute the intensity and survival probability term in the objective function (4.8) respectively.

Computing the intensity function. Each time when a new event $\boldsymbol{e}_n$ happens between user s_n and item o_n, their corresponding feature embeddings will evolve according to a computational graph. Figure 4.3(a) shows an illustration in the recommendation system setting. Due to the change of feature embedding, all the dimensions related to s_n or o_n are also influenced and the intensity functions for these dimensions change consequently. In our implementation, we first compute the corresponding intensity $\lambda^{s_n,o_n,r_n}(t_n|t'_n)$ according to (4.7), and then update the embedding of s_n and o_n. This operation takes $O(K)$ complexity, and is independent to the number of entities.

Computing the survival function. To compute $-\int_{T_0}^{T} \lambda^{s,o,r}(\tau|\tau')\mathrm{d}\tau$ for each triplet (s, o, r), we first collect all the time stamps $\{t_k\}$ that have events related to either s or o. For notation simplicity, let $|\{t_k\}| = n_{s,o}$ and $t_1 = T_0, t_{n_{s,o}} = T$. Since the embeddings are piecewise constant, the corresponding intensity function is piecewise linear according to (4.7). Thus, the integration is decomposed into each time interval where the intensity is linear, *i.e.*,

$$\int_{T0}^{T} \lambda^{s,o,r}(\tau|\tau')\mathrm{d}\tau = \sum_{k=1}^{n_{s,o,r}-1} \int_{t_k}^{t_{k+1}} \lambda^{s,o,r}(\tau|\tau')\mathrm{d}\tau \tag{4.9}$$

$$= \sum_{k=1}^{n_{s,o}-1} (t_{k+1}^2 - t_k^2) \exp\left(\boldsymbol{f}_s(t_k)^\top \boldsymbol{R}^r \boldsymbol{f}_o(t_k)\right) \tag{4.10}$$

Figure 4.4(a) illustrates the details of computation in the recommendation system setting.

Although the survival probability term exists in closed form, it is still expensive to compute it for each possible triplet. Moreover, since such interaction in the temporal dynamic graph is very sparse, it is not necessary to monitor each dimension in the

Algorithm 4 Learning Coevolutionary Process

Input: Dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{Y})$
Initialize parameters in the set θ randomly.
for $i = 1$ **to** max iterations **do**
  Sample K consecutive event edges $\{e_n\}_{n=k}^{k+K}$ uniform randomly from $\mathcal{G}$.
  Construct unrolled graph $\tilde{\mathcal{G}}$ for the current mini-batch events as in Sec 4.2.2.
  for $j = k$ **to** $k + K$ **do**
    Update embeddings of $f(\widetilde{s}_{j(t_j)})$ and $f(\widetilde{o}_{j(t_j)})$ according to Eq 4.1 and Eq 4.2.
  end for
  Compute loss function $\tilde{l}$ which approximates the true data likelihood in Eq 4.8.
  Update $\theta^i = \theta^{i-1} + \lambda_i \nabla_{\theta^{i-1}} \tilde{l}$.
end for
return θ

stochastic training setting. To speed up the computation, we use a random-sampling scheme as follows.

Note that the intensity term in the objective function (4.8) tries to maximize the bilinear term among (subject, object, type) triplet that has interaction event, while the survival term normalizes over all such triplets. We observe that this is similar to Softmax computing for classification problem. Hence, inspired by the noise-contrastive estimation method (NCE) [82] that is widely used in language models [83], we keep the dimensions that have events on them, while randomly sample dimensions without events in current mini-batch to speed up the computation.

Finally, another challenge in training lies in the fact that the entity interactions vary a lot across mini-batches, hence the corresponding computational graph also changes greatly. To make the training efficient, we use the graph embedding framework [84] which allows training deep learning models where each term in the objective has a different computational graphs but with shared parameters. The Adam Optimizer [85] and gradient clip is used in our experiment.

4.4 Prediction with DeepCoevolve

Since we use DeepCoevolve to model the intensities of multivariate point processes, our model can make two types of predictions, namely the entity prediction and event time prediction. The precise event time prediction is especially novel and not possible by most prior work. More specifically,

Entity prediction. Many applications can be modeled as entity prediction in a graph. The key to the prediction is a ranking function which uses the conditional likelihood

$$p^{s,o,r}(t) = \lambda^{s,o,r}(t) S^{s,o,r}(t) \tag{4.11}$$

Next we discuss two applications:

- *Next item prediction in recommendation system.* Given a pair of user and time (s, t), we aim at answering the following question: *what is the item the user s will interact at time t?* To answer this problem, we rank all items in the descending order in term of the value of the corresponding conditional density at time t, and the best prediction is made to the item with the largest conditional density. Using this formulation, at different point in time, a different prediction/recommendation can be made, allowing the prediction to be time-sensitive. Figure 4.4(b) illustrates such scenario.
- *Link prediction in dynamic knowledge graphs.* Given a test triplet (s, r, t), we aim at predicting the object that involves in this relationship r with subject s at time t. Again we use (4.11) to rank all the candidate objects. We also conduct testing after applying the filtering techniques described in [86] - we only rank against the entities that do not generate a true triplet (seen in train) when it replaces ground truth object.

Time prediction. Given a triplet (s, o, r), we aim at answering the following question: *When this subject will interact with this object with event type r in the*

future? We predict this quantity by computing the expected next event time under $p^{s,o,r}(t)$. Since the intensity model is a Rayleigh model, the expected event time can be computed in closed form as

$$\mathbb{E}_{t\sim p^{s,o,r}(t)}[t] = \sqrt{\frac{\pi}{2\exp\left(\boldsymbol{f}_s(t-)^\top \boldsymbol{R}^r \boldsymbol{f}_o(t-)\right)}} \tag{4.12}$$

4.5 Complexity Analysis

In this section, we provide an in depth analysis of our approach in terms of the model size, the training and testing complexity. We measure these terms as functions of the number of entities and the number of events. Other factors, such as dimensionality of latent representations, are treated as constant.

Model size. If the baseline profile feature for entities are not available, we can use one-hot representation of those entities, and the basic feature embedding takes $O(M)$ parameters. The interaction features (e.g., bag of words features for reviews) are independent of the number of users and number of items. Moreover, the parameters of RNN are also independent of the dataset. Thus, our model is as compact as the traditional matrix factorization methods.

Training complexity. Using BPTT with a budget limit, each mini-batch training will only involve consecutive K samples. When an event happens, the embeddings of the corresponding dimension are updated. Thus we need $O(K)$ operations for updating embeddings. For each dimension triplet (s, o, r) that has event on it, we use NCE to sample C dimensions that survive from last event to current event as mentioned in Section 4.3. This allows us to further reduce the computational cost to $O(C \times K)$. Thus in summary, each stochastic training step takes $O(M)$ cost, which is linear to the number of samples in mini-batch.

Prediction complexity. The entity prediction in (4.11) requires comparing each entity with the current partial information. Since (4.11) has a closed form,

the complexity is $O(M)$ where M is the number of entities. This can further be improved by other methods such as fast inner product search [87]. Since the event time prediction in (4.12) is in closed form, the complexity for this is $O(1)$.

4.6 Experiments

Our DeepCoevolve model can be applied to any application where there is a continuous-time evolving graph, in this section we focus on the temporal knowledge graph reasoning problem.

We use two datasets: Global Database of Events, Language, and Tone (GDELT) [88] and Integrated Crisis Early Warning System (ICEWS) [89]. GDELT data is collected from April 1, 2015 to Mar 31, 2016 (temporal granularity of 15 mins). ICEWS dataset is collected from Jan 1, 2014 to Dec 31, 2014 (temporal granularity of 24 hrs). Both datasets contain records of events that include two actors, action type and timestamp of event. We use different hierarchy of actions in two datasets - (top level 20 relations for GDELT while last level 260 relations for ICEWS) - to test on variety of knowledge tensor configurations. Note that this does not filter any record from the dataset. We process both datasets to remove any duplicate quadruples, any mono-actor events (*i.e.*, we use only dyadic events), and self-loops. We report our main results on full versions of each dataset. We create smaller version of both datasets for exploration purposes. Table 4.1 provides statistics about the data and Table 4.2 demonstrates the sparsity of knowledge tensor.

We use the same evaluation metrics as these in the experiments on recommendation systems in the previous chapter.

- Link prediction metric. We report Mean Absolute Rank (MAR), Standard Deviation for MAR and HITS@10 (correct entity in top 10 predictions) for both Raw and Filtered Versions.
- Time prediction metric. We report the Mean Absolute Error (MAE) between the

predicted and true time in hours.

4.6.1 Competitors

We compare with following relational learning methods: RESCAL [90], Neural Tensor Network (NTN) [91], Multiway Neural Network (ER-MLP) [92], TransE [86] and TransR [93]. To the best of our knowledge, there are no existing relational learning approaches that can predict time for a new fact. Hence we devised two baseline methods for evaluating time prediction performance:

- Multi-dimensional Hawkes process (MHP): We model dyadic entity interactions as multi-dimensional Hawkes process similar to [5]. Here, an entity pair constitutes a dimension and for each pair we collect sequence of events on its dimension and train and test on that sequence. Relationship is not modeled in this setup.
- Recurrent Temporal Point Process (RTPP): We implement a simplified version of RMTPP [81] where we do not predict the marker. For training, we concatenate static entity and relationship embeddings and augment the resulting vector with temporal feature. This augmented unit is used as input to global RNN which produces output vector $\mathbf{h_t}$. During test time, for a given triplet, we use this vector $\mathbf{h_t}$ to compute conditional intensity of the event given history which is further used to predict next event time.

4.6.2 Experimental Results on Real-world Data

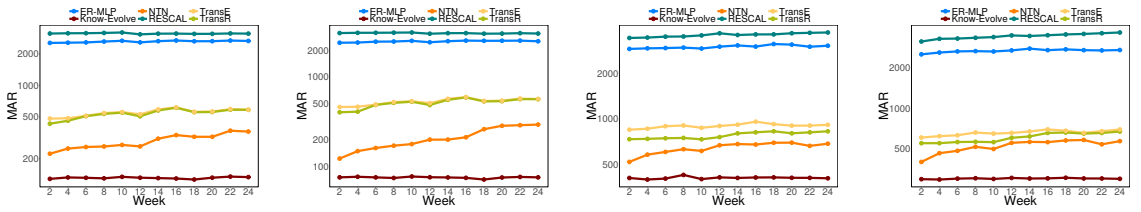
Link prediction. Figure (4.5, 4.6, 4.7) demonstrate link prediction performance comparison on both datasets. Our method (Know-Evolve) significantly and consistently outperforms all competitors in terms of prediction rank without any deterioration over time. Neural Tensor Network's second best performance compared to other baselines demonstrate its rich expressive power but it fails to capture the evolving dynamics of intricate dependencies over time. This is further substantiated by its

Table 4.1: Statistics of Each Dataset.

Dataset Name	# Entities	# Relations	# Events
GDELT-full	14018	20	31.29M
GDELT-500	500	20	3.42M
ICEWS-full	12498	260	0.67M
ICEWS-500	500	256	0.45M

Table 4.2: Sparsity of Knowledge Tensor.

Dataset Name	# Possible Entries	# Available Entries	% Proportion
GDELT-full	3.93B	4.52M	0.12
GDELT-500	5M	0.76M	15.21
ICEWS-full	39.98B	0.31M	7e-3
ICEWS-500	64M	0.15M	0.24



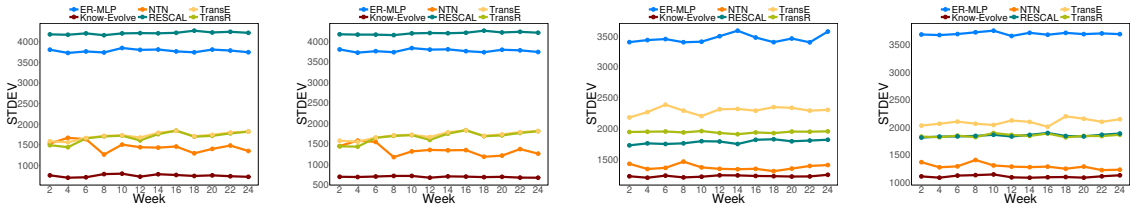
(a) ICEWS-raw (b) ICEWS-filtered (c) GDELT-raw (d) GDELT-filtered

Figure 4.5: Mean Average Rank (MAR) for Entity Prediction on both datasets.

decreasing performance as we move test window further in time.

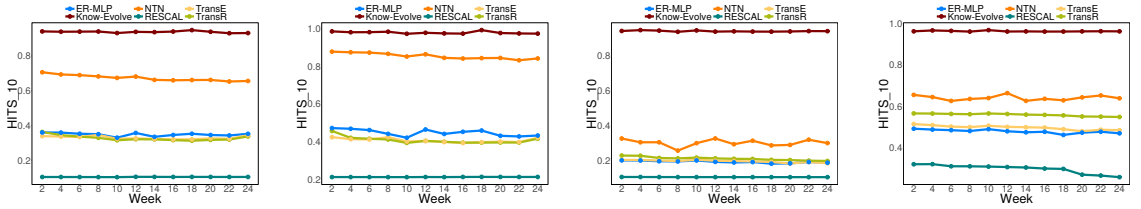
The second row represents deviation for MAR across samples in a given test window. Our method achieves significantly low deviation error. Finally, high performance on HITS@10 metric demonstrates extensive discriminative ability of Know-Evolve. For instance, GDELT has only 20 relations but 32M events where many entities interact with each other in multiple relationships. In this complex setting, other methods depend only on static entity embeddings to perform prediction unlike our method which does effectively infers new knowledge using powerful evolutionary network and provides accurate prediction results.

Time prediction. Figure 4.8 demonstrates that Know-Evolve performs significantly better. MHP uses a specific parametric form of the intensity function which



(a) ICEWS-raw (b) ICEWS-filtered (c) GDELT-raw (d) GDELT-filtered

Figure 4.6: Standard Deviation (STD) in MAR for entity prediction on both datasets.

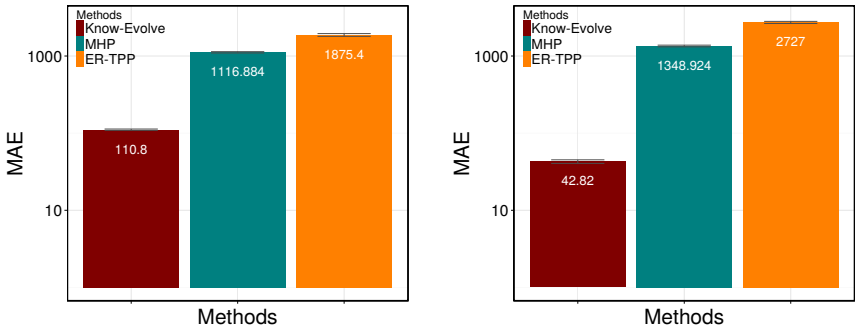


(a) ICEWS-raw (b) ICEWS-filtered (c) GDELT-raw (d) GDELT-filtered

Figure 4.7: HITS@10 for entity prediction on both datasets.

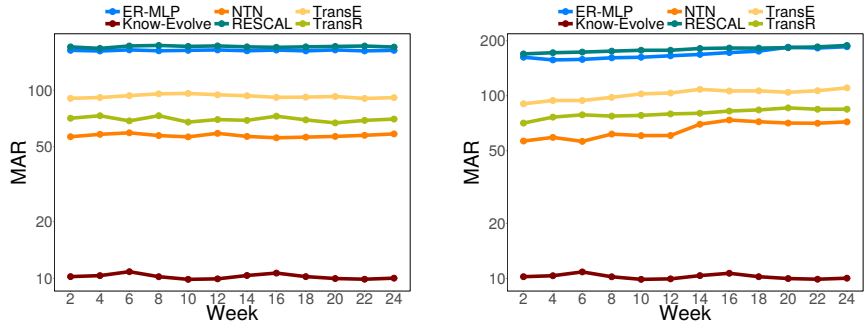
limits its expressiveness. Further, each entity pair interaction is modeled as an independent dimension and does not take into account relational feature which fails to capture the intricate influence of different entities on each other. On the other hand, RTPP uses relational features as part of input, but it sees all events globally and cannot model the intricate evolutionary dependencies on past events. We observe that our method effectively captures such non-linear relational and temporal dynamics.

Sliding window training. We further partition our test set in 12 different slides and report results in each window. For both datasets, each slide included 2 weeks of time. Unlike competitors, the entity embeddings in our model get updated after every event in the test, but the model parameters remain unchanged after training. To balance out the advantage that this may give to our method, we explore the use of sliding window training paradigm for baselines: We train on first six months of dataset and evaluate on the first test window. Next we throw away as many days (2 weeks) from start of train set as found in test set and incorporate the test data into training. We retrain the model using previously learned parameters as warm start.



(a) GDELT-500 (b) ICEWS-500

Figure 4.8: Time prediction performance (Unit is hours).

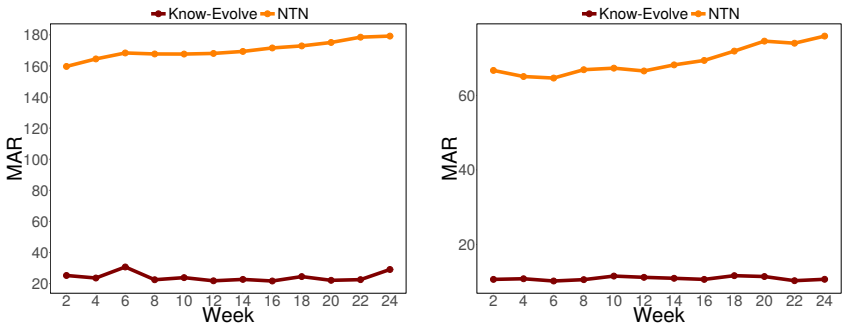


(a) Sliding window training (b) Non-sliding window training

Figure 4.9: Performance comparison of sliding window vs. non-sliding window training on GDELT-500 data.

This can effectively aid the baselines to adapt to the evolving knowledge over time. Figure 4.9 shows that the sliding window training contributes to stable performance of baselines across the time window (i.e.the temporal deterioration is no longer observed significantly for baselines). But the overall performance of our method still surpasses all the competitors.

Recurrent facts vs. new facts. One fundamental distinction in our multi-relational setting is the existence of recurrence relations which is not the case for traditional knowledge graphs. To that end, we compare our method with the best performing competitor - NTN on two different testing setups: 1) only Recurrent Facts in test set, and 2) only New facts in test set. We call a test fact "new" if it was never seen in training. As one can expect, the proportion of new facts will increase as we



(a) New facts only (b) Recurrent facts only

Figure 4.10: Comparison with NTN over recurrent and non-recurrent test version on GDELT-500.

move further in time. In our case, it ranges from 40%-60% of the total number of events in a specific test window. Figure 4.10 demonstrates that our method performs consistently and significantly better in both cases.

4.7 Summary

In this chapter, we have proposed an expressive and efficient framework to model the nonlinear coevolution nature of users' and items' embeddings. Moreover, the user and item's evolving and coevolving processes are captured by a novel recurrent evolving network. We further developed an efficient stochastic training algorithm for the coevolving user-item networks. We demonstrate the superior performance of our method on both the time and item prediction task in the application of temporal reasoning in knowledge graphs. They achieve the state of the art predictive performance compared with the epoch based methods. There are many interesting lines for future work. For example, we can extend it to other applications in social sciences, such as modeling the group dynamics in message services. We can also apply this framework to design more explainable models by exploiting the idea of modeling a fine-grained evolving network. Our stochastic training algorithm can also be applied in many applications to improve the efficiency in the dynamic computation for graphs and hyper-graphs.

We have designed expressive models to understand users' temporal behaviors in

various applications, such as recommendation systems and knowledge graph reasoning. Besides the modeling perspective, another key question is how to fully exploit these learned point process models and predict users' temporal behaviors in the future. In the next chapter, we will discuss our principled solution to this challenging problem.

CHAPTER 5
SCALABLE USER ACTIVITY LEVEL PREDICTION IN POINT PROCESS MODELS

5.1 Introduction

Online social platforms, such as Facebook and Twitter, enable users to post opinions, share information, and influence peers. Recently, user-generated event data archived in fine-grained temporal resolutions are becoming increasingly available, which calls for expressive models and algorithms to understand, predict and distill knowledge from complex dynamics of these data. Particularly, temporal point processes are well-suited to model the event pattern of user behaviors and have been successfully applied in modeling event sequence data [1, 2, 3, 4, 5, 8, 9, 94, 95, 96, 97].

A fundamental task in social networks is to predict user activity levels based on learned point process models. Mathematically, the goal is to compute $\mathbb{E}[f(N(t))]$, where $N(t)$ is a given point process that is learned from user behaviors, t is a fixed future time, and f is an application-dependent function. A framework for doing this is critically important. For example, for social networking services, an accurate inference of the number of reshares of a post enables the network moderator to detect trending posts and improve its content delivery networks [98, 99]; an accurate estimate of the change of network topology (the number of new followers of a user) facilitates the moderator to identify influential users and suppress the spread of terrorist propaganda and cyber-attacks [4]; an accurate inference of the activity level (number of posts in the network) allows us to gain fundamental insight into the predictability of collective behaviors [100]. Moreover, for online merchants such as Amazon, an accurate estimate of the number of future purchases of a product helps optimizing future advertisement

placements [1, 5].

Despite the prevalence of prediction problems, an accurate prediction is very challenging for two reasons. First, the function f is arbitrary. For instance, to evaluate the homogeneity of user activities, we set $f(x) = x \log(x)$ to compute the Shannon entropy; to measure the distance between a predicted activity level and a target x^*, we set $f(x) = (x - x^*)^2$. However, most works [28, 101, 98, 102, 103, 99] are problem specific and only designed for the simple task with $f(x) = x$; hence these works are not generalizable. Second, point process models typically have intertwined stochasticity and can co-evolve over time [1, 4], *e.g.*, in the influence propagation problem, the information diffusion over networks can change the structure of networks, which adversely influences the diffusion process [4]. However, previous works often ignore parts of the stochasticity in the intensity function [6] or make heuristic approximations [98, 99]. Hence, there is an urgent need for a method that is applicable to an arbitrary function f and keeps all the stochasticity in the process, which is largely nonexistent to date.

We propose HYBRID, a generic framework that provides an efficient estimator of the probability mass of point processes. Figure 5.1 illustrates our framework. We also make the following contributions:

- **Unifying framework**. Our framework is applicable to general point processes and does not depend on specific parameterization of intensity functions. It incorporates all stochasticity in point processes and is applicable to prediction tasks with an arbitrary function f.
- **Technical challenges**. We reformulate the prediction problem and design a random variable with reduced variance. To derive an analytical form of this random variable, we also propose a mass transport equation to compute the conditional probability mass of point processes. We further transform this equation to an Ordinary Differential Equation and provide a scalable algorithm.

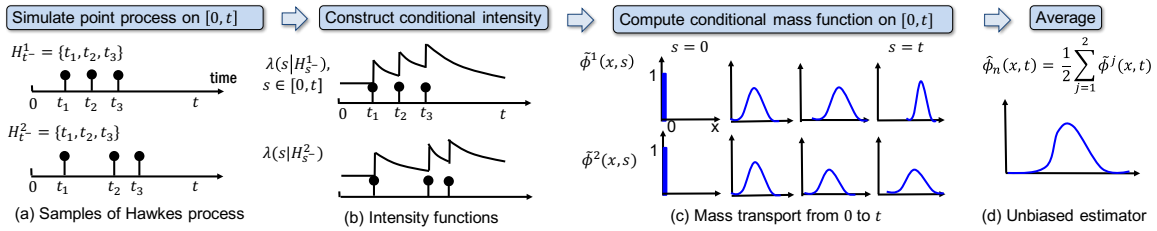


Figure 5.1: An illustration of HYBRID using Hawkes process. Our method first generates two samples $\{\mathcal{H}_{t^-}^i\}$ of events; then it constructs intensity functions; with these inputs, it computes conditional probability mass functions $\tilde{\phi}^i(x,s) := \mathbb{P}[N(s) = x|\mathcal{H}_{s^-}^i]$ using a mass transport equation. Panel (c) shows the transport of conditional mass at four different times (the initial probability mass $\tilde{\phi}(x,0)$ is an indicator function $\mathbb{I}[x=0]$, as there is no event with probability one). Finally, the average of conditional mass functions yields our estimator of the probability mass.

- **Superior performance**. Our framework significantly reduces the sample size to estimate the probability mass function of point processes in real-world applications. For example, to infer the number of tweeting and retweeting events of users in the co-evolution model of information diffusion and social link creation [4], our method needs 10^3 samples and 14.4 minutes, while Monte Carlo needs 10^6 samples and 27.8 hours to achieve the same relative error of 0.1.

Background on Monte Carlo (MC). To compute the probability mass of a point process, MC simulates n realizations of history $\{\mathcal{H}_t^i\}$ using the thinning algorithm [59]. The number of events in sample i is defined as $N^i(t) = |\mathcal{H}_t^i|$. Let $\phi(x,t) := \mathbb{P}[N(t) = x]$, where $x \in \mathbb{N}$, be the probability mass. Then its estimator $\hat{\phi}_n^{mc}(x,t)$ and the estimator $\hat{\mu}_n^{mc}(t)$ for $\mu(t) := \mathbb{E}[f(N(t))]$ are defined as $\hat{\phi}_n^{mc}(x,t) = \frac{1}{n}\sum_i \mathbb{I}[N^i(t) = x]$ and $\hat{\mu}_n^{mc}(t) = \frac{1}{n}\sum_i f(N^i(t))$. The root mean square error (RMSE) is defined as

$$\varepsilon(\hat{\mu}_n^{mc}(t)) = \sqrt{\mathbb{E}[\hat{\mu}_n^{mc}(t) - \mu(t)]^2} = \sqrt{\mathbb{VAR}[f(N(t))]/n}. \tag{5.1}$$

5.2 Solution Overview

Given an arbitrary point process $N(t)$ that is learned from data, existing prediction methods for computing $\mathbb{E}[f(N(t))]$ have three major limitations:

- **Generalizability**. Most methods [28, 101, 98, 102, 103, 99] only predict $\mathbb{E}[N(t)]$ and are not generalizable to an arbitrary function f. Moreover, they typically rely on specific parameterizations of the intensity functions, such as the reinforced Poisson process [98] and Hawkes process [104, 99]; hence they are not applicable to general point processes.
- **Approximation and heuristics**. These works also ignore parts of the stochasticity in the intensity functions [6] or make heuristic approximations to the point process [98, 99]. Hence the accuracy is limited by the approximations and heuristic corrections.
- **Large sample size**. The MC method overcomes the above limitations since it has an unbiased estimator of the probability mass. However, the high stochasticity in point processes leads to a large value of $\mathbb{VAR}[f(N(t))]$, which requires a large number of samples to achieve a small error.

To address these challenges, we propose a generic framework with a novel estimator of the probability mass, which has a smaller sample size than MC. Our framework has the following key steps.

I. New random variable. We design a random variable $g(\mathcal{H}_{t^-})$, a conditional expectation given the history. Its variance is guaranteed to be smaller than that of $f(N(t))$. For a fixed number of samples, the error of MC is decided by the variance of the random variable of interest, as shown in (5.1). Hence, to achieve the same error, applying MC to estimate the new objective $\mathbb{E}_{\mathcal{H}_{t^-}}[g(\mathcal{H}_{t^-})]$ requires smaller number of samples compared with the procedure that directly estimates $\mathbb{E}[f(N(t))]$.

II. Mass transport equation. To compute $g(\mathcal{H}_{t^-})$, we derive a differential-

difference equation that describes the evolutionary dynamics of the conditional probability mass $\mathbb{P}[N(t) = x|\mathcal{H}_{t^-}]$. We further formulate this equation as an Ordinary Differential Equation, and provide a scalable algorithm.

5.3 Hybrid Inference Machine with Probability Mass Transport

In this section, we present technical details of our framework. We first design a new random variable for prediction; then we propose a mass transport equation to compute this random variable analytically. Finally, we combine the mass transport equation with the sampling scheme to compute the probability mass function of general point processes and solve prediction tasks with an arbitrary function f.

5.3.1 New Random Variable with Reduced Variance

We reformulate the problem and design a new random variable $g(\mathcal{H}_{t^-})$, which has a smaller variance than $f(N(t))$ and the same expectation. To do this, we express $\mathbb{E}[f(N(t))]$ as an iterated expectation

$$\mathbb{E}[f(N(t))] = \mathbb{E}_{\mathcal{H}_{t^-}}\Big[\mathbb{E}_{N(t)|\mathcal{H}_{t^-}}\big[f(N(t))|\mathcal{H}_{t^-}\big]\Big] = \mathbb{E}_{\mathcal{H}_{t^-}}\Big[g(\mathcal{H}_{t^-})\Big], \tag{5.2}$$

where $\mathbb{E}_{\mathcal{H}_{t^-}}$ is w.r.t. the randomness of the history and $\mathbb{E}_{N(t)|\mathcal{H}_{t^-}}$ is w.r.t. the randomness of the point process given the history. We design the random variable as a conditional expectation given the history: $g(\mathcal{H}_{t^-}) = \mathbb{E}_{N(t)|\mathcal{H}_{t^-}}[f(N(t))|\mathcal{H}_{t^-}]$. Theorem 6 shows that it has a smaller variance.

Theorem 7. *For time $t > 0$ and an arbitrary function f, we have $\mathbb{VAR}[g(\mathcal{H}_{t^-})] < \mathbb{VAR}[f(N(t))]$.*

Theorem 7 extends the Rao-Blackwell (RB) theorem [105] to point processes. RB says that if $\hat{\theta}$ is an estimator of a parameter θ and T is a sufficient statistic for θ; then $\mathbb{VAR}[\mathbb{E}[\hat{\theta}|T]] \leqslant \mathbb{VAR}[\hat{\theta}]$, *i.e.*, the sufficient statistic reduces uncertainty of

$\hat{\theta}$. However, the RB theorem is not applicable to point processes since it studies a different problem (improving the estimator of a distribution's parameter), while we focus on the prediction problem for general point processes, which introduces two new technical challenges:

(i) Is there a definition in point processes whose role is similar to the sufficient statistic in RB? Our first contribution shows that the history $\mathcal{H}_{t^-}$ contains all the necessary information in a point process and reduces the uncertainty of $N(t)$. Hence, $g(\mathcal{H}_{t^-})$ is an improved variable for prediction. Moreover, in contrast to the RB theorem, the inequality in Theorem 6 is *strict* because the counting process $N(t)$ is right-continuous in time t and not predictable [106] (a predictable process is measurable w.r.t. $\mathcal{H}_{t^-}$, such as the processes that are left-continuous). Appendix B.2 contains details on the proof.

(ii) Is $g(\mathcal{H}_{t^-})$ computable for *general* point processes and an *arbitrary* function f? An efficient computation will enable us to estimate $\mathbb{E}_{\mathcal{H}_{t^-}}[g(\mathcal{H}_{t^-})]$ using the sampling method. Specifically, let $\hat{\mu}_n(t) = \frac{1}{n}\sum_i g(\mathcal{H}^i_{t^-})$ be the estimator computed from n samples; then from the definition of RMSE in (5.1), this estimator has smaller error than MC: $\varepsilon(\hat{\mu}_n(t)) < \varepsilon(\hat{\mu}^{mc}_n(t))$.

However, the challenge in our new formulation is that it seems very hard to compute this conditional expectation, as one typically needs another round of sampling, which is undesirable as it will increase the variance of the estimator. To address this challenge, next we propose a mass transport equation.

5.3.2 Transport Equation for Conditional Probability Mass Function

We present a novel mass transport equation that computes the conditional probability mass $\tilde{\phi}(x,t) := \mathbb{P}[N(t) = x|\mathcal{H}_{t^-}]$ of general point processes. With this definition, we derive an analytical expression for the conditional expectation: $g(\mathcal{H}_{t^-}) = \sum_x f(x)\tilde{\phi}(x,t)$. The transport equation is as follows.

Theorem 8 (Mass Transport Equation for Point Processes)**.** *Let* $\lambda(t) := \lambda(t|\mathcal{H}_{t^-})$ *be the conditional intensity function of the point process* $N(t)$ *and* $\tilde{\phi}(x,t) := \mathbb{P}[N(t) = x|\mathcal{H}_{t^-}]$ *be its conditional probability mass function; then* $\tilde{\phi}(x,t)$ *satisfies the following differential-difference equation:*

$$\underset{\substack{\uparrow \\ \textit{rate of change in mass}}}{\tilde{\phi}_t(x,t) := \frac{\partial \tilde{\phi}(x,t)}{\partial t}} = \begin{cases} -\lambda(t)\tilde{\phi}(x,t) & \text{if } x = 0 \\ \underbrace{-\lambda(t)\tilde{\phi}(x,t)}_{\textit{loss in mass, at rate } \lambda(t)} + \underbrace{\lambda(t)\tilde{\phi}(x-1,t)}_{\textit{gain in mass, at rate } \lambda(t)} & \text{if } x = 1, 2, 3, \cdots \end{cases} \tag{5.3}$$

Proof sketch. For the simplicity of notation, we set the right-hand-side of (5.3) to be $\mathcal{F}[\tilde{\phi}]$, where $\mathcal{F}$ is a functional operator on $\tilde{\phi}$. We also define the inner product between functions $u : \mathbb{N} \to \mathbb{R}$ and $v : \mathbb{N} \to \mathbb{R}$ as $(u, v) := \sum_x u(x)v(x)$. The main idea in our proof is to show that the equality $(v, \tilde{\phi}_t) = (v, \mathcal{F}[\tilde{\phi}])$ holds for any test function v; then $\tilde{\phi}_t = \mathcal{F}[\tilde{\phi}]$ follows from the fundamental lemma of the calculus of variations [107]. Specifically, the proof contains two parts as follows.

We first prove $(v, \tilde{\phi}_t) = (\mathcal{B}[v], \tilde{\phi})$, where $\mathcal{B}[v]$ is a functional operator defined as $\mathcal{B}[v] = (v(x+1) - v(x))\lambda(t)$. This equality can be proved by the property of point processes and the definition of conditional mass. Second, we show $(\mathcal{B}[v], \tilde{\phi}) = (v, \mathcal{F}[\tilde{\phi}])$ using a variable substitution technique. Mathematically, this equality means $\mathcal{B}$ and $\mathcal{F}$ are *adjoint* operators on the function space. Combining these two equalities yields the mass transport equation. Appendix B.1 contains details on the proof.

Mass transport dynamics. This differential-difference equation describes the time evolution of the conditional mass. Specifically, the differential term $\tilde{\phi}_t$, *i.e.*, the instantaneous rate of change in the probability mass, is equal to a first order difference equation on the right-hand-side. This difference equation is a summation of two terms: (i) the negative loss of its own probability mass $\tilde{\phi}(x,t)$ at rate $\lambda(t)$, and (ii) the positive gain of probability mass $\tilde{\phi}(x-1,t)$ from last state $x-1$ at

Algorithm 5 CONDITIONAL MASS FUNCTION

Input: $\mathcal{H}_{t^-} = \{t_k\}_{k=1}^K$, $\Delta\tau$, set $t = t_{K+1}$
Output: Conditional probability mass function $\tilde{\boldsymbol{\phi}}(t)$
for $k = 0, \cdots, K$ **do**
    Construct $\lambda(s)$ and $\boldsymbol{Q}(s)$ on $[t_k, t_{k+1}]$
    $\tilde{\boldsymbol{\phi}}(t_{k+1}) = \text{ODE45}[\tilde{\boldsymbol{\phi}}(t_k), \boldsymbol{Q}(s), \Delta\tau)]$ (RK Alg)
end for
$\tilde{\boldsymbol{\phi}}(t) = \tilde{\boldsymbol{\phi}}(t_{K+1})$

Algorithm 6 HYBRID MASS TRANSPORT

Input: Sample size n, time t, $\Delta\tau$
Output: $\hat{\mu}_n(t), \hat{\phi}_n(x,t)$
for $i = 1, \cdots, n$ **do**
    $\tilde{\phi}^i(x,t) = \text{COND-MASS-FUNC}(\mathcal{H}^i_{t^-}, \Delta\tau)$
end for
$\hat{\phi}_n(x,t) = \frac{1}{n}\sum_i \tilde{\phi}^i(x,t)$, $\hat{\mu}_n(t) = \sum_x f(x)\hat{\phi}_n(x,t)$

rate $\lambda(t)$. Moreover, since initially no event happens with probability one, we have $\tilde{\phi}(x,0) = \mathbb{I}[x=0]$. Solving this transport equation on $[0,t]$ essentially transports the initial mass to the mass at time t.

5.3.3 Mass Transport as s Banded Linear Ordinary Differential Equation

To efficiently solve the mass transport equation, we reformulate it as a banded linear ODE. Specifically, we set the upper bound for x to be M, and set $\tilde{\boldsymbol{\phi}}(t)$ to be a vector that includes the value of $\tilde{\phi}(x,t)$ for each integer x: $\tilde{\boldsymbol{\phi}}(t) = (\tilde{\phi}(0,t), \tilde{\phi}(1,t), \cdots, \tilde{\phi}(M,t))^\top$. With this representation of the conditional mass, the mass transport equation in (5.3) can be expressed as a simple banded linear ODE:

$$\tilde{\boldsymbol{\phi}}(t)' = \boldsymbol{Q}(t)\tilde{\boldsymbol{\phi}}(t), \tag{5.4}$$

where $\tilde{\boldsymbol{\phi}}(t)' = (\tilde{\phi}_t(0,t), \cdots, \tilde{\phi}_t(M,t))^\top$, and the matrix $\boldsymbol{Q}(t)$ is a sparse bi-diagonal matrix with $Q_{i,i} = -\lambda(t)$ and $Q_{i-1,i} = \lambda(t)$. The following equation visualizes the

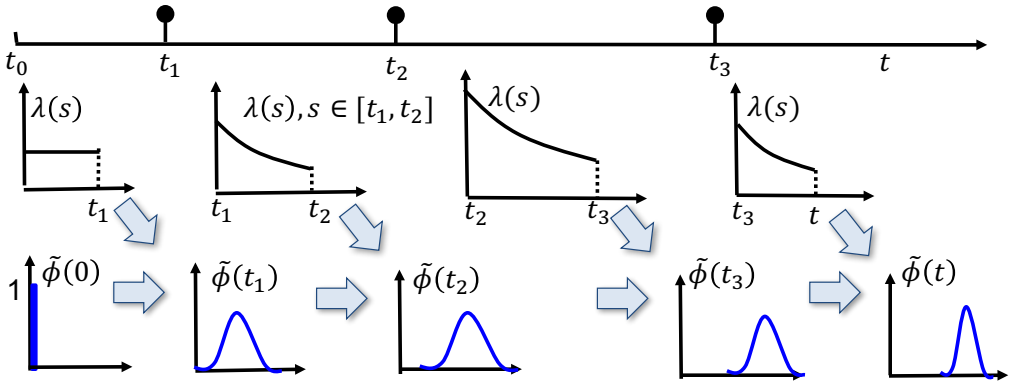


Figure 5.2: Illustration of Algorithm 5 using Hawkes process. The intensity is updated after each event t_k. Within $[t_k, t_{k+1}]$, we use $\boldsymbol{\phi}(t_k)$ and the intensity $\lambda(s)$ to solve the ODE and obtain $\boldsymbol{\phi}(t_{k+1})$.

ODE in (5.4) when $M = 2$.

$$\begin{pmatrix} \tilde{\phi}_t(0,t) \\ \tilde{\phi}_t(1,t) \\ \tilde{\phi}_t(2,t) \end{pmatrix} = \begin{pmatrix} -\lambda(t) & & \\ \lambda(t) & -\lambda(t) & \\ & \lambda(t) & -\lambda(t) \end{pmatrix} \begin{pmatrix} \tilde{\phi}(0,t) \\ \tilde{\phi}(1,t) \\ \tilde{\phi}(2,t) \end{pmatrix}. \tag{5.5}$$

This dynamic ODE is a compact representation of the transport equation in (5.3) and M decides the dimension of the ODE in (5.4). In theory, M can be unbounded. However, the conditional probability mass is tends to zero when M becomes large. Hence, in practice we choose a finite support $\{0, 1, \cdots, M\}$ for the conditional probability mass function. To choose a proper M, we generate samples from the point process. Suppose the largest number of events in the samples is L, we set $M = 2L$ such that it is reasonably large. Next, with the initial probability mass $\tilde{\boldsymbol{\phi}}(t_0) = (1, 0, \cdots, 0)^\top$, we present an efficient algorithm to solve the ODE.

5.3.4 Scalable Algorithm for Solving the ODE

We present the algorithm that transports the initial mass $\tilde{\boldsymbol{\phi}}(t_0)$ to $\tilde{\boldsymbol{\phi}}(t)$ by solving the ODE.

Since the intensity function is history-dependent and has a discrete jump when an

event happens at time t_k, the matrix $\boldsymbol{Q}(t)$ in the ODE is discontinuous at t_k. Hence we split $[0, t]$ into intervals $[t_k, t_{k+1}]$. On each interval, the intensity is continuous and we can use the classic numerical Runge-Kutta (RK) method [108] to solve the ODE. Figure 5.2 illustrates the overall algorithm.

Our algorithm works as follows. First, with the initial intensity on $[0, t_1]$ and $\tilde{\boldsymbol{\phi}}(t_0)$ as input, the RK method solves the ODE on $[0, t_1]$ and outputs $\tilde{\boldsymbol{\phi}}(t_1)$. Since an event happens at t_1, the intensity is updated on $[t_1, t_2]$. Next, with the updated intensity and $\tilde{\boldsymbol{\phi}}(t_1)$ as the initial value, the RK method solves the ODE on $[t_1, t_2]$ and outputs $\tilde{\boldsymbol{\phi}}(t_2)$. This procedure repeats for each $[t_k, t_{k+1}]$ until time t.

Now we present the RK method that solves the ODE on each interval $[t_k, t_{k+1}]$. RK divides this interval into equally-spaced subintervals $[\tau_i, \tau_{i+1}]$, for $i = 0, \cdots, I$ and $\Delta\tau = \tau_{i+1} - \tau_i$. It then conducts linear extrapolation on each subinterval. It starts from $\tau_0 = t_k$ and uses $\tilde{\boldsymbol{\phi}}(\tau_0)$ and the approximation of the gradient $\tilde{\boldsymbol{\phi}}(\tau_0)'$ to compute $\tilde{\boldsymbol{\phi}}(\tau_1)$. Next, $\tilde{\boldsymbol{\phi}}(\tau_1)$ is taken as the initial value and the process is repeated until $\tau_I = t_{k+1}$.

The RK method approximates the gradient $\tilde{\boldsymbol{\phi}}(t)'$ with different levels of accuracy, called states s. When $s = 1$, it is the Euler method, which uses the first order approximation $\tilde{\boldsymbol{\phi}}(\tau_{i+1}) - \tilde{\boldsymbol{\phi}}(\tau_i)/\Delta\tau$. We use the ODE45 solver in MATLAB and choose the stage $s = 4$ for RK. Moreover, the main computation in the RK method comes from the matrix-vector product. Since the matrix $\boldsymbol{Q}(t)$ is sparse and bi-diagonal with $O(M)$ non-zero elements, the cost for this operation is only $O(M)$.

5.3.5 Hybrid Inference Machine with Mass Transport Equation

With the conditional probability mass, we are now ready to express $g(\mathcal{H}_{t^-})$ in closed form and estimate $\mathbb{E}_{\mathcal{H}_{t^-}}[g(\mathcal{H}_{t^-})]$ using the MC sampling method. We present our framework HYBRID:

(i) Generate n samples $\{\mathcal{H}^i_{t^-}\}$ from a point process $N(t)$ with a stochastic intensity

$\lambda(t)$.

(ii) For each sample $\mathcal{H}_{t^-}^i$, we compute the value of intensity function $\lambda(s|\mathcal{H}_{s^-}^i)$, for each $s \in [0, t]$; then we solve (5.4) to compute the conditional probability mass $\tilde{\phi}^i(x, t)$.

(iii) We obtain the estimator of the probability mass function $\phi(x, t)$ and $\mu(t)$ by taking the average: $\hat{\phi}_n(x, t) = \frac{1}{n} \sum_{i=1}^n \tilde{\phi}^i(x, t), \quad \hat{\mu}_n(t) = \sum_x f(x)\hat{\phi}_n(x, t)$

Algorithm 8 summarizes the above procedure. Next, we discuss two properties of HYBRID.

First, our framework efficiently uses all event information in each sample. In fact, each event t_k influences the transport rate of the conditional probability mass (Figure 5.2). This feature is in sharp contrast to MC that only uses the information of the total number of events and neglects the differences in event times. For instance, the two samples in Figure 5.1(a) both have three events and MC treats them equally; hence its estimator is an indicator function $\hat{\phi}_n^{mc}(x, t) = \mathbb{I}[x = 3]$. However, for HYBRID, these samples have different event information and conditional probability mass functions, and our estimator in Figure 5.1(d) is much more informative than an indicator function.

Moreover, our estimator for the probability mass is unbiased if we can solve the mass transport equation in (5.3) exactly. To prove this property, we show that the following equality holds for an arbitrary function f: $(f, \phi) = \mathbb{E}[f(N(t))] = \mathbb{E}_{\mathcal{H}_{t^-}}[g(\mathcal{H}_{t^-})] = (f, \mathbb{E}_{\mathcal{H}_{t^-}}[\tilde{\phi}])$. Then $\mathbb{E}_{\mathcal{H}_{t^-}}[\hat{\phi}_n] = \phi$ follows from the fundamental lemma of the calculus of variations [107]. In practice, we choose a reasonable finite support for the conditional probability mass in order to solve the mass transport ODE in (5.4). Hence our estimator is nearly unbiased.

5.4 Applications and Extensions to Multi-dimensional Point Processes

In this section, we present two real world applications, where the point process models have intertwined stochasticity and co-evolving intensity functions.

Predicting the activeness and popularity of users in social networks. The co-evolution model [4] uses a Hawkes process $N_{us}(t)$ to model information diffusion (tweets/retweets), and a survival process $A_{us}(t)$ to model the dynamics of network topology (link creation process). The intensity of $N_{us}(t)$ depends on the network topology $A_{us}(t)$, and the intensity of $A_{us}(t)$ also depends on $N_{us}(t)$; hence these processes co-evolve over time. We focus on two tasks in this model: (i) inferring the activeness of a user by $\mathbb{E}[\sum_u N_{us}(t)]$, which is the number of tweets and retweets from user s; and (ii) inferring the popularity of a user by $\mathbb{E}[\sum_u A_{us}(t)]$, which is the number of new links created to the user.

Predicting the popularity of items in recommender systems. Recent works on recommendation systems [5, 1] use a point process $N_{ui}(t)$ to model user u's sequential interaction with item i. The intensity function $\lambda_{ui}(t)$ denotes user's interest to the item. As users interact with items over time, the user latent feature $\boldsymbol{u}_u(t)$ and item latent feature $\boldsymbol{i}_u(t)$ co-evolve over time, and are mutually dependent [1]. The intensity is parameterized as $\lambda_{ui}(t) = \eta_{ui} + \boldsymbol{u}_u(t)^\top \boldsymbol{i}_i(t)$, where η_{ui} is a baseline term representing the long-term preference, and the tendency for u to interact with i depends on the compatibility of their instantaneous latent features $\boldsymbol{u}_u(t)^\top \boldsymbol{i}_i(t)$. With this model, we can infer an item's popularity by evaluating $\mathbb{E}[\sum_u N_{ui}(t)]$, which is the number of events happened to item i.

To solve these prediction tasks, we extend the transport equation to the multivariate case. Specifically, we create a new stochastic process $x(t) = \sum_u N_{us}(t)$ and compute its conditional mass function.

Theorem 9 (Mass Transport for Multidimensional Point Processes)**.** *Let $N_{us}(t)$ be the point process with intensity $\lambda_{us}(t)$, $x(t) = \sum_{u=1}^{U} N_{us}(t)$, and $\tilde{\phi}(x,t) = \mathbb{P}[x(t) = x|\mathcal{H}_{t^-}]$ be the conditional probability mass of $x(t)$; then $\tilde{\phi}$ satisfies:*

$$\tilde{\phi}_t = -\big(\sum\nolimits_u \lambda_{us}(t)\big)\tilde{\phi}(x,t) + \big(\sum\nolimits_u \lambda_{us}(t)\big)\tilde{\phi}(x-1,t).$$

To compute the conditional probability mass, we also solve the ODE in (5.4), where the diagonal and off-diagonal of $\boldsymbol{Q}(t)$ is now the negative and positive summation of intensities in all dimensions.

5.5 Experiments

In this section, we evaluate the predictive performance of HYBRID in two real world applications in Section 5.4 and a synthetic dataset. We use the following metrics:

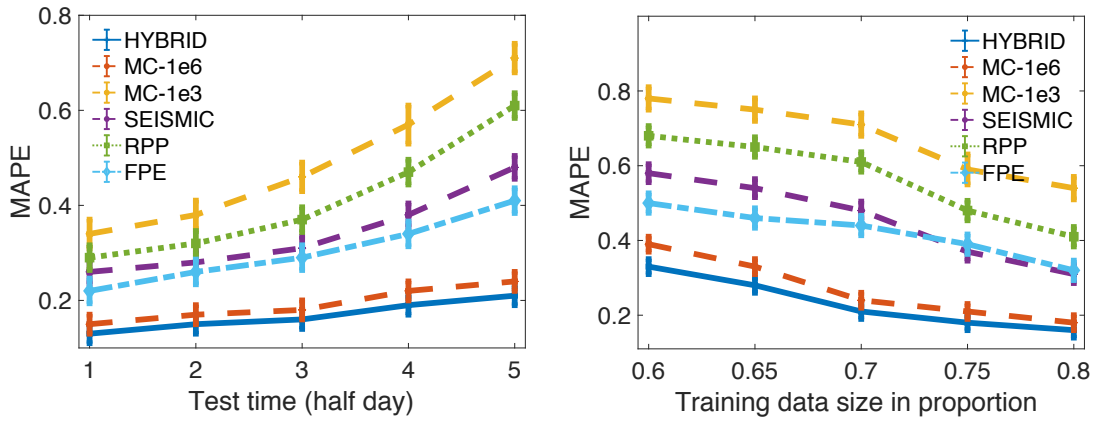
- Mean Average Percentage Error (MAPE). Given a prediction time t, we compute the MAPE $|\hat{\mu}_n(t) - \mu(t)|/\mu(t)$ between the estimated value and the ground truth.
- Rank correlation. For all users/items, we obtain two lists of ranks according to the true and estimated value of user activeness/user popularity/item popularity. The accuracy is evaluated by the Kendall-τ rank correlation [109] between two lists.

5.5.1 Experiments on Real-world Data

We show HYBRID has both accuracy and efficiency improvement in predicting the activeness and popularity of users in social networks and predicting the popularity of items in recommender systems.

Competitors. We use 10^3 samples for HYBRID and compare it with the following the state of the art.

- SEISMIC [99]. It defines a self-exciting process with a post infectiousness factor. It uses the branching property of Hawkes process and heuristic corrections for prediction.
- RPP [98]. It adds a reinforcement coefficient to Poisson process that depicts the self-excitation phenomena. It sets $\mathrm{d}N(t) = \lambda(t)\mathrm{d}t$ and solves a deterministic equation for prediction.
- FPE [6]. It uses a deterministic function to approximate the stochastic intensity function.



(a) MAPE vs. test time (b) MAPE vs. train size

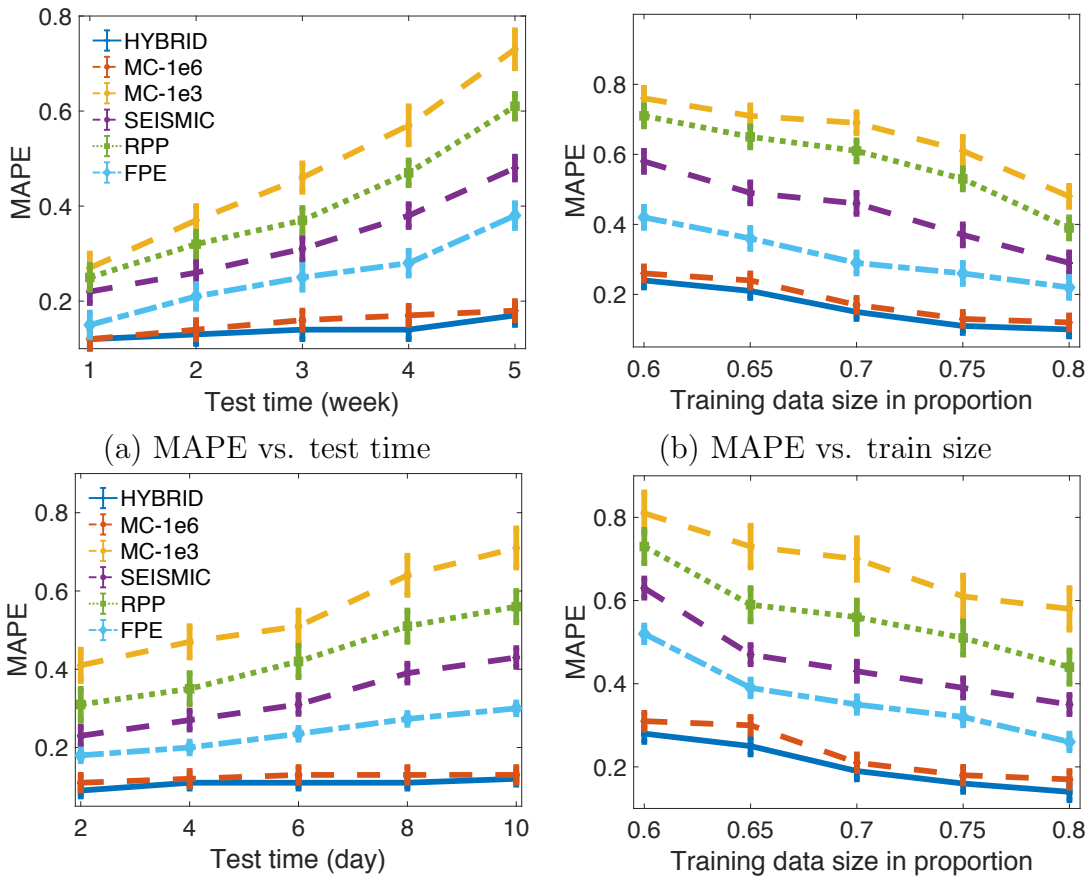
(c) MAPE vs. test time (d) MAPE vs. train size

Figure 5.3: Prediction results for user activeness and user popularity. (a,b) user activeness: predicting the number of posts per user; (c,d) user popularity: predicting the number of new links per user. Test times are the relative times after the end of train time. The train data is fixed with 70% of total data.

- MC-1E3. It is the MC sampling method with 10^3 samples (same as these for HYBRID), and MC-1E6 uses 10^6 samples.

Predicting the activeness and popularity of users in social networks

We use a Twitter dataset [110] that contains 280,000 users with 550,000 tweet, retweet, and link creation events during Sep. 21 - 30, 2012. This data is previously used to validate the network co-evolution model [4]. The parameters for tweeting/retweeting processes and link creation process are learned using maximum likelihood estimation [4]. SEISMIC and RPP are not designed for the popularity pre-



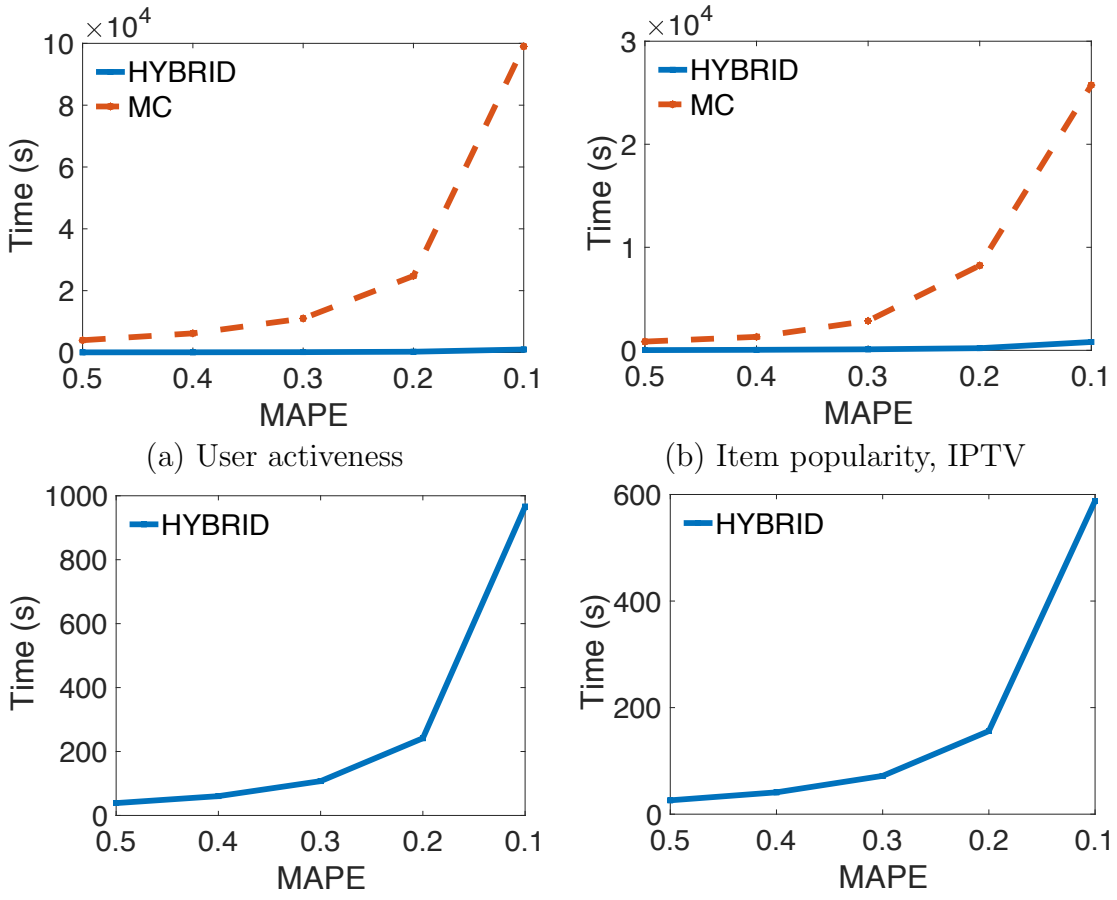
(a) MAPE vs. test time (b) MAPE vs. train size

(c) MAPE vs. test time (d) MAPE vs. train size

Figure 5.4: Prediction results for item popularity. (a,b) predicting the number of watching events per program on IPTV; (c,d) predicting the number of discussions per group on Reddit.

diction task since they do not consider the evolution of network topology. We use p proportion of total data as the training data to learn parameters of all methods, and the rest as test data. We make predictions for each user and report the averaged results.

Predictive performance. Figure 5.3(a) shows that MAPE increases as test time increases, since the model's stochasticity increases. HYBRID has the smallest error. Figure 5.3(b) shows that MAPE decreases as training data increases since model parameters are more accurate. Moreover, HYBRID is more accurate than SEISMIC and FPE with only 60% of training data, while these works need 80%. Thus, we make accurate predictions by observing users in the early stage. This feature is important



(a) User activeness (b) Item popularity, IPTV

(c) User activeness (d) Item popularity, IPTV

Figure 5.5: Scalability analysis: computation time as a function of error. (a,b) comparison between HYBRID and MC in different problems; (c,d) scalability plots for HYBRID.

for network moderators to identify malicious users and suppress the propagation undesired content.

Moreover, the consistent performance improvement shows two messages: (i) *considering all the randomness is important.* HYBRID is 2× more accurate than SEISMIC and FPE because HYBRID naturally considers all the stochasticity, but SEISMIC, FPE, and RPP need heuristics or approximations that discard parts of the stochasticity; (ii) *sampling efficiently is important.* To consider all the stochasticity, we need to use the sampling scheme, and HYBRID has a much smaller sample size. Specifically, HYBRID uses the same 10^3 samples, but has 4× error reduction compared with MC-

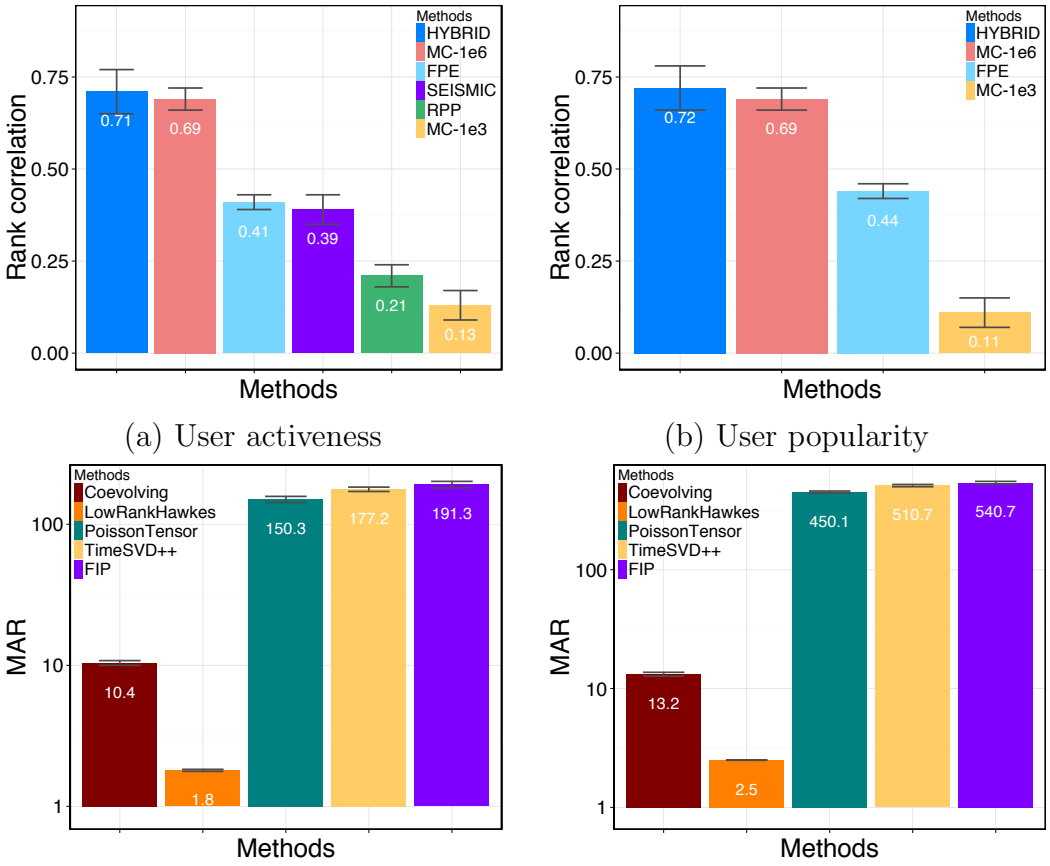


Figure 5.6: Rank correlation results in different problems. We vary the proportion p of training data from 0.6 to 0.8, and the error bar represents the variance over different training sets.

1E3. MC-1E6 has a similar predictive performance as HYBRID, but needs $10^3\times$ more samples.

Scalability. How does the reduction in sample size improve the speed? Figure 5.5(a) shows that as the error decreases from 0.5 to 0.1, MC has higher computation cost, since it needs much more samples than HYBRID to achieve the same error. We include the plots of HYBRID in (c). In particular, to achieve the error of 0.1, MC needs 10^6 samples in 27.8 hours, but HYBRID only needs 14.4 minutes with 10^3 samples. We use the machine with 16 cores, 2.4 GHz Intel Core i5 CPU and 64 GB memory.

Rank correlation. We rank all users according to the predicted level of active-

ness and level of popularity separately. Figure 5.6(a,b) show that HYBRID performs the best with the accuracy around 80%, and it consistently identifies around 30% items more correctly than FPE on both tasks.

Predicting the popularity of items in recommender systems

In the recommendation system setting, we use two datasets from [1]. The IPTV dataset contains 7,100 users' watching history of 436 TV programs in 11 months, with around 2M events. The Reddit dataset contains online discussions of 1,000 users in 1,403 groups, with 10,000 discussion events. The predictive and scalability performance are consistent with the application in social networks. Figure 5.4 shows that HYBRID is 15% more accurate than FPE and 20% than SEISMIC. Figure 5.5 also shows that HYBRID needs much smaller amount of computation time than MC-1E6. To achieve the error of 0.1, it takes 9.8 minutes for HYBRID and 7.5 hours for MC-1E6. Figure 5.6(c,d) show that HYBRID achieves the rank correlation accuracy of 77%, with 20% improvement over FPE.

5.5.2 Experiments on Synthetic Data

We compare HYBRID with MC in two aspects: (i) the significance of the reduction in the error and sample size, and (ii) estimators of the probability mass function. We study a Hawkes process and set the parameters of its intensity function as $\eta = 1.2$, and $\alpha = 0.5$. We fix the prediction time to be $t = 30$. The ground truth is computed with 10^8 samples from MC simulations.

Error vs. number of samples. In four tasks with different f, Figure 5.7 shows that given the same number of samples, HYBRID has a smaller error. Moreover, to achieve the same error, HYBRID needs 100× less samples than MC. In particular, to achieve the error of 0.01, (a) shows HYBRID needs 10^3 and MC needs 10^5 samples; (b) shows HYBRID needs 10^4 and MC needs 10^6 samples.

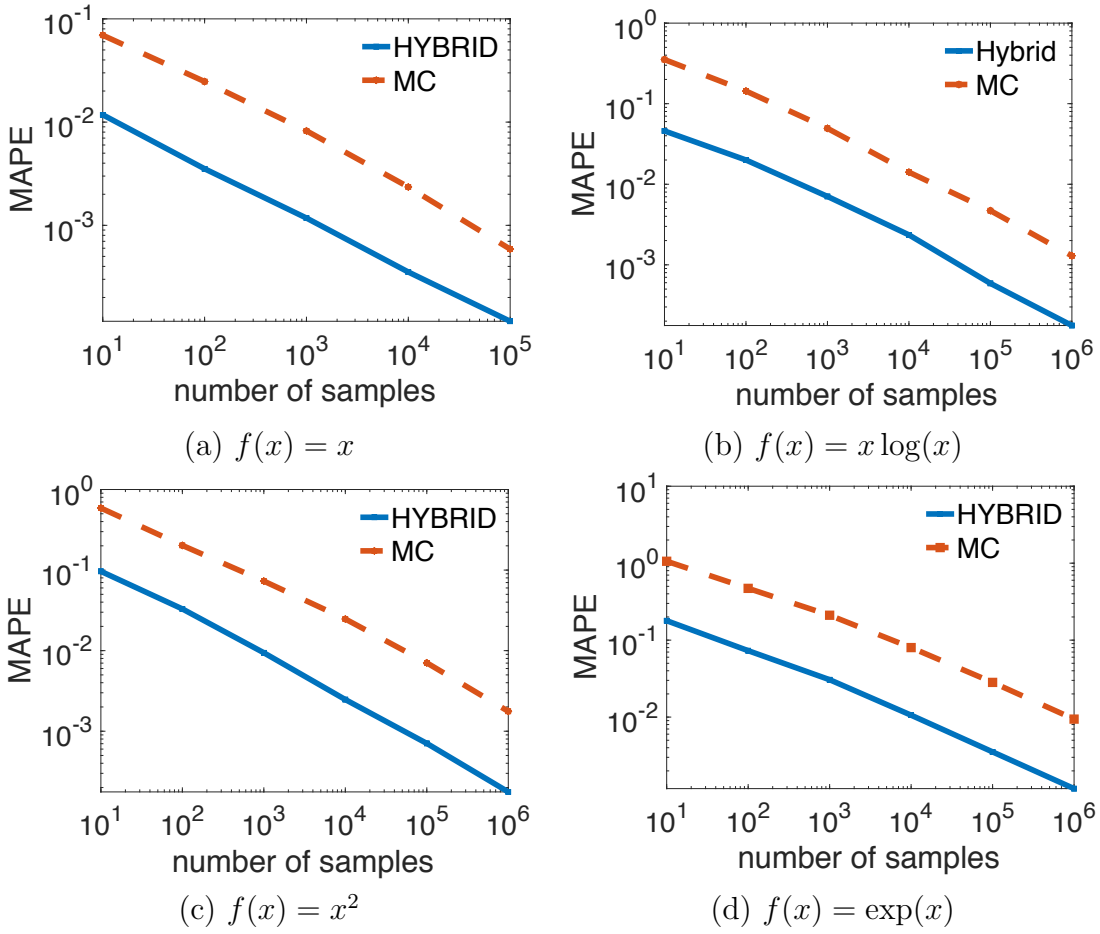
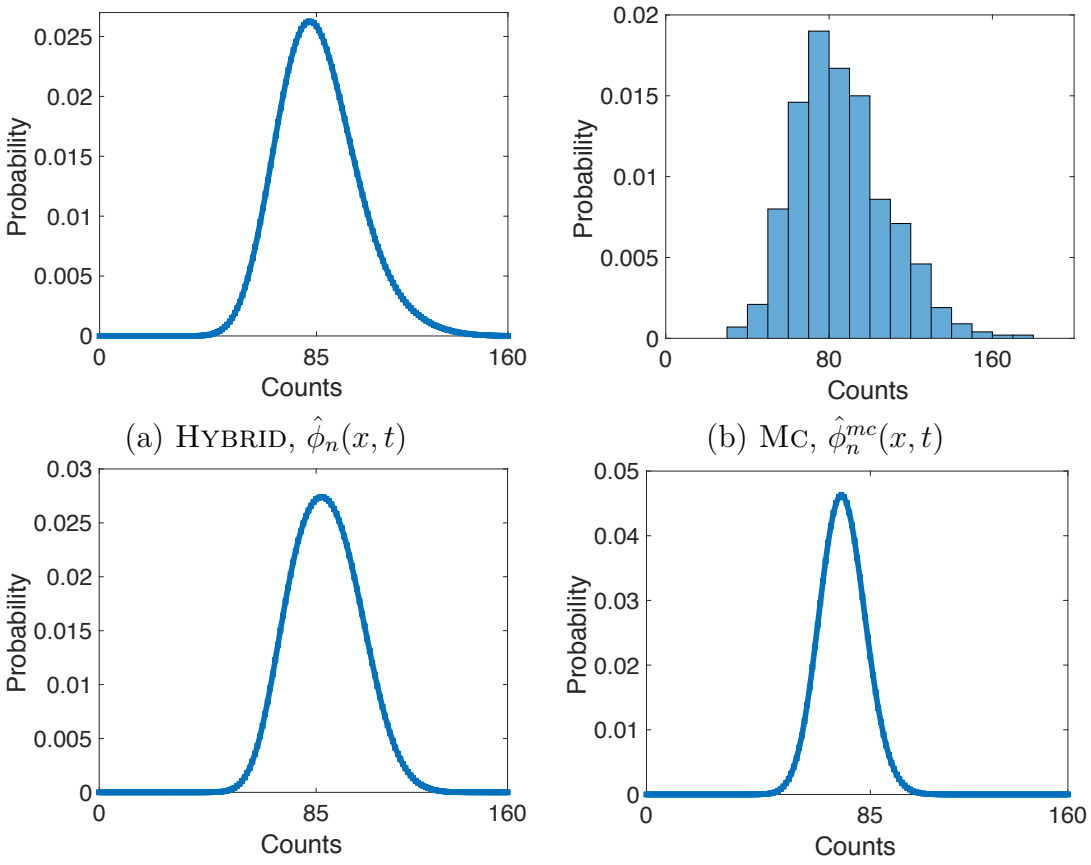


Figure 5.7: Error of $\mathbb{E}[f(N(t))]$ as a function of sample size (loglog scale). (a-d) different choices of f.

Probability mass functions. We compare our estimator of the probability mass with MC. Figure 5.8(a,b) show that our estimator is much smoother than MC, because our estimator is the average of conditional probability mass functions, which are computed by solving the mass transport equation. Moreover, our estimator centers around 85, which is the ground truth of $\mathbb{E}[N(t)]$, while that of MC centers around 80. Hence HYBRID is more accurate. We also plot two conditional mass functions in (c,d). The average of 1000 conditional mass functions yields (a). Thus, this averaging procedure in HYBRID adjusts the shape of the estimated probability mass. On the contrary, given one sample, the estimator in MC is just an indicator function and cannot capture the shape of the probability mass.



(a) HYBRID, $\hat{\phi}_n(x,t)$ (b) MC, $\hat{\phi}_n^{mc}(x,t)$

(c) HYBRID, 1 sample (d) HYBRID, 1 sample

Figure 5.8: Comparison of estimators of probability mass functions in HYBRID and MC. (a,b) estimators with the same 1000 samples. (c,d) estimator with one sample in HYBRID.

5.6 Summary

In this chapter, we have proposed HYBRID, a generic framework with a new formulation of the prediction problem in point processes and a novel mass transport equation. This equation efficiently uses the event information to update the transport rate and compute the conditional mass function. Moreover, HYBRID is applicable to general point processes and prediction tasks with an arbitrary function f. Hence it can take any point process models as input, and the predictive performance of our framework can be further improved with the advancement of point process models. Experiments on real world and synthetic data demonstrate that HYBRID outperforms the

state of the art both in terms of accuracy and efficiency. There are many interesting lines for future research. For example, HYBRID can be generalized to marked point processes [106], where a mark is observed along with the timing of each event.

Besides understanding and predicting users' temporal behaviors, it is also important to design "closed loop" systems that can incorporate users' feedback adaptively and help users make better strategic decisions. In the next chapter, I develop an efficient framework to incorporate users' feedback when designing optimal policies.

CHAPTER 6
A STOCHASTIC DIFFERENTIAL EQUATION FRAMEWORK FOR GUIDING ONLINE USER ACTIVITIES IN CLOSED LOOP

6.1 Introduction

Online social and information platforms have brought to the public a new style of social lives: people use these platforms to receive information, create content, and share opinions. The large-scale temporal event data generated by online users have created new research avenues and scientific questions at the intersection of social sciences and machine learning. These questions are directly related to the development of new models as well as learning and inference algorithms to understand and predict user activities from these data [111, 99, 112, 113, 114, 94, 95, 115].

Recently, point processes have been widely applied to model user activities. Instead of discretizing time into intervals, these models treat timestamps as random variables, and propose a mechanism to link model parameters with the observed timestamps. Such fine-grained modeling of temporal information has led to improved predictive performance in diverse applications, such as information diffusion [1, 2, 28, 35, 29], recommender systems [1, 5], and evolutionary networks [4]. However, most works deal with the "open loop" setting where models are used mainly for prediction, but user feedbacks are not incorporated into the model. Typically, we are interested in the "closed loop" setting where we want to design a policy to guide user activities and incorporate feedbacks timely. It is not clear how the current models can be used for such "closed loop" task.

For instance, a decision maker seeks to determine the best intervention policy such that the sentiment in user generated contents can be guided towards a positive

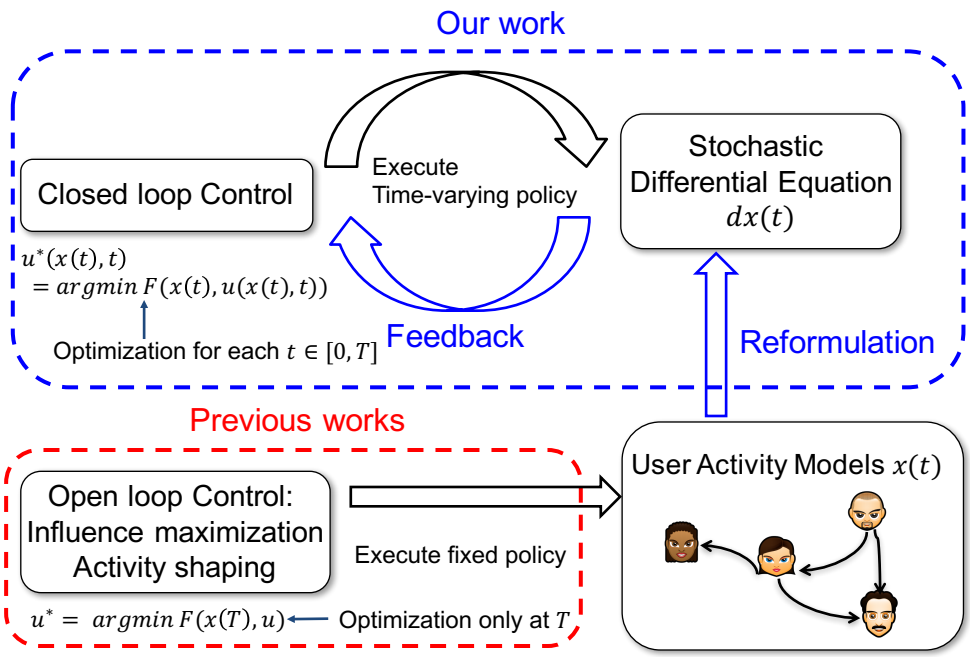


Figure 6.1: Comparison between our work and previous works in guiding user activities. Given a user activity model that is learned from observed behaviors, our "closed loop" guiding framework aims to find an optimal policy $u(x(t),t) : \Re \times \Re \to \Re$ that maps a user's current state $x(t)$, *e.g.*, opinion, to an action. On the contrary, previous works are "open loop": they only optimize the objective function at terminal time and compute a fixed scalar policy $u \in \Re$.

state. This is significantly more challenging than traditional influence maximization or activity shaping tasks [35, 116, 117, 118, 119], where the policy is determined before the process unfolds, and they do not take into account the instantiation of the process. A framework for doing this is critically important for understanding the vulnerabilities of information networks, designing policies to suppress the spread of undesired contents, and exploring new ways of performing content recommendation. For instance, a network moderator may want to effectively contain the spread of rumors and misinformation, and an online platform may want to promote the level of long-term user activities rather than the short-term click-through rate.

In this chapter, we provide a novel view of point process models, and reformulate them into stochastic differential equations (SDEs). This reformulation allows us to

significantly generalize existing point process models to SDEs, and plays a critical role in connecting the task of guiding user activities to stochastic optimal control theory, which are often used in robotics. Hence, we can bring in and extend lots of tools from stochastic optimal control literature to address the "closed loop" sequential decision making problem. Figure 6.1 illustrates our framework.

Interestingly, these problems also introduce new technical challenges. Previous works in stochastic optimal control study SDEs driven by Wiener processes and/or Poisson processes [120, 121, 122, 123]. Online user activity modeling requires us to consider more advanced processes, such as (i) Hawkes processes for long term memory and mutual exciting phenomena in social interactions, (ii) survival processes [54] for self-terminating behavior in influence propagation and link creation, and (iii) node birth processes for evolving networks [4, 124]. Thus, many technical results from control theory need to be extended for the activity guiding problem. In summary, we make the following contributions:

- We provide a general way to reformulate point process based user activity models into stochastic differential equations (SDEs), which allows us to bring in and extend tools from stochastic control literature to address the "closed loop" activity guiding problem.
- We extend technical results in stochastic optimal control theory and derive generalized Ito's lemma and HJB equation for SDEs driven by more general point processes.
- We propose an algorithm that efficiently guides the dynamics of user activities towards a target. The algorithm can also deal with the challenging cases of time-varying networks with node births.

Finally, with synthetic and real networks, we showed our framework is robust, able to steer user activities to desired states with faster convergence rate and less variance than the state-of-art methods.

Further related work. Most previous works in influence maximization [111, 118, 119] focus on selecting sources to maximize the spread of information in infinite time, which is an "open loop" setting. There is typically no quantitative prescription on how much incentive should be provided to each user. Moreover, in most cases, a finite time window must be considered. For example, a politician would like to maximize his support by a million people in one week instead of fifty years.

Another relevant area is optimal control for epidemic processes [125, 114, 126, 100]. However, the epidemic processes are modeled by deterministic differential equations. Therefore, these works neither consider the influence of abrupt event nor the stochastic nature of user behaviors in social networks. Hawkes process models [35, 127] overcome this limitation by treating the user-generated event time as a random variable and model users' stochastic behaviors. They address the problem of designing the base intensity of Hawkes process to achieve a desired steady behavior. However, these methods are open loop and do not consider user feedbacks, and the variance of the dynamics can still be very large. Recently, [124] studied the problem of controlling network growth in the macroscopic level, while we focus on guiding users' microscopic temporal behaviors based on point processes and SDEs.

Background on SDEs. A jump diffusion SDE is a differential equation in which one or more terms is a stochastic process: $\mathrm{d}x(t) = f(x)\mathrm{d}t + g(x)\mathrm{d}w(t) + h(x)\mathrm{d}N(t)$, where $\mathrm{d}x(t) := x(t + \mathrm{d}t) - x(t)$ is the differential of $x(t)$. This SDE contains a drift, a diffusion and a jump term: the drift term $f(x)\mathrm{d}t$ models the intrinsic evolution of $x(t)$; the diffusion Wiener process $w(t) \sim \mathcal{N}(0, t)$ captures the Gaussian noise; the jump point process $N(t)$ captures the influence of abrupt events.

6.2 Stochastic Differential Equations for User Activity Models

In this section, we establish a framework to reformulate many point process models into SDEs. This reformulation framework plays a critical role in connecting the task

of guiding user activities to stochastic optimal control theory often used in robotics and designing closed loop policies.

6.2.1 User Activity Models

We first introduce the generic point process models for user activities, and present three examples.

Definition 10 (User Activity Model)**.** *For a network with U users, the point process $N_i(t)$ models the generation of event times from user i, and its intensity f is defined as*

$$\lambda_i(t) = \eta_i(t) + \sum_{j=1}^{U} \beta_{ij} \sum_{t_j \in \mathcal{H}_j(t)} \kappa_{\omega_1}(t - t_j), \tag{6.1}$$

where $\eta_i(t)$ is the base intensity, $\beta_{ij} \geqslant 0$ models the strength of influence from user j to i, $\mathcal{H}_j(t)$ is the history of events for user j, and $\kappa_{\omega_1}(t) = \exp(-\omega_1 t)$ is the triggering kernel capturing the influence of past event. We also assume the additional event feature/content $x_i(t)$ follows the model

$$x_i(t) = b_i(t) + \sum_{j=1}^{U} \alpha_{ij} \sum_{t_j \in \mathcal{H}_j(t)} \kappa_{\omega_2}(t - t_j)\, h(x_j(t_j)) \tag{6.2}$$

where $b_i(t)$ is the base content, α_{ij} is the influence from user j to i, the function $h(\cdot)$ captures the influence of the activity content and typical forms include $h = 1$ and $h(x) = x$.

This model generalizes point processes since it not only models the on-off temporal behavior, but also the activity content $x_i(t)$, *e.g.*, opinion or a vector of interested topics. It captures both exogenous and endogenous properties of networks. The exogenous term is the base intensity/activity content, and the endogenous term captures the fact that one's intensity/activity content is influenced by neighbors. α_{ij}, β_{ij} measure the strength of such influence, the kernel $\kappa_\omega(t)$ captures the decay of influ-

ence of past events/content over time, and the summation captures the influence of neighbors' events. Next we present three examples.

- **Continuous-time information propagation** [28]. The information propagation in social networks begins with a set of source nodes, and the contagion is transmitted from the sources along their out-going edges to their direct neighbors. We set $N_{ij}(t)$ to be the *survival process* capturing the infection on the edge $i \to j$, and $N_{ij}(t) = 1$ means node j is infected by node i at time t. Since there is only one event for an instantiation of this process, we set the infection intensity as follows:

$$\lambda_{ij}(t) = \eta_{ij}(1 - N_{ij}(t)) \tag{6.3}$$

  where η_{ij} is intensity that i infects j, and $(1-N_{ji}(t))$ ensures the infection happens only once.

- **Hawkes processes** [36]. This model captures the mutual excitation phenomena between events, and has been successfully applied to analyze user behaviors in social networks [5, 35, 4, 1, 2, 6]. The process $N_i(t)$ counts the number of events generated by user i up to time t and its intensity $\lambda_i(t)$ models mutual excitation between the events from a collection of U users as follows:

$$\lambda_i(t) = \eta_i + \sum\nolimits_{j=1}^{U} \beta_{ij} \sum\nolimits_{t_j \in \mathcal{H}_j(t)} \kappa_{\omega_1}(t - t_j)$$

  Here, the occurrence of each historical event from one's neighbors increases the intensity by a certain amount determined by $\kappa_{\omega_1}(t)$ and α_{ij}.

- **Opinion dynamics** [29, 127]. This model considers both the timing and content of each event. It assigns each user i a Hawkes intensity $\lambda_i(t)$ to generate events and an opinion process $x_i(t)$, where $x_i(t) = 0$ corresponds to neutral opinion and large positive/negative values correspond to extreme opinions. The opinion of user i is

modeled as a temporally discounted average of neighbors' opinion:

$$x_i(t) = b_i + \sum\nolimits_j \alpha_{ij} \sum\nolimits_{t_j \in \mathcal{H}_j(t)} \kappa_{\omega_2}(t - t_j) x_j(t_j)$$

This model has superior performance in predicting opinions. However, it is not clear whether it can be used to design feedback policies to guide the dynamics precisely to some target states.

6.2.2 Equivalent SDE Reformulation

We are now ready to show the novel SDE reformulation of the user activity model in Definition 10.

Theorem 11 (Transformation Framework)**.** *The equivalent SDE form of the user activity model is:*

$$\begin{aligned} \mathrm{d}\lambda_i(t) &= \mathrm{d}\eta_i + \omega_1\big(\eta_i - \lambda_i\big)\mathrm{d}t + \sum\nolimits_j \beta_{ij}\mathrm{d}N_j(t) \\ \mathrm{d}x_i(t) &= \mathrm{d}b_i + \omega_2\big(b_i - x_i(t)\big)\mathrm{d}t + \sum\nolimits_j \alpha_{ij} h(x_j)\mathrm{d}N_j(t) \end{aligned} \tag{6.4}$$

Proof sketch. We first define a convolution operator and reformulate the user activity model in (6.1) and (6.2), next we apply a differential operator d to the reformulated equations and derive the differential form of $\lambda_i(t)$ and $x_i(t)$, which lead to two SDEs. Appendix C.1 contains details.

These SDEs describe how the intensity $\lambda_i(t)$ and content $x_i(t)$ change on $[t, t+\mathrm{d}t)$; each consists of three terms:

- **Baseline change**. The differential $\mathrm{d}b_i(t)$ captures the infinitesimal change of base activity content.
- **Drift**. The change rate of the activity content, $\mathrm{d}x_i(t)/\mathrm{d}t$, is proportional to $-x_i(t)$, which means the activity content $x_i(t)$ tends to stabilize over time. In fact, if ignor-

ing the jump term and $\mathrm{d}b_i(t)$, the expected activity content $\mathbb{E}[x_i(t)]$ will converge to $b_i(t)$ as t increases. This can be proved by setting $\mathbb{E}[\mathrm{d}x_i(t)] = 0$. Similarly, equation (6.4) shows a user's intensity tends to stabilize over time.

- **Jump**. This term captures the influence of each event in the network and is a weighted summation of the neighbors' influence. The coefficient α_{ij} ensures that only neighbors' effect will be considered, and $\mathrm{d}N_j(t) \in \{0, 1\}$ models whether user j generates an event. Similarly, equation (6.4) shows that user i's intensity increases by β_{ij} if his neighbor j generates an event.

  Next, we present three applications of Theorem 11.

- **SDE for continuous-time information propagation**. The intensity in (6.3) is a simplified version of (6.1) without the historical influence term. Its SDE version is:

$$\mathrm{d}\lambda_{ij}(t) = -\eta_{ij}\mathrm{d}N_{ij}(t)$$

  This SDE keeps the key property of survival process: before an infection happens, the intensity is constant, *i.e.*, $\lambda_{ij}(t) = \eta_{ij}$; after the infection happens, the intensity is 0, *i.e.*, $\lambda_{ij}(t + \mathrm{d}t) = \lambda_{ij}(t) + \mathrm{d}\lambda_{ij}(t) = 0$.

- **SDE for Hawkes process**. We set $\eta_i(t) = \eta_i$ and obtain:

$$\mathrm{d}\lambda_i(t) = \omega_1\big(\eta_i - \lambda_i(t)\big)\mathrm{d}t + \sum\nolimits_j \alpha_{ij}\mathrm{d}N_j(t)$$

  This SDE shows that the user's intensity tends to stabilize and its change is influenced by his neighbors' activities.

- **SDE for opinion dynamics.** We set $b_i(t) = b_i$, $h_j(x_j) = x_j$, and further generalize this model by adding a Wiener process term:

$$\mathrm{d}x_i = \omega_2\big(b_i - x_i\big)\mathrm{d}t + \beta\mathrm{d}w_i + \sum\nolimits_j \alpha_{ij}x_j\mathrm{d}N_j$$

where the Wiener process $w_i(t)$ captures the Gaussian noise, such as fluctuations in the dynamics due to unobserved factors and activities outside the social platform. The jump term models the fact that the change of opinion is influenced by his neighbors' opinion.

6.2.3 Benefit of the SDE Modeling Framework

Our SDE formulation opens a new gate to extend tools from optimal control theory. Hence we can solve many important problems in social sciences, such as the least square activity guiding and activity maximization problem. Without this view, it is not easy to design algorithms with closed loop policies. Besides transforming an existing model to an SDE, we can also directly design an SDE to model many other factors. For example, one can model the Gaussian noise by adding a Wiener process to an SDE. Next, we show how to optimally control the SDE to guide user activities.

6.3 A Convex Activity Guiding Framework

In this section, we define the activity guiding problem. Let $\boldsymbol{x}(t) = (x_1(t), \cdots, x_U(t))^\top$ be the vector of each user's activity content, then we study the vector version of the SDE in (6.4) and reformulate this SDE with an extra control policy $\boldsymbol{u}(\boldsymbol{x}(t), t) : \Re^U \times \Re \to \Re^U$ as follows.

$$\mathrm{d}\boldsymbol{x} = (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u})\mathrm{d}t + \boldsymbol{g}(\boldsymbol{x})\mathrm{d}\boldsymbol{w}(t) + \boldsymbol{h}(\boldsymbol{x})\mathrm{d}\boldsymbol{N}(t) \tag{6.5}$$

This policy $\boldsymbol{u}$ maps the activity content $\boldsymbol{x}$ to an action for all users. Next we present two examples.

- **Guiding Hawkes process.** To steer user activities to a desired level, we provide incentives to users and add a control policy $u_i(\lambda_i(t), t)$ to the SDE formulation of

the Hawkes intensity function as follows:

$$\mathrm{d}\lambda_i(t) = (\eta_i + u_i(\lambda_i(t), t) - \lambda_i(t))\mathrm{d}t + \sum\nolimits_j \beta_{ij}\mathrm{d}N_j(t),$$

where $u_i(\lambda_i(t), t)$ captures the amount of additional influence to change the baseline intensity η_i.

- **Guiding opinion dynamics.** We can guide the opinion SDE with a policy $u_i(x_i(t), t)$ as follows:

$$\mathrm{d}x_i = (b_i + u_i - x_i)\mathrm{d}t + \theta\mathrm{d}w_i + \sum\nolimits_j \alpha_{ij}x_j\mathrm{d}N_j \tag{6.6}$$

where $u_i(x_i, t)$ determines how fast the opinion needs to be changed for user i. For example, a network moderator may request the user to change his opinion from -3 to 1 in one day, and this control policy quantifies the amount of change in unit time.

Next we present the objective for optimizing the SDE.

Definition 12 (Stochastic User Activity Guiding Problem)**.** *For the SDE in (6.5), given the initial condition $(\boldsymbol{x}_0, t_0)$, we aim to find an optimal policy $\boldsymbol{u}^*(\boldsymbol{x}, t)$ for $t \in (t_0, T]$, which minimizes the convex objective function:*

$$V(\boldsymbol{x}_0, t_0) := \min_{\boldsymbol{u}} \mathbb{E}\Big[\phi\big(\boldsymbol{x}(T)\big) + \int_{t_0}^{T} \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t)\mathrm{d}t\Big] \tag{6.7}$$

where V is called the value function that summarizes the optimal expected cost if $\boldsymbol{u}^*$ is executed from t_0 to T. It is a function of the initial state. The expectation $\mathbb{E}$ is over stochastic processes $\{\boldsymbol{w}(t), \boldsymbol{N}(t)\}$ for $t \in (t_0, T]$. ϕ is the terminal cost and $\mathcal{L}$ is the instantaneous cost.

Terminal cost ϕ. It is the cost at final time T. We discuss several functional forms as follows:

- **Least square guiding.** The goal is to guide the expected state to a pre-specified target $\boldsymbol{a}$. For opinion dynamics, the goal can be to ensure nobody believes the rumor. Mathematically, we set $\phi = \|\boldsymbol{x}(T) - \boldsymbol{a}\|^2$. Moreover, to influence users' intensity of generating events, one can set the desired level of the intensity to be at a high level $\boldsymbol{a}$ and conduct activity guiding: $\phi = \|\boldsymbol{\lambda}(T) - \boldsymbol{a}\|^2$.
- **Information/activity maximization.** The goal is to maximize the activity content of all users. For example, the goal for an educator is to maximize the students' recognition of the value of education, and we set $\phi = -\sum_u x_u(T)$ to maximize each user's positive opinion. Moreover, to improve the activity level in social platforms, one can also maximize the intensity: $\phi = -\sum_u \lambda_u(T)$.

Instantaneous cost $\mathcal{L}$. This is the cost at $t \in [t_0, T]$ and in the form of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) = q(\boldsymbol{x}(t)) + \rho c(\boldsymbol{u}(t))$. The state cost $q(\boldsymbol{x}(t))$ is optional and the control cost $c(\boldsymbol{u}(t)) = \|\boldsymbol{u}(t)\|^2$ is necessary. We set $q = 0$ if the cost only occurs at T; otherwise q captures the cost at the intermediate time, *e.g.*, the cost incurred by maximizing students' positive recognition of the value of education over consecutive weeks. Its function form is the same as the terminal cost: $q = \phi$. The control cost captures the fact that the policy costs money or human efforts. The scalar ρ is the trade-off between control cost and state cost.

Solving this activity guiding problem is challenging, since the objective function involves taking expectation over complex stochastic processes. Furthermore, it is a functional optimization problem since the optimal policy is a function of both state and time. Fortunately, the SDE formulations allow us to connect the problem to that of stochastic dynamic programming methods. As a result, we can extend lots of tools in stochastic optimal control to address sequential decision making problems.

6.4 Algorithm for Computing Optimal Policy

In this section, we will find the optimal policy posed in (6.7). Prior works in control theory mostly study the SDE where the jump is a Poisson process [120, 123, 122, 121]. However, in our model, the jump process is a more complex point process with *stochastic intensity functions*, *e.g.*, Hawkes process. Hence significant generalizations, both in theory and algorithms, are needed. We first derive the HJB partial differential equation (PDE) for a deterministic system, then generalize the procedure to the activity guiding problem. Further, we extend our framework to guide user activities in the challenging time-varying networks.

6.4.1 HJB Equation for Deterministic Systems

To obtain the optimal policy, we need to compute the value function V in (6.7) subject to the constraint of the SDE. We will break down the complex optimization into simpler subproblems. First, the initial condition $\boldsymbol{x}(t_0)$ needs to be replaced by an arbitrary start $\boldsymbol{x}(t)$, so that the start can be analytically manipulated and we obtain a time-varying objective $V(\boldsymbol{x}, t)$ amenable to analysis. Next, since the value function consists of an integral term, we break the integral into $[t, t + \mathrm{d}t]$ and $[t + \mathrm{d}t, T]$. If the system is deterministic, we can further split the value function as:

$$V(\boldsymbol{x}, t) = \underbrace{\min_{\boldsymbol{u}} \left[\phi + \int_{t+\mathrm{d}t}^{T-} \mathcal{L} \, \mathrm{d}\tau \right]}_{V(\boldsymbol{x}(t+\mathrm{d}t), t+\mathrm{d}t)} + \underbrace{\int_{t}^{t+\mathrm{d}t} \mathcal{L} \, \mathrm{d}\tau}_{\text{cost } t \to t+\mathrm{d}t} \Bigg] \tag{6.8}$$

The first term is $V(\boldsymbol{x}(t + \mathrm{d}t), t + \mathrm{d}t)$ and the second term is the optimal cost on $[t, t + \mathrm{d}t]$. Hence (6.8) follows the structure of a dynamic programming and we can solve it recursively: given $V(\boldsymbol{x}(t + \mathrm{d}t), t + \mathrm{d}t)$, we only need to proceed optimally on $[t, t + \mathrm{d}t]$ to compute V backward.

To further simplify (6.8), we perform deterministic Taylor expansion up to second

order of the first term on right-hand side as $V(\boldsymbol{x}(t+\mathrm{d}t), t+\mathrm{d}t) := V(\boldsymbol{x}, t) + \mathrm{d}V(\boldsymbol{x}, t)$, where $\mathrm{d}V = V_t \mathrm{d}t + V_x^\top \mathrm{d}\boldsymbol{x} + \frac{1}{2}\mathrm{d}\boldsymbol{x}^\top V_{xx} \mathrm{d}\boldsymbol{x} + \mathrm{d}\boldsymbol{x}^\top V_{xt} \mathrm{d}t + \frac{1}{2} V_{tt} \mathrm{d}t^2$. Then we can cancel $V(\boldsymbol{x}, t)$ on both sides of (6.8), divide it by $\mathrm{d}t$, and take the limit as $\mathrm{d}t \to 0$. Since $\mathrm{d}\boldsymbol{x} = \boldsymbol{x}'(t)\mathrm{d}t$, all the second order term in $\mathrm{d}V$ goes to 0. Hence we obtain the Hamilton-Jacobi-Bellman (HJB) equation:

$$-V_t = \min_u [\boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{u}, t) + V_x^\top \boldsymbol{x}']$$

However, our system is stochastic and the above procedure needs to be generalized significantly.

6.4.2 HJB Equation for Guiding User Activities

To derive the HJB equation for our model, we need to address two challenges: (i) computing the stochastic Taylor expansion $\mathrm{d}V$ under our SDE, and (ii) taking expectation of the stochastic terms in (6.8) before minimization. We first derive Theorem 13 to compute $\mathrm{d}V$, which generalizes Ito's Lemma and is applicable to SDEs driven by point processes with stochastic intensity functions. Then we compute the expectation and derive a HJB equation in Theorem 14.

Theorem 13 (Generalized Ito's Lemma for Jump Diffusion SDEs)**.** *Given the SDE in (6.5), let $V(\boldsymbol{x}, t)$ be twice differentiable in $\boldsymbol{x}$ and once in t, then we have*

$$\begin{aligned} \mathrm{d}V = &\Big\{ V_t + \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}} \boldsymbol{g}\boldsymbol{g}^\top) + V_{\boldsymbol{x}}^\top (\boldsymbol{f} + \boldsymbol{u}) \Big\} \mathrm{d}t + V_{\boldsymbol{x}}^\top \boldsymbol{g} \mathrm{d}\boldsymbol{w} \\ &+ \big(V(\boldsymbol{x} + \boldsymbol{h}, t) - V(\boldsymbol{x}, t) \big) \mathrm{d}\boldsymbol{N}(t) \end{aligned} \tag{6.9}$$

Theorem 14 (HJB Equation)**.** *Let $\boldsymbol{h}_j(\boldsymbol{x}, t)$ be the j-th column of $\boldsymbol{h}(\boldsymbol{x}, t)$. Then the*

HJB Partial Differential Equation for the user activity guiding problem is

$$-V_t = \min_{\boldsymbol{u}} \Big[\mathcal{L} + \frac{1}{2}\mathrm{tr}\big(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top\big) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u}) + \sum\nolimits_j \lambda_j(t)\big(V(\boldsymbol{x}+\boldsymbol{h}_j(\boldsymbol{x},t),t) - V(\boldsymbol{x},t)\big)\Big] \tag{6.10}$$

To prove Theorem 13, we derive a new set of stochastic calculus rules for general point processes and conduct stochastic Taylor expansion in the Ito's mean square limit sense. Appendix C.2 contains details. To prove Theorem 14, we combine the Generalized Ito's Lemma with the property of point processes. Appendix C.3 contains details. Next, we solve the HJB equation to obtain the value function V. We will show that under the optimal parameterizations of V, this HJB equation can be solved efficiently.

6.4.3 Parameterization of the Value Function

To solve the HJB equation, we first show the structure of the value function in the following proposition.

Proposition 15. *If the SDE in (6.5) is linear in $\boldsymbol{x}$, and the terminal and instantaneous cost are quadratic/linear in $\boldsymbol{x}$, then the value function $V(\boldsymbol{x},t)$ in (6.7) must be quadratic/linear.*

This result is intuitive since the V is the optimal value of the summation of quadratic/linear functions. Appendix C.4 contains the proof. This proposition is applicable to two important problems, including the least square activity guiding problem and the linear activity maximization problem.

In this section, we present derivations for the least square guiding problem, and Appendix C.6 contains derivations for the activity maximization problem. Specifically, we set $V(\boldsymbol{x},t)$ to be quadratic in $\boldsymbol{x}$ with unknown coefficients $\boldsymbol{v}_1(t) \in \Re^U$,

Algorithm 7 Optimal Control Policy

1: **Input:** network $\boldsymbol{A} = (\alpha_{ij})$, model parameters $\{\boldsymbol{\eta}, \theta, \boldsymbol{b}\}$, timestamps $\{\tau_k\}_{k=1}^m$, events $\{t_i\}_{i=1}^n$, target $\boldsymbol{a}$
2: **Output:** $\boldsymbol{v}_{11}(\tau_k), \boldsymbol{v}_1(\tau_k)$, $k = 1, \cdots, m$
3: **for** $k = 1$ **to** m **do**
4: Compute $\boldsymbol{\lambda}_u(\tau_k) = \eta_u + \sum_{j:t_i<\tau_k} \alpha_{uu_i}\kappa(\tau_k - t_i)$, $\boldsymbol{\Lambda}(\tau_k) = \sum_{j=1}^U \lambda_j(\tau_k)\boldsymbol{B}^j$ for each user u
5: **end for**
6: Compute $\boldsymbol{v}_{11}(\tau_k), \boldsymbol{v}_1(\tau_k)$ using Ode45; then compute $\boldsymbol{u}(\tau_k)$ in (6.11).

$\boldsymbol{v}_{11}(t) \in \Re^{U\times U}$ and $v_0(t) \in \Re$:

$$V(\boldsymbol{x}, t) = v_0(t) + \boldsymbol{v}_1(t)^\top \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{v}_{11}(t)\boldsymbol{x}$$

To find the optimal control, we substitute $V(\boldsymbol{x}, t)$ to the HJB equation in (6.10) and set the gradient of its right-hand-side to $\mathbf{0}$. This yields the optimal feedback control policy:

$$\boldsymbol{u}^*(\boldsymbol{x}(t), t) = -V_{\boldsymbol{x}}/\rho = -\big(\boldsymbol{v}_1(t) + \boldsymbol{v}_{11}(t)\boldsymbol{x}(t)\big)/\rho \tag{6.11}$$

This policy consists of two terms: the feedforward term $\boldsymbol{v}_1(t)$ controls the system as time goes by; the feedback term updates the policy based on the current state $\boldsymbol{x}(t)$. Moreover, ρ controls the tradeoff between control and state cost, and $\rho \to \infty$ means low budget; hence $\boldsymbol{u}^* \to \mathbf{0}$.

6.4.4 Stochastic Optimal Control Algorithm

Given the form of optimal policy $\boldsymbol{u}^*$ in (6.11), the final step is to compute its unknown coefficients $\{\boldsymbol{v}_1(t), \boldsymbol{v}_{11}(t)\}$. We substitute this optimal policy to the HJB equation in (6.10). Since the value function $V(\boldsymbol{x}, t)$ is quadratic, we can separate the HJB equation into terms that are scalar, linear, and quadratic in $\boldsymbol{x}$. Grouping coefficients

of these terms leads to two Ordinary Differential Equations (ODEs) as follows:

$$-\boldsymbol{v}_{11}' = \boldsymbol{I} + 2\boldsymbol{v}_{11}(-1+\boldsymbol{\Lambda}) - \frac{\boldsymbol{v}_{11}^2}{\rho} + \sum\nolimits_j \lambda_j \boldsymbol{B}^{j\top}\boldsymbol{v}_{11}\boldsymbol{B}^j$$
$$-\boldsymbol{v}_1'(t) = -\boldsymbol{a} + \big(-1 + \boldsymbol{\Lambda}^\top - \boldsymbol{v}_{11}(t)/\rho\big)\boldsymbol{v}_1(t) + \boldsymbol{v}_{11}(t)\boldsymbol{b}$$

where $\boldsymbol{\Lambda}(t) = \sum_j \lambda_j(t)\boldsymbol{B}^j$, matrix $\boldsymbol{B}^j$ has the j-th column as $(\alpha_{1j}, \cdots, \alpha_{Uj})^\top$ and zero elsewhere. The terminal conditions are $\boldsymbol{v}_{11}(T) = \boldsymbol{I}$ and $\boldsymbol{v}_1(T) = -\boldsymbol{a}$. We use the Runge-Kutta method [108] to solve the ODEs offline. Specifically, we partition $(t_0, T]$ to equally-spaced timestamps $\{\tau_k\}$ and obtain values of $\boldsymbol{v}_{11}(t), \boldsymbol{v}_1(t)$ at these timestamps. We use the ODE45 solver in MATLAB. Finally we update the policy online according to (6.11). Algorithm 7 summarizes the procedure.

6.4.5 Extensions to Time-varying Networks

Real world social networks can change over time. Users can follow or unfollow each other as time goes by and new users can join the network [4]. In this section, we extend our framework to networks with time-varying edges and node birth processes.

First, for a fixed network, the expectation in the objective function in (6.7) is over the stochastic pair $\{\boldsymbol{w}(t), \boldsymbol{N}(t)\}$ for $t \in (t_0, T]$. Since the network is stochastic now, we also need to take the expectation of the adjacency matrix $\boldsymbol{A}(t) = (\alpha_{ij}(t))$ to derive the HJB equation. Hence the input to Algorithm 7 is $\mathbb{E}[\boldsymbol{A}(t)] = (\mathbb{E}[\alpha_{ij}(t)])$ instead of $\boldsymbol{A}$. Specifically, we replace $\boldsymbol{h}_j(\boldsymbol{x})$ in the HJB equation (6.10) by $\mathbb{E}[\boldsymbol{h}_j(\boldsymbol{x})]$:

$$\sum\nolimits_j \lambda_j(t)(V(\boldsymbol{x} + \mathbb{E}[\boldsymbol{h}_j(\boldsymbol{x}, t)], t) - V(\boldsymbol{x}, t)) \tag{6.12}$$

where $\mathbb{E}[\boldsymbol{h}_j(\boldsymbol{x}, t)] = (\mathbb{E}[h_{1j}(t)], \cdots, \mathbb{E}[h_{Uj}(t)])^\top$ and $\mathbb{E}[h_{ij}(t)] = \mathbb{E}[\alpha_{ij}(t)]x_j(t)$. Next, we compute $\mathbb{E}[\alpha_{ij}(t)]$ in two types of networks.

Networks with link creation. We model the creation of link from node $i \to j$ as a survival process $\alpha_{ij}(t)$. If a link is created, $\alpha_{ij}(t) = 1$ and zero otherwise. Its

intensity function is defined as

$$\sigma_{ij}(t) = (1 - \alpha_{ij}(t))\gamma_i, \tag{6.13}$$

where the term $\gamma_i \geqslant 0$ denotes the Poisson intensity, which models the node i's own initiative to create links to others. The coefficient $1 - \alpha_{ij}(t)$ ensures a link is created only once, and intensity is set to 0 after that. Given a sequence of link creation events, we can learn $\{\gamma_i\}$ using maximum likelihood estimation [54] as follows.

Parameter estimation of the link creation process. Given data $e_i = (t_i, u_i, s_i)$, which means at time t_i node u_i is added to the network and connects to s_i, we set $\mathcal{E} = \{e_i\}$ and optimize the concave log-likelihood function to learn the parameters of the Poisson intensity $\boldsymbol{\gamma} = (\gamma_1, \cdots, \gamma_U)^\top$:

$$\max_{\boldsymbol{\gamma} \geqslant 0} \sum_{e_i \in \mathcal{E}} \log(\sigma_{u_i s_i}(t_i)) - \sum_{u,s \in [n]} \int_0^T \sigma_{us}(\tau) \mathrm{d}\tau$$

This objective function can be solved efficiently with many optimization algorithms, such as the Quasi-Newton algorithm.

Next, given the learned parameters, we obtain the following ordinary differential equation (ODE) that describes the time-evolution of $\mathbb{E}[\alpha_{ij}(t)]$:

$$\mathrm{d}\mathbb{E}[\alpha_{ij}(t)] \underset{(a)}{=} \mathbb{E}[\mathrm{d}\alpha_{ij}(t)] \underset{(b)}{=} \sigma_{ij}(t)\mathrm{d}t \underset{(c)}{=} (1 - \mathbb{E}[\alpha_{ij}(t)])\gamma_i \mathrm{d}t, \tag{6.14}$$

where (a) holds because the operator d and $\mathbb{E}$ are exchangeable, (b) is from the definition of intensity function, and (c) is from (6.13). The initial condition is $\mathbb{E}[\alpha_{ij}(0)] = 0$ since i and j are not connected initially. We can easily solve this ODE in analytical form:

$$\mathbb{E}[\alpha_{ij}(t)] = 1 - \exp(-\gamma_i t)$$

Networks with node birth. The network's dimension can grow as new users join it.

Since the dimension of $\boldsymbol{A}(t)$ changes over time, it is very challenging to control such network, and it remains unknown how to derive the HJB equation for such case. We propose an efficient method by connecting the stochasticity of the node birth process to that of link creation process. More specifically, we have the following observation.

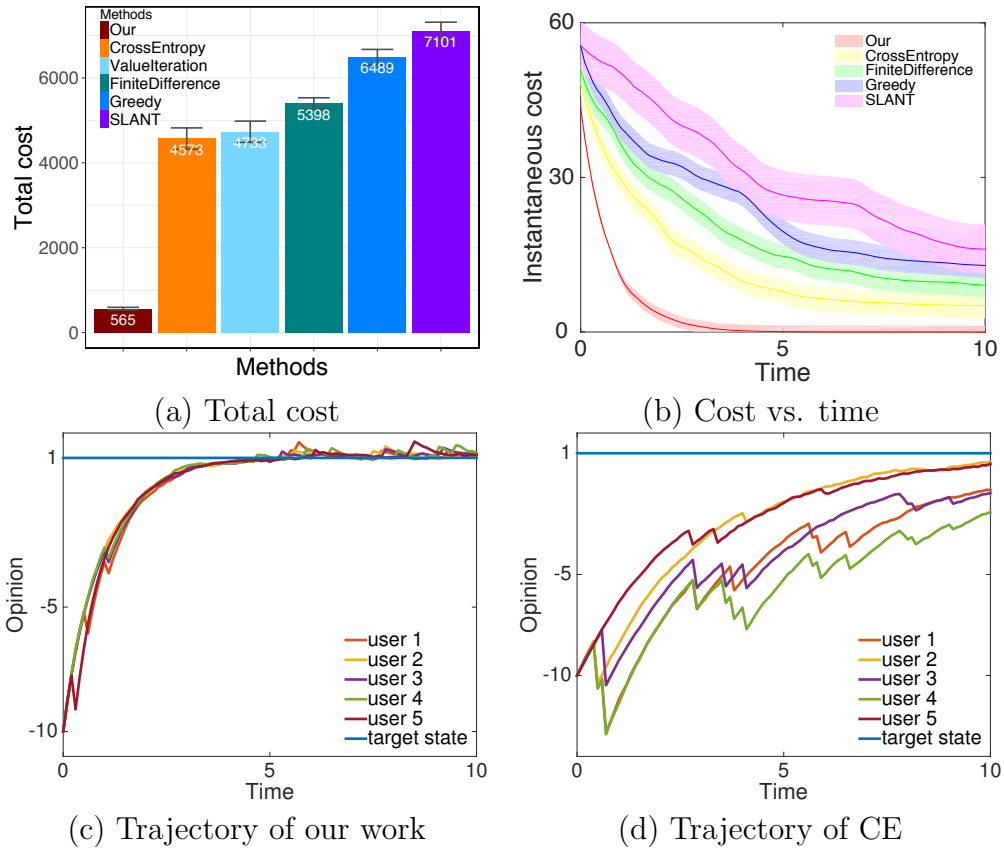
Observation. *The process of adding a new user v to the existing network $\boldsymbol{A} \in \Re^{(N-1)\times(N-1)}$ and connects to user s is equivalent to link creation process of setting $\boldsymbol{A}(t) \in \Re^{N\times N}$ to be the existing network and letting $\alpha_{vs}(t) = 1$.*

With this observation, we can fix the dimension of $\boldsymbol{A}(t)$ beforehand, and add a link whenever a user joins the network. This procedure is memory-efficient since we do not need to maintain a sequence of size-growing matrices. More importantly, we transform the stochasticity of the network's dimension to the stochasticity of link creation process with a fixed network dimension. Finally, the difference between link creation and node birth is: we control each node in the link creation case, but do not control the node until it joins the network in the node birth case.

6.5 Experiments

We focus on two tasks: least square opinion guiding (LSOG) and opinion influence maximization (OIM). We compare with the following state-of-art stochastic optimization methods that are applicable to control SDEs and point processes.

- Value Iteration (VI) [128]: we directly formulate a discrete-time version of the opinion dynamics and compute the optimal policy using the value iteration algorithm. The discretized timestamps are the same as that for solving the HJB PDE.
- Cross Entropy (CE) [129]: it samples policies from a normal distribution, sorts them in ascending order w.r.t. the cost and recomputes the distribution parameters from the first K elite samples. This procedure repeats with new distribution until costs converge.

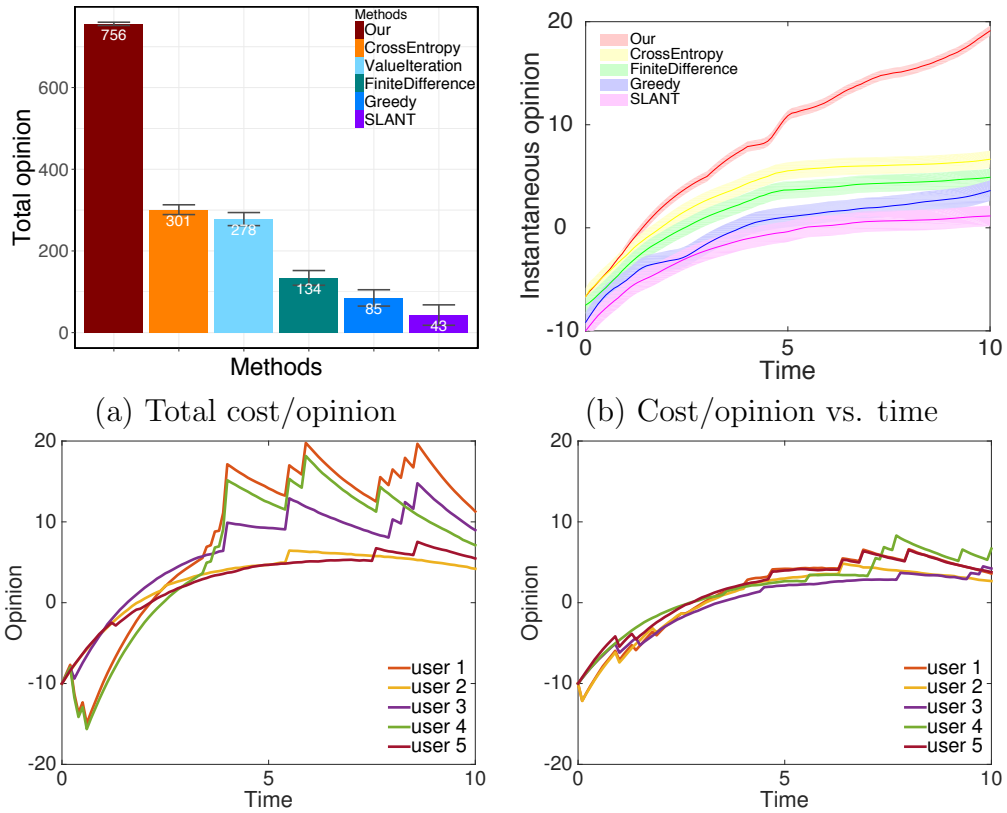


(a) Total cost (b) Cost vs. time

(c) Trajectory of our work (d) Trajectory of CE

Figure 6.2: Results in Least Square Opinion Guiding (LSOG). (a) total cost for LSOG and total opinion for OIM per user. Error bar is the variance; (b) instantaneous cost/opinion per user. Line is the mean and pale region is the variance; (c,d) sample trajectories of five users.

- Finite Difference (FD) [130]: it generates samples of perturbed policies and compute perturbed costs. Then it uses them to approximate the gradient of the cost w.r.t the policy. The cost in CE and FD is evaluated by executing the policy on the SDE.
- Greedy: It controls the SDE when the state cost is high. We divide time horizon into n timestamps. At each timestamp, we compute the state cost and control the system based on pre-specified rules if current cost is more than k times of the optimal cost of our method. We vary k from 1 to 5, n from 1 to 100 and report the best performance.
- Slant [127]: It sets the open loop control policy only at the initial time.



(a) Total cost/opinion

(b) Cost/opinion vs. time

(c) Trajectory of our work

(d) Trajectory of CE

Figure 6.3: Results in Opinion Influence Maximization (OIM). (a) total opinion for OIM per user. Error bar is the variance; (b) instantaneous cost/opinion per user. Line is the mean and pale region is the variance; (c,d) sample trajectories of five users.

6.5.1 Experiments on Synthetic Data

We generate a synthetic a network with 1000 users, where the topology matrix is randomly generated with a sparsity of 0.001. We simulate the opinion SDE on $[0, 10]$ using the Euler method [123] to compute its difference form. The time window is divided into 100 equally spaced intervals. We set the base opinion uniformly at random, $b_i \sim \mathcal{U}[-1, 1]$, $\omega = 1$, noise level $\theta = 0.2$, $\alpha_{ij} \sim \mathcal{U}[0, 0.01]$, and $x_i(0) = -10$. The Wiener process is simulated from the Gaussian distribution and the Hawkes process is simulated using the Thinning algorithm [59]. We set the budget level parameter $\rho = 10$, and our results generalize beyond this value. We repeat simulations of the SDE for ten times and report the averaged performance.

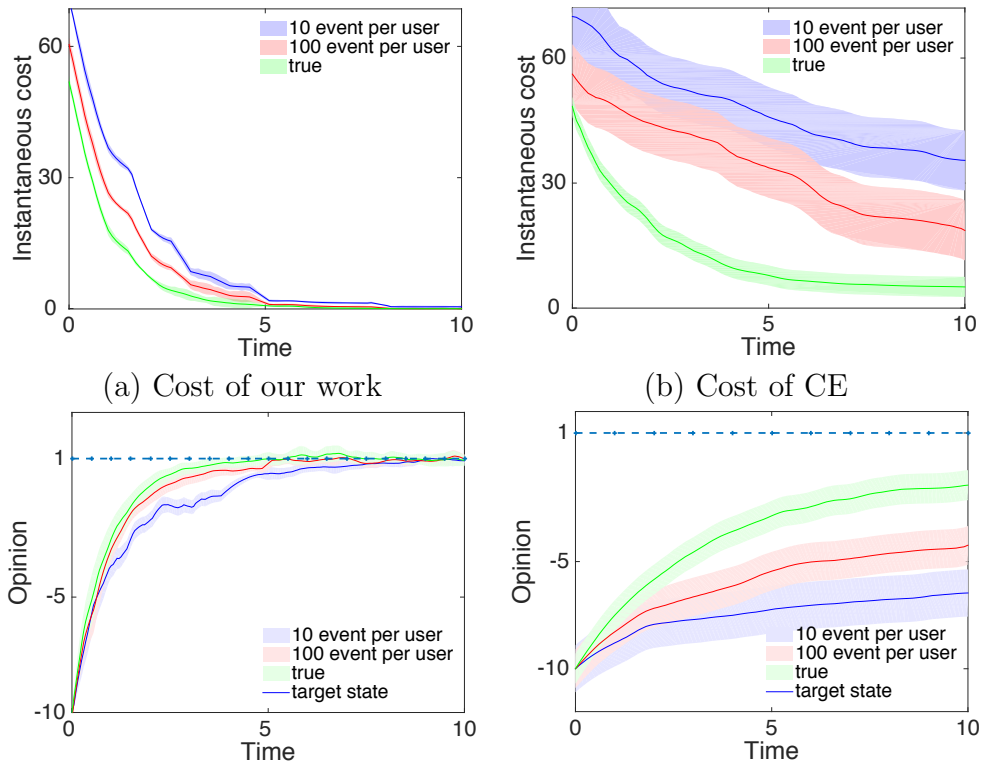
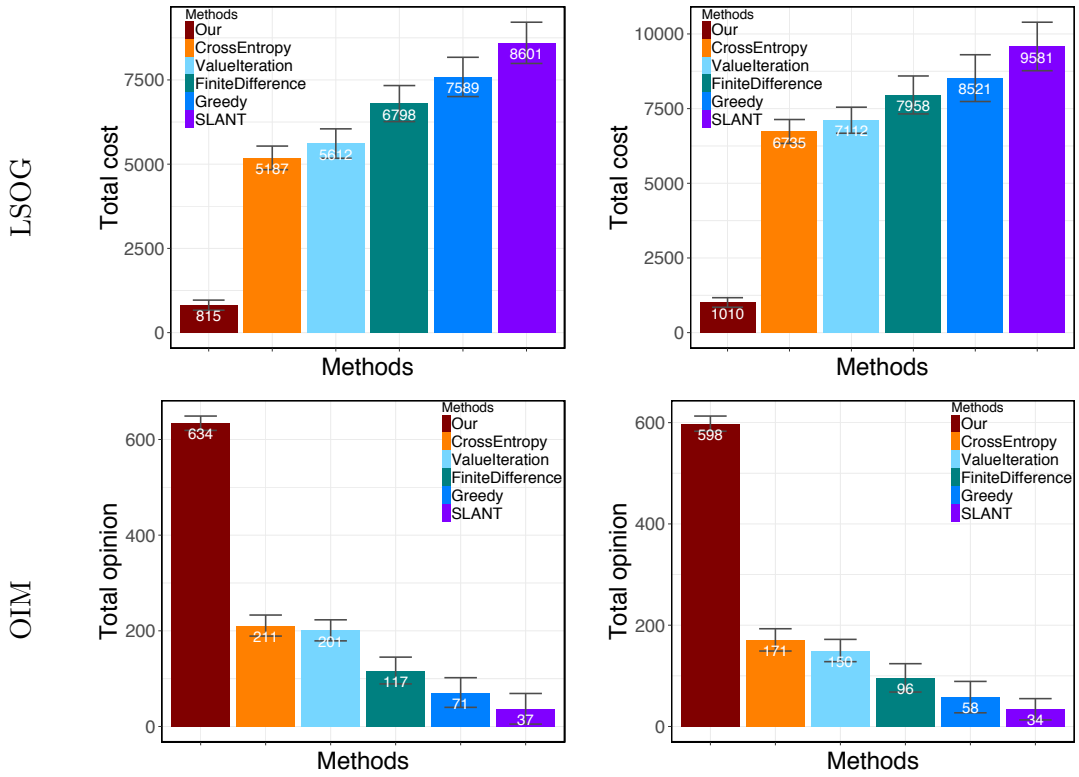


Figure 6.4: Robustness analysis when parameters are learned with different sizes of training data. (a) and (b) instantaneous cost; (c) and (d) opinion trajectory for one randomly sampled user.

Total cost. For LSOG, we set the target $a_i = 1$. The total cost per user is computed by dividing the value function by # users. Since OIM aims to maximize positive opinions, the total opinion per user is computed by dividing the negative value function by # users. Figure 6.2(a) and Figure 6.3(a) show that our method has around 2.5× improvement over CE and 4× improvement over FD. CE assumes the policy is sampled from a Gaussian distribution, and FD approximates the gradient. However, our method does not have such restrictions or approximations, and it exactly minimizes the cost.

Importance of the SDE formulation. Our framework significantly outperforms VI, since our SDE reformulation of user activity models preserves the *continuous time* property and our policy exactly optimizes the objective function using the

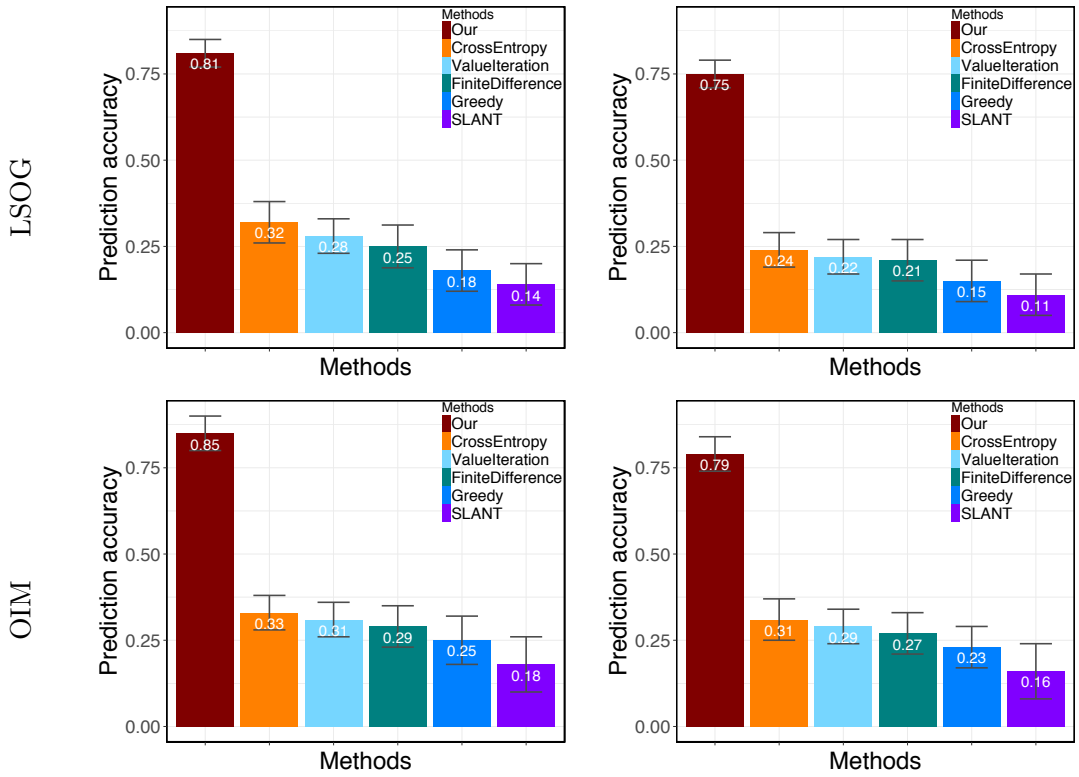


(a) Cost/opinion, Meme (b) Cost/opinion, Twitter

Figure 6.5: Results in LSOG and OIM over real-world networks with node birth processes. (a,b) total cost (for LSOG) and opinion (for OIM) in two datasets.

HJB equation. In contrast, discretizing the original opinion model introduces approximation errors, which further influences the policies in VI. Hence it is important to directly reformulating these user activity models into SDEs and study the *continuous time* control problem.

Instantaneous cost & trajectory. Figure 6.2(b) and Figure 6.3(b) shows the instantaneous cost per user over time. Our method has the fastest convergence rate to the optimal cost and the cost is much smaller than competitors. Moreover, our method has the smallest variance and is stable despite multiple runs and the stochasticity in the SDE. Figure 6.2(c,d) and Figure 6.3(c,d) compare opinion trajectories. The jumps in the opinion trajectories correspond to the opinion posting events that are modulated by the Hawkes process. For LSOG, the opinion converges to the target the fastest. For OIM, our method maximizes the opinion from the negative initial

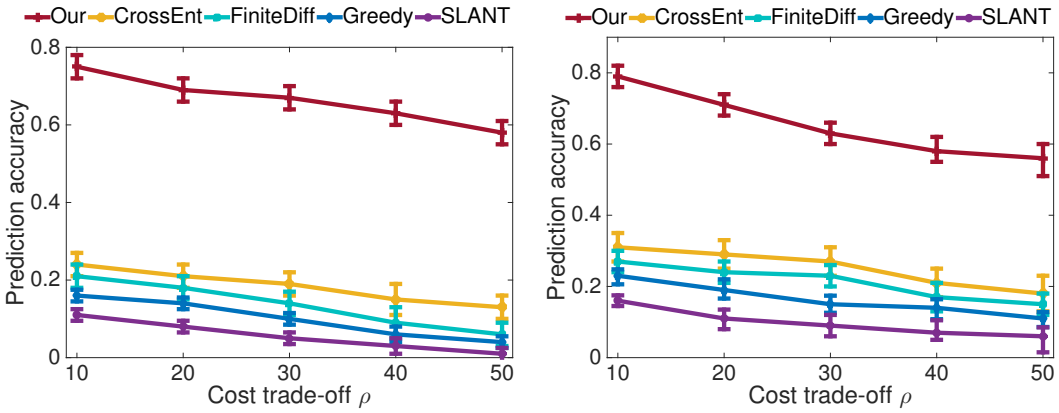


(a) Prediction, Meme (b) Prediction, Twitter

Figure 6.6: Prediction results in LSOG and OIM over real-world networks with node birth processes.

value quickly: around time 2.5, all users' opinion are positive. Moreover, our method achieves the largest opinion value, *e.g.*, 20, while that of CrossEntropy is smaller than 10.

Robustness. Since error exists between estimated parameters and ground truth, we investigate how our method performs with this discrepancy. We generate data with 10 and 100 events per user, and learn parameters by maximum likelihood estimation [131]. Figure 6.4(a,c) show that learned parameters are close to ground truth as the training data increases. Moreover, even with less accurate parameters, our cost and trajectories are close to ground-truth, while CE has high variance.



(a) Prediction in LSOG task (b) Prediction in OIM task

Figure 6.7: Budget sensitivity analysis: prediction accuracy as a function of ρ on Twitter.

6.5.2 Experiments on Real-world Data

We study the least square opinion guiding problem over two node birth networks. *Twitter* [4] contains nearly 550,000 tweet, retweet and link creation events from around 280,000 users. We use events from Sep. 21-30, 2012 and use the data before Sep. 21 to construct the initial social network. We consider the links created in the second 10-day period to be the node birth. *MemeTracker* [132] contains online social media activities from August 2008 to April 2009. Users track the posts of others and the network growth is captured by hyperlinks of comments on one site to others. We extract 11,321,362 posts among 5000 nodes. We use the data in Aug. 2008 to construct the initial network and use the LIWC [133] toolbox to extract opinions from posts. We learn parameters of the opinion dynamics and the link creation process by maximizing the likelihood [131].

We use two evaluation procedures. First, we have a real network and learned parameters; hence we simulate user behaviors, control simulated behaviors, and evaluate the total cost of different policies. The second and more interesting evaluation scheme would entail carrying real policy in a social platform. Since it is very challenging to evaluate on a real platform, we mimic such procedure using held-out data. The key

idea is to predict which real trajectory reaches the objective better (has lower cost), by comparing it to the optimal trajectory $\boldsymbol{x}^*$. Different methods yield different $\boldsymbol{x}^*$, and the prediction accuracy depends on how optimal $\boldsymbol{x}^*$ is. If it is optimal, it is accurate if we use it to order the real trajectories, and the predicted list should be similar to the ground truth, which is close to the accuracy of 1.

Total cost. Figure 6.5(a,b) show that our method performs the best for the two time-varying networks. Compared with CrossEntropy, it achieves around 6× improvement on LSOG and 3× on OIM. This result suggests that controlling the SDE over time-varying networks is a challenging problem for traditional stochastic optimal control algorithms. Moreover, the total costs of all methods for *Twitter* are higher than that of *Memetracker*. This is because *Twitter* has a much higher frequency of node birth, *i.e.*, users join the network in the timescale of minute-to-minute rather than day-to-day in *Memetracker*. Hence it is more challenging to control due to the high stochasticity in the network.

Prediction accuracy & budget sensitivity. Figure 6.6(a,b) show that our method achieves more than 0.4+ improvement over CrossEntropy. It means that our method accommodates 40% more of the total realizations correctly. An accurate prediction means that if applying our control policy, we will achieve the objective better than alternatives. Figure 6.7 shows that our method performs the best as the budget level decreases. Large value of the cost tradeoff parameter ρ means small budget.

6.6 Summary

In this chapter, we have proposed a novel SDE reformulation for user activity models and presented the activity guiding problem, which builds a new bridge between the problem of guiding of user activities in "closed loop" and stochastic optimal control theory. Moreover, we have shown that it is important to incorporate the system

status information to design a feedback control policy, which will achieve a lower cost with faster speed. Our method also provides an efficient way to guide user activities over time-varying networks with link creation and node birth processes.

Besides controlling the drift part of user's behavior system represented as as SDE, another relevant and important question is to design policies to control the jump part of the system, *i.e.*, controlling users' intensity function of generating events. This problem is challenging and the techniques developed in this chapter are not applicable. In the next chapter, we propose a novel measure-theoretic view of this problem and design a generic framework that is applicable to general nonlinear SDE systems.

CHAPTER 7
VARIATIONAL POLICY FOR GUIDING POINT PROCESSES

7.1 Introduction

Nowadays, user generated event data are becoming increasingly available. Each user is typically logged in the database with the precise time-stamp of the event, together with additional context such as tag, text, image, and video. Furthermore, these data are generated in an asynchronous fashion since any user can generate an event at any time and there may not be any coordination or synchronization between two events. Among different representations of user behaviors, temporal point processes have been widely applied to model the complex dynamics of online user behaviors [1, 2, 3, 9, 29, 5, 97, 134, 113, 135].

In spite of the broad applicability of point processes, there is little work in the area of controlling these processes to influence user behaviors. In this chapter, we study the problem of designing the best intervention policy to influence the intensity function of point processes, such that user behaviors can be influenced towards a target state.

A framework for doing this is critically important. For example, government agents may want to effectively suppress the spread of terrorist propaganda, which is important for understanding the vulnerabilities of social networks and increasing their resilience to rumor and false information; online merchants may want to promote users' frequency of visiting the website to increase sales; administrators of Q&A sites such as StackOverflow design various badges to motivate users to answer questions and provide feedbacks to increase the online engagement [136]; to gain more attention, a broadcaster on Twitter may want to design a smart tweeting strategy such that his

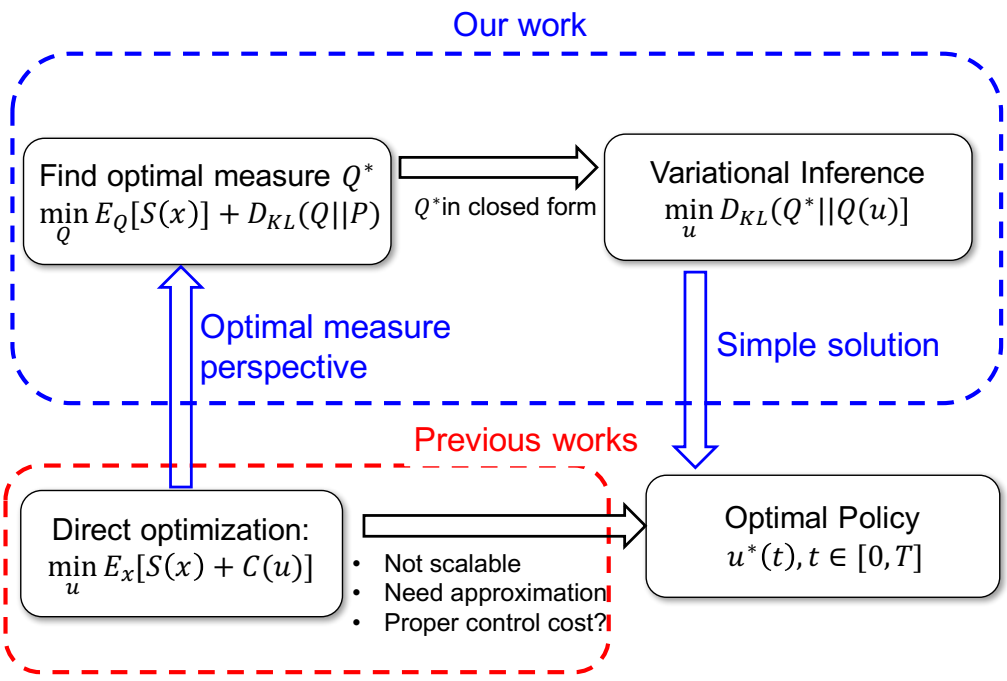


Figure 7.1: Illustration of the measure-theoretic view and benefit of our framework compared with existing approaches.

posts always remain on top of his followers' feeds [137].

Interestingly, the social science setting also introduces new challenges. Previous stochastic optimal control methods [120, 121, 122, 123] in robotics are not applicable for four reasons: (i) they mostly focus on the cases where the policy is in the drift part of the system, which is quite different from our case where the policy is on the intensity function; (ii) they require linear approximations of the nonlinear system and quadratic approximations of the objective function; (iii) to obtain a feedback control policy, these methods require the solution of the Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation, which have severe limitations in scalability and feasibility to the nonlinear systems, especially in social applications where the system's dimension is huge; (iv) the systems they study are driven by Wiener processes and Poisson processes. However, social sciences require us to consider more advanced processes, such as Hawkes processes, which are models for long term memory process and mutual exciting phenomena in social interactions.

To address these limitations, we propose an efficient framework by exploiting the novel view of measure-theoretic formulation and variational inference. Figure 7.1 illustrates our method. We make the following contributions:

- **Unified framework**. Our work offers a generic way to control nonlinear stochastic differential equations driven by point processes with stochastic intensities. Unlike prior works [122], no approximations of the system or the objective function are needed.
- **Natural control cost**. Our framework provides a meaningful control cost function to optimize: it arises naturally from the structure of the stochastic dynamics. This property is in stark contrast with the stochastic dynamic programming methods in control theory, where the control cost is imposed beforehand, despite the form of the dynamics.
- **Superior performance**. We propose a scalable model predictive control algorithm. The control policy is computed with forward sampling; hence it is scalable with parallel sampling and runs in real time. Moreover, it enjoys superior empirical performance on diverse social applications.

Related work. We first review relevant works in the machine learning community. Some works focus on controlling the point process itself, but they are not generalizable for two reasons: (i) the processes are simple, such as Poisson process [106] and a power-law decaying function [138]; (ii) the systems only contain point process. However, in social sciences, the system can be driven by many other stochastic processes. Based on Hawkes process, [35] designed its baseline intensity to achieve a steady state behavior. However, this policy does not incorporate system feedback. Recently, [139] proposed to control a user's posting intensity, which is driven by a homogeneous Poisson process. The intensity of this user's competitors is driven by Hawkes processes, and the SDE system has linear coefficients. This method computes the optimal policy by solving a HJB PDE.

In the area of stochastic optimal control, a relevant line of research focuses on event triggered control [140, 141, 142, 143]. But the problem is different: their system is linear and only contains a diffusion process, with the control affine in drift and updated at event time. The event times are driven by a fixed point process. However, we study jump diffusion SDEs and directly control the intensity that drives the time of event. Hence our work is unique among previous works.

7.2 Intensity Stochastic Control Problem

In this section, we first define the control policy and the controlled stochastic processes; then formulate the stochastic intensity control problem.

Definition 16 (Controlled Stochastic Processes)**.** *Set $\lambda_i(t)$ as the original (uncontrolled) intensity for $N_i(t)$, $u_i(t) > 0$ as the control policy, and $\tilde{\lambda}_i(u_i(t), t)$ as the controlled intensity of controlled point process $\tilde{N}_i(u_i(t), t)$. Let the uncontrolled SDE be*

$$\mathrm{d}x_i = f(x_i)\mathrm{d}t + g(x_i)\mathrm{d}w_i + \sum\nolimits_{j=1}^{M} h(x_j)\mathrm{d}N_j(t) \tag{7.1}$$

Then this uncontrolled SDE is modified as the controlled SDE:

$$\mathrm{d}x_i = f(x_i)\mathrm{d}t + g(x_i)\mathrm{d}w_i + \sum\nolimits_{j=1}^{M} h(x_j)\mathrm{d}\tilde{N}_j(u_j, t) \tag{7.2}$$

For each user i, the form of control policy is:

$$\tilde{\lambda}_i(u_i(t), t) = \lambda_i(t)u_i(t),\ i = 1, \cdots, M \tag{7.3}$$

The control policy $u_i(t)$ helps each user i decide the scale of changes to his original intensity $\lambda_i(t)$ at time t, and controls the frequency of generating events. The larger $u_i(t)$, the more likely an event will happen. Moreover, the control policy is in the multiplicative form. The rationale behind this choice is that it makes the policy easy

to execute and meaningful in practice. For example, a network moderator may request a user to reduce his tweeting intensity five times if he spreads rumors, or double the original intensity if he posts educational topics. Alternative policy formulations that are based on addition are less intuitive and not easy to execute in practice. For example, if the moderator asks the user to decrease his posting intensity by one, this instruction is difficult to be interpreted in a meaningful way. Finally, since intensity functions are positive, we set $u_i(t) > 0$.

Our goal is to find the best control policy such that this controlled SDE achieves a target state. Next, we formulate the stochastic intensity control problem.

Definition 17 (Intensity Control Problem)**.** *Given the controlled SDE in (7.2), the goal is to find* $\boldsymbol{u}^*(t)$ *for* $t \in [0, T]$*, such that the following objective function is minimized:*

$$\boldsymbol{u}^* = \operatorname{argmin}_{\boldsymbol{u}>0} \mathbb{E}_{\boldsymbol{x}}\Big[S(\boldsymbol{x}) + \gamma C(\boldsymbol{u})\Big], \tag{7.4}$$

where $\boldsymbol{x} := \{\boldsymbol{x}(t) | t \in [0, T]\}$ *is the controlled SDE trajectory on* $[0, T]$*,* $\boldsymbol{u}$ *denotes the policy on* $[0, T]$*, The expectation* $\mathbb{E}_{\boldsymbol{x}}$ *is taken over all trajectories of* $\boldsymbol{x}$*, whose stochasticity comes from the Wiener process* $\boldsymbol{w}(t)$ *and controlled point process* $\tilde{\boldsymbol{N}}(\boldsymbol{u}, t)$ *on* $[0, T]$*. The function* $C(\boldsymbol{u})$ *is the control cost, and* $S(\boldsymbol{x})$ *is the state cost defined as follows:*

$$S(\boldsymbol{x}) = \phi(\boldsymbol{x}(T), T) + \int_0^{T^-} q(\boldsymbol{x}(t), t)\mathrm{d}t \tag{7.5}$$

It is a function of the trajectory $\boldsymbol{x}$ *and measures its cost on* $[0, T]$*.* $q(\boldsymbol{x}(t), t)$ *is the instantaneous state cost at time* t*, and* $\phi(\boldsymbol{x}(T), T)$ *is the terminal state cost. The scalar* γ *controls the trade-off between state cost and control cost.*

The state cost is a user-defined function and its form depends on different applications. We will provide detailed examples in section 7.5 later. The control cost captures the budget and effort, such as time and money, to control the system.

7.3 Solution Overview

Directly computing the optimal policy in (7.4) is difficult using previous control methods [121, 122, 123]. The challenges are as follows.

Challenges. The first two challenges lie in different problem scopes. First, the control policy in these works is in the drift of SDE, and not directly applicable to the intensity control problem. Second, these works typically consider simple Poisson processes with deterministic intensity. However, in our problem the intensity can also be stochastic, which adds another layer of stochasticity. Besides the problem scopes, these works have two fundamental technical challenges:

I. Choice of control cost. These works need to define the form of control cost beforehand, which is nontrivial. For example, $u_i(t) = 1$ means there is no control. However, it is not clear which of the two heuristic forms works better:

$$\int_0^T \|\boldsymbol{u}(t) - \mathbf{1}\|^2, \int_0^T \sum_i (u_i(t) - 1) - \log\big(u_i(t)\big)\mathrm{d}t \tag{7.6}$$

Unfortunately, prior works need tedious and heuristic tuning of the function forms of control cost $C(\boldsymbol{u})$.

II. Scalability and approximations. Prior works rely on the Bellman optimality condition and use stochastic programming to derive the corresponding Hamilton-Jacobi-Bellman (HJB) partial differential equation (PDE). Solving this PDE for multi-dimensional nonlinear SDEs is difficult due to scalability limitations, *i.e.*, *curse of dimensionality* [123]. This is especially challenging in social network applications where the SDE has thousands or millions of dimensions (each user represents one dimension). Efficient solution for the PDE only exists in the special case of *linear* SDE and *quadratic* control cost and state cost. This case is restrictive when the underlying model is a nonlinear SDE, and the state cost is arbitrary function.

Our approach. To address the above challenges, we propose a generic framework with the following key steps.

I. Optimal measure-theoretic formulation. We establish a novel view of the intensity control problem by linking it to the optimal probability measure. The key insight is to compute the optimal measure $\mathbb{Q}^*$, which is induced by optimal policy $\boldsymbol{u}^*$. With this view, the control cost comes naturally as a KL-divergence term (Section 7.4.1):

$$\boxed{\mathbb{Q}^* = \text{argmin}_{\mathbb{Q}} \left[\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] + \gamma \mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P}) \right]}$$

II. Variational inference for the optimal policy. It is much easier to find the optimal measure $\mathbb{Q}^*$ compared with directly solving (7.4). Based on its form, we then parameterize $\mathbb{Q}(\boldsymbol{u})$, and compute $\boldsymbol{u}^*$ by minimizing the distance between $\mathbb{Q}^*$ and $\mathbb{Q}(\boldsymbol{u})$. This approach leads to a scalable and simple algorithm, and does not need any approximations to the nonlinear SDE or cost functions (Section 7.4.2):

$$\boxed{\boldsymbol{u}^* = \text{argmin}_{\boldsymbol{u}>0} \, \mathbb{D}_{KL}\big(\mathbb{Q}^*||\mathbb{Q}(\boldsymbol{u})\big)}$$

Finally, we transform the open-loop policy to the feedback policy and develop a scalable algorithm.

7.4 A Variational Policy Framework

In this section, we will present technical details of our framework, Variational Policy. We first provide a measure-theoretic view of the control problem, and show that finding optimal measure is equivalent to finding the optimal control. Then we compute the optimal measure and find the optimal control policy from the view of variational inference.

7.4.1 Optimal Measure-theoretic Formulation of Intensity Optimal Control Problem

Each trajectory (sample path) of a SDE is stochastic. Hence we can define a probability measure on all possible trajectories, and a SDE uniquely induces a probability measure. At a conceptual level, the SDE and the measure induced by the SDE are equivalent mathematical representations: obtaining a trajectory from this SDE by simulation (forward propagating the SDE) is equivalent to generating a sample from the probability measure induced by the SDE.

Next, we link this probability measure view to the intensity control problem. The problem in (7.4) aims at finding an optimal policy, which uniquely determines the optimal controlled SDE. Since the SDE induces a measure, (7.4) is equivalent to the problem of finding the optimal measure.

Mathematically, we set $\mathbb{P}$ as the probability measure induced by the uncontrolled SDE in (7.1), and set $\mathbb{Q}$ as the measure induced by the controlled SDE in (7.2). Hence $\mathbb{E}_{\boldsymbol{x}} = \mathbb{E}_{\mathbb{Q}}$, *i.e.*, taking the expectation over stochastic trajectories $\boldsymbol{x}$ in the original objective function is essentially taking expectation over the measure $\mathbb{Q}$. Moreover, the difference between $\mathbb{P}$ and $\mathbb{Q}$ is just the effect of the control policy. Therefore, $\boldsymbol{u}^*$ uniquely induces $\mathbb{Q}^*$. Figure 7.2 demonstrates $\mathbb{P}$ and $\mathbb{Q}$.

Based on this idea, instead of directly computing $\boldsymbol{u}^*$, we aim at finding the optimal measure $\mathbb{Q}^*$, such that $\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})]$ is minimized. We set the constraint such that $\mathbb{Q}$ is as close to $\mathbb{P}$ as possible, and propose the following objective function:

$$\min_{\mathbb{Q}} \Big[\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] + \gamma \mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P})\Big], \text{ s.t. } \int \mathrm{d}\mathbb{Q} = 1 \tag{7.7}$$

where $\int \mathrm{d}\mathbb{Q} = 1$ ensures $\mathbb{Q}$ is a probability measure, and $\mathrm{d}\mathbb{Q}$ is the probability density. $\mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}}[\log(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}})]$ is the KL divergence between these two measures.

Natural control cost. This KL divergence term provides an elegant way of measuring the distance between controlled and uncontrolled SDEs. Minimizing this

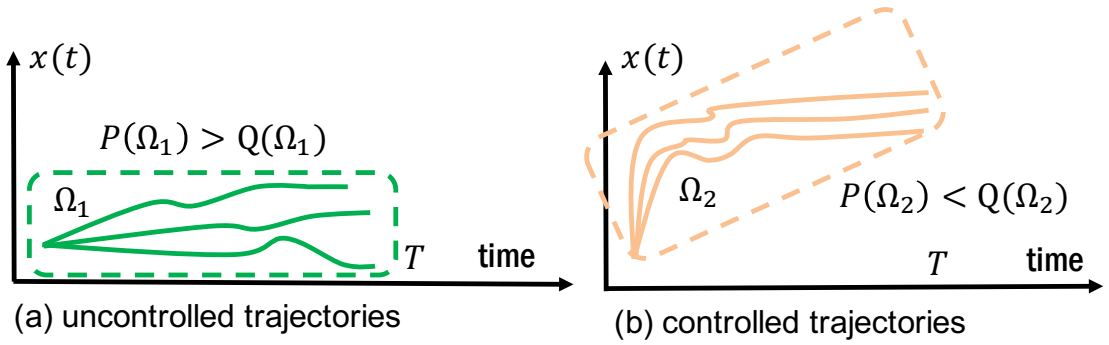


Figure 7.2: Explanation of the measures induced by SDEs. (a) the three green uncontrolled trajectories are in the region of Ω_1. Since $\mathbb{P}$ is induced by the uncontrolled SDE, naturally it has high probability on the region Ω_1 compared with $\mathbb{Q}$. Similarly, the three yellow trajectories are in Ω_2, and $\mathbb{Q}$ has high probability in this region since $\mathbb{Q}$ is induced by the controlled SDE.

term sets $\mathbb{Q}$ to be close to $\mathbb{P}$; hence it provides an implicit measure of the control cost. Mathematically, we express it as follows:

$$\begin{aligned} \mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P}) &:= \mathbb{E}_{\mathbb{Q}}[\log(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}})] \\ &= \mathbb{E}_{\mathbb{Q}}\Big[\int_0^T \sum_i \Big(\log(u_i(t)) + \frac{1}{u_i(t)} - 1\Big)\lambda_i(t)u_i(t)\mathrm{d}t\Big] \end{aligned} \tag{7.8}$$

Appendix D.4 contains derivations. With this formulation, we set the control cost $C(\boldsymbol{u}) = \log(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}})$. This function reaches its minimum when $u_i(t) = 1$, since the function $f(x) = (\log(x) + \frac{1}{x} - 1)x$ reaches the minimum when $x = 1$. Interestingly, $C(\boldsymbol{u})$ is none of the heuristics in (7.6). Hence our control cost comes naturally from the dynamics.

Another benefit of our formulation is that the probability measure that minimizes (7.7) is easy to derive (Appendix D.1 contains derivations). The optimal measure is

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} = \frac{\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]} \tag{7.9}$$

The term $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}$ is called the Radon-Nikodym derivative [144, 145]. This expression is intuitive: if a trajectory $\boldsymbol{x}$ has low state cost, then $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}$ is large. This means that

this trajectory is likely to be sampled from $\mathbb{Q}^*$. In summary, our first contribution is the link between the problem of finding optimal control to that of finding optimal measure. Computing the optimal measure is much easier than directly solving (7.4).

However, the main challenge in our measure-theoretic formulation is that there is no *explicit* transformation between the optimal measure $\mathbb{Q}^*$ and the optimal control $\boldsymbol{u}^*$. To solve this problem, next we design a convex objective function by matching probability measures.

7.4.2 Finding Optimal Policy with Variational Inference

We formulate our objective function based on the optimal measure. More specifically, we find a control $\boldsymbol{u}$ which pushes the induced measure $\mathbb{Q}(\boldsymbol{u})$, as close to the optimal measure as possible. Mathematically, we have:

$$\boldsymbol{u}^* = \text{argmin}_{\boldsymbol{u}>0}\, \mathbb{D}_{KL}\big(\mathbb{Q}^* || \mathbb{Q}(\boldsymbol{u})\big) \tag{7.10}$$

From the view of variational inference [146, 147], our objective function describes the amount of information loss when $\mathbb{Q}(\boldsymbol{u})$ is used to approximate $\mathbb{Q}^*$. This objective is in sharp contrast to traditional methods that solve the problem by computing the solution of the HJB PDE, which have severe limitations in scalability and feasibility to nonlinear SDEs [122, 123].

Next, we simplify the objective in (7.10) and compute the optimal control policy. From the definition of KL divergence and chain rule of derivatives, (7.10) is expressed as:

$$\mathbb{D}_{KL}(\mathbb{Q}^* || \mathbb{Q}(\boldsymbol{u})) = \mathbb{E}_{\mathbb{Q}^*}\Big[\log\big(\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\boldsymbol{u})}\big)\Big]. \tag{7.11}$$

The derivative $\mathrm{d}\mathbb{Q}^*/\mathrm{d}\mathbb{P}$ is given in (7.9), and we only need to compute $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$. This derivative is the relative density of probability distribution $\mathbb{P}$ w.r.t. $\mathbb{Q}(\boldsymbol{u})$. The change of probability measure happens because the intensity is changed from $\boldsymbol{\lambda}(t)$ to

$\tilde{\boldsymbol{\lambda}}(\boldsymbol{u}, t)$. Hence $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$ is essentially the *likelihood ratio* between the uncontrolled and controlled point process. We summarize its form in Theorem 18.

Theorem 18. *For the intensity control problem, we have:* $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u}) = \exp\big(\mathcal{D}(\boldsymbol{u})\big)$, *where* $\mathcal{D}(\boldsymbol{u})$ *is expressed as:*

$$\sum\nolimits_{i=1}^{M} \int_0^T \big(u_i(s) - 1\big)\lambda_i(s)\mathrm{d}s - \int_0^T \log\big(u_i(s)\big)\mathrm{d}N_i(s)$$

Appendix D.2 contains details of the proof. Next we substitute $\mathrm{d}\mathbb{Q}^*/\mathrm{d}\mathbb{P}$ and $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$ to (7.11). After removing terms independent of $\boldsymbol{u}$, the objective function is simplified as:

$$\boldsymbol{u}^* = \mathrm{argmin}_{\boldsymbol{u}>0}\, \mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\boldsymbol{u})]$$

Next, we will solve this optimization problem to compute $\boldsymbol{u}^*$. As in traditional stochastic optimal control works [122, 123], a control policy is obtained by solving the HJB PDE at discrete timestamps on $[0, T]$. Hence it suffices to parameterize our policy $\boldsymbol{u}(t)$ as a piecewise constant function on $[0, T]$.

We denote the k-th piece of $\boldsymbol{u}$ as $\boldsymbol{u}^k$, which is defined on $[k\Delta t, (k+1)\Delta t)$, with $k = 0, \cdots, K-1$, $t_k = k\Delta t$ and $T = t_K$. Now we express the objective function as follows.

$$\mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\boldsymbol{u})] = \sum\nolimits_i \sum\nolimits_k \Big(\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}} (u_i^k - 1)\lambda_i(s)\mathrm{d}s\big] - \mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}} \log(u_i^k)\mathrm{d}N_i(s)\big]\Big) \tag{7.12}$$

where u_i^k denotes the i-th dimension of $\boldsymbol{u}^k$. We just need to focus on the parts that involves u_i^k and move it outside of the expectation. Further we can show the final expression is convex in u_i^k. Finally, setting the gradient to zero yields the following

optimal control policy, denoted as u_i^{k*}:

$$u_i^{k*} = \frac{\mathbb{E}_{\mathbb{P}}\big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\int_{t_k}^{t_{k+1}} \mathrm{d}N_i(s)\big]}{\mathbb{E}_{\mathbb{P}}\big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\int_{t_k}^{t_{k+1}} \lambda_i(s)\mathrm{d}s\big]} \tag{7.13}$$

Appendix D.3 contains complete derivations. Note we have transformed $\mathbb{E}_{\mathbb{Q}^*}$ to $\mathbb{E}_{\mathbb{P}}$ using (7.9). It is important because $\mathbb{E}_{\mathbb{Q}^*}$ is not directly computable. Inspired by the idea of importance sampling, since we only know the SDE of the uncontrolled dynamics in (7.1) and can only compute the expectation under $\mathbb{P}$, the change of expectation is necessary.

To compute $\mathbb{E}_{\mathbb{P}}$, we use the Monte Carlo method to sample I trajectories from (7.1) on $[0, T]$ and take the sample average. To obtain the m-th sample $\boldsymbol{x}^m$, we use the classic routine: sample point process $\boldsymbol{N}^m(t)$ (*e.g.*, Hawkes process) using thinning algorithm [59], sample Wiener process $\boldsymbol{w}^m(t)$ from Gaussion distribution, and apply the Euler method [123] to obtain $\boldsymbol{x}^m$. Since each sample is independent, it can be scaled up easily with parallelization.

Next, we compute $w^m = \exp(-S(\boldsymbol{x}^m)/\gamma)$ by evaluating the state cost, and compute $\int_{t_k}^{t_{k+1}} \mathrm{d}N_i^m(s)$ as the number of events that occurred during $[t_k, t_{k+1})$ at the i-th dimension. Moreover, since $\lambda_i^m(t)$ is history-dependent, given the events history in the m-th sample, $\lambda_i^m(t)$ is fixed with a parametric form. Hence $\int_{t_k}^{t_{k+1}} \lambda_i^m(s)\mathrm{d}s$ can also be computed numerically or in closed form. The closed form expression exists for the Hawkes process. In summary, the sample average approximation of (7.13) is:

$$u_i^{k*} = \frac{\sum_{m=1}^{I} w^m \int_{t_k}^{t_{k+1}} \mathrm{d}N_i^m(s)}{\sum_{m=1}^{I} w^m \int_{t_k}^{t_{k+1}} \lambda_i^m(s)\mathrm{d}s} \tag{7.14}$$

Next, we discuss the properties of our policy.

Stochastic intensity. The intensity function $\lambda_i(t)$ is history independent and stochastic, *e.g.*, Hawkes process. Since $\lambda_i(t)$ is inside the expectation $\mathbb{E}_{\mathbb{P}}$ in (7.13), our policy naturally considers its stochasticity by taking the expectation.

Algorithm 8 KL - Model Predictive Control

Input: sample size I, optimization window length $\tilde{T}$, total time window T, timestamps $\{t_k\}$ on $[0, T]$.
Output: optimal control $\boldsymbol{u}^*$ at each t_k on $[0, T]$.
for $k = 0$ **to** $K - 1$ **do**
  for $m = 1$ **to** I **do**
    Sample $\mathrm{d}\boldsymbol{N}(t)$, $\mathrm{d}\boldsymbol{w}(t)$ and generate $\boldsymbol{x}^m$ on $[t_k, t_k + \tilde{T}]$ according to (7.1) and the current state.
    $S(\boldsymbol{x}^m) = \int_0^T q(\boldsymbol{x}^m)\mathrm{d}t + \phi(\boldsymbol{x}^m)$, $w^m = \exp(-\frac{1}{\gamma}S)$
  end for
  Compute u_i^{k*} from (7.14) for each i, and execute $\boldsymbol{u}^{k*}$, receive state feedback and update state.
end for

General SDE & arbitrary cost. Since we only need the SDE system to sample trajectories, our framework is applicable to general nonlinear SDEs and arbitrary cost functions.

7.4.3 From Open-loop Policy to Feedback Policy

The current control policy in (7.14) does not depend on the system's feedback. However, a more effective policy should consider the current state of SDE, and integrate such feedback into the policy. In this section, we will transform the open-loop policy into a feedback policy.

To design this feedback policy, we use the model predictive control (MPC) scheme, where the *Model* of the process is used to *Predict* the future evolution of the process to optimize the *Control* [148]. In MPC, online optimization and execution are interleaved as follows.

(i) Optimization. At time t, we compute the control policy $\boldsymbol{u}^*$ on $[t, t+\tilde{T}]$ using (7.14) for a short time horizon $\tilde{T} \ll T$ in the future. Therefore, we only need to sample trajectories on $[t, t + \tilde{T}]$ for computation instead of $[0, T]$.

(ii) Execution. We apply the first optimal move $\boldsymbol{u}^*(t)$ at this time t, and observe the new system state.

(iii) Feedback & re-optimization. At time $t+1$, with the new observed state, we re-compute the control and repeat the above process. Algorithm 8 summarizes the procedure.

The advantage of MPC is that it yields a *feedback* control that implicitly depends on the current state $\boldsymbol{x}(t)$. Moreover, separating the optimization horizon $\tilde{T}$ from T is also advantageous since it makes little sense to consider choosing a deterministic set of actions far out into the future.

7.5 Applications

In this section, we apply our framework to two real-world applications in social sciences.

Guiding opinion diffusion. The continuous-time opinion model considers the opinion and timing of each posting event [127, 29]. It assigns each user i a Hawkes intensity $\lambda_i(t)$ and an opinion process $x_i(t) \in \mathbb{R}$ where $x_i(t) = 0$ corresponds to neutral opinion. Users are connected according to a network adjacency matrix $\boldsymbol{A} = (\alpha_{ij})$. The opinion change of user is captured by three terms:

$$\mathrm{d}x_i(t) = \big(b_i - x_i\big)\mathrm{d}t + \beta \mathrm{d}w_i(t) + \sum\nolimits_j \alpha_{ij} x_j \mathrm{d}N_j(t) \tag{7.15}$$

where b_i is the baseline opinion, *i.e.*, personal characteristics. The noise process $\mathrm{d}w_i(t)$ captures the normal fluctuations in the dynamics due to unobserved factors such as activity outside the social platform and unexpected events. The jump term captures the fact that the change of user i's opinion is a weighted summation of his neighbors' influence, and α_{ij} ensures only the opinion of a user's neighbor is considered.

How to control users' posting intensity, such that the opinion dynamics is steered towards a target? We can modify each user's opinion posting process $N_j(t)$ as $\tilde{N}_j(u_j, t)$ with policy $u_j(t)$. Common choices of state costs are as follows:

- *Least square opinion shaping.* The goal is to make the expected opinion to achieve the target $\boldsymbol{a}$, *e.g.*, nobody believes the rumor during the period. Mathematically, we set $q = \|\boldsymbol{x}(t) - \boldsymbol{a}\|^2$ and $\phi = \|\boldsymbol{x}(T) - \boldsymbol{a}\|^2$.
- *Opinion influence maximization.* The goal is to maximize each user's positive opinion, *e.g.*, a political party maximizes the support during the election period. Mathematically, we set $q = -\sum_i x_i(t)$ and $\phi = -\sum_i x_i(T)$.

Guiding broadcasting behavior. When a user posts in social network, he competes with others that his followers follow, and he will gain greater attention if his posts remain top among followers' feeds. His position defined as the rank of his post among his followers. [137] models the change of a broadcaster's position due to the posting behavior of other competitors and himself as follows.

$$\mathrm{d}x_j(t) = \mathrm{d}N_o(t) - \big(x_j(t) - 1\big)\mathrm{d}N_i(t) \tag{7.16}$$

where i is the broadcaster and $j \in \mathcal{F}(i)$ denote one follower of i. The stochastic process $x_j(t) \in \mathbb{N}$ denotes the rank of broadcaster i's posts among all the posts that his follower j receives. Rank $x_j = 1$ means i's posts is the top-1 among all posts j receives. $N_i(t)$ is a Poisson process capturing the broadcaster's posting behavior. $N_o(t)$ is the Hawkes process for the behavior of all other broadcasters that j follows.

How to change the posting intensity of the broadcaster, such that his posts always remain on top? We use the policy to change $N_i(t)$ to $\tilde{N}_i(u_i, t)$ and help user i decide when to post messages. The state cost minimizes his rank among all followers' news feed. Specifically, we set the state and terminal cost as $q = \sum_{j\in\mathcal{F}(i)} x_j(t)$ and $\phi = \sum_{j\in\mathcal{F}(i)} x_j(T)$.

7.6 Experiments

We focus on two applications in the previous section: least square opinion guiding and smart broadcasting. We compare with suitable stochastic optimization approaches that are popular in reinforcement learning and heuristics.

- Cross Entropy (CE) [129]: It samples controls from a Gaussian distribution, sorts the samples in ascending order with respect to the cost and recomputes the distribution parameters based on the first K elite samples. Then it returns to the first step with a new distribution until the cost converges.
- Finite Difference (FD) [130]: It generates I samples of perturbed policies $\boldsymbol{u} + \Delta\boldsymbol{u}$ and computes perturbed cost $S + \Delta S$. Then it uses them to approximate the true gradient of the cost with respect to the policy.
- Greedy: It controls the system when local state cost is high. We divide the window into n state cost observation timestamps. At each timestamp, Greedy computes state cost and controls the system based on pre-specified control rules if current cost is more than k times of the optimal cost of our algorithm. It will stop if it has reached the current budget bound. We vary k from 1 to 5, n from 1 to 100 and report the best performance.
- Base Intensity (BI) [35]: It sets the policy for the base parameterization of the intensity only at initial time and does not consider the system feedback.

We provide both MPC and open-loop (OL) versions for our KL algorithm, Finite Difference and Cross Entropy. For MPC, we set the optimization window $\tilde{T} = T/10$ and sample size $I = 10,000$. It is efficient to generate these samples and takes less than one second using parallelization.

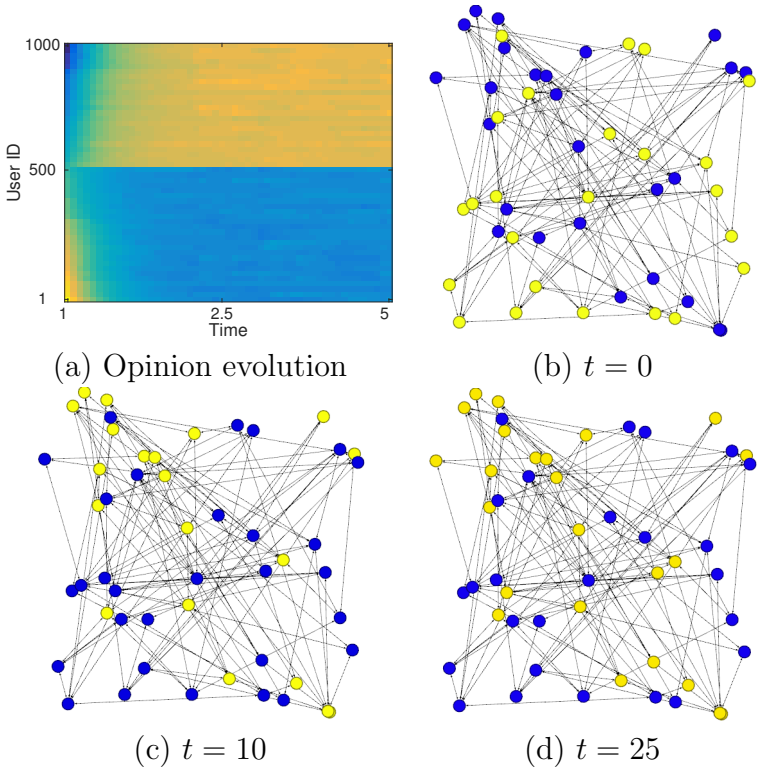
7.6.1 Experiments on Opinion Guiding

We generate a synthetic network with 1000 users. We simulate the opinion SDE on window $[0, 50]$ by applying Euler forward method [149] to compute the difference form of the SDE in (7.1). The time window is divided into 500 timestamps. We set the initial opinion $x_i(0) = -10$ and the target opinion $a_i = 1$ for each user. For model parameters, we set $\beta = 0.2$, and adjacency matrix $\boldsymbol{A}$ generated uniformly on $[0, 0.01]$ with sparsity of 0.001. We simulate the Hawkes process using thethinning algorithm [59]. We set the base intensity in the Hawkes process to be $\boldsymbol{\mu} = \mathbf{0.01}$; the influence matrix is the same as the adjacency matrix $\boldsymbol{A}$. We set the cost tradeoff parameter to be $\gamma = 10$.

Figure 7.3 shows the controlled opinion at different times. Our method works efficiently with fast convergence speed. Figure 7.4(a) shows the instantaneous cost $\|\boldsymbol{x}(t) - \boldsymbol{a}\|$ at each time t. The opinion system is gradually steered towards the target, and the cost decreases over time. Our KL-MPC achieves the lowest instantaneous cost at each time and has the fastest convergence to the optimal cost. Hence the total state cost is also the lowest.

Figure 7.4(b) shows that KL-MPC has 3× cost improvement than CE-MPC, with less variance and faster convergence. This is because KL-MPC is more flexible and has less restrictions on the control policy. CE-MPC is a popular method for the traditional control problem in robotics, where the SDE does not contain the jump term and control is in the drift. However, CE-MPC assumes the control is sampled from a Gaussian distribution, which might not be the ideal assumption in the intensity control problem. FD performs worse than CE due to the error in the gradient estimation process. Finally, for the same method, the MPC always performs better than open-loop version, which shows the importance of incorporating state feedback to the policy.

Figure 7.5(a,b) compare the controlled intensity with the uncontrolled intensity



(a) Opinion evolution (b) $t = 0$

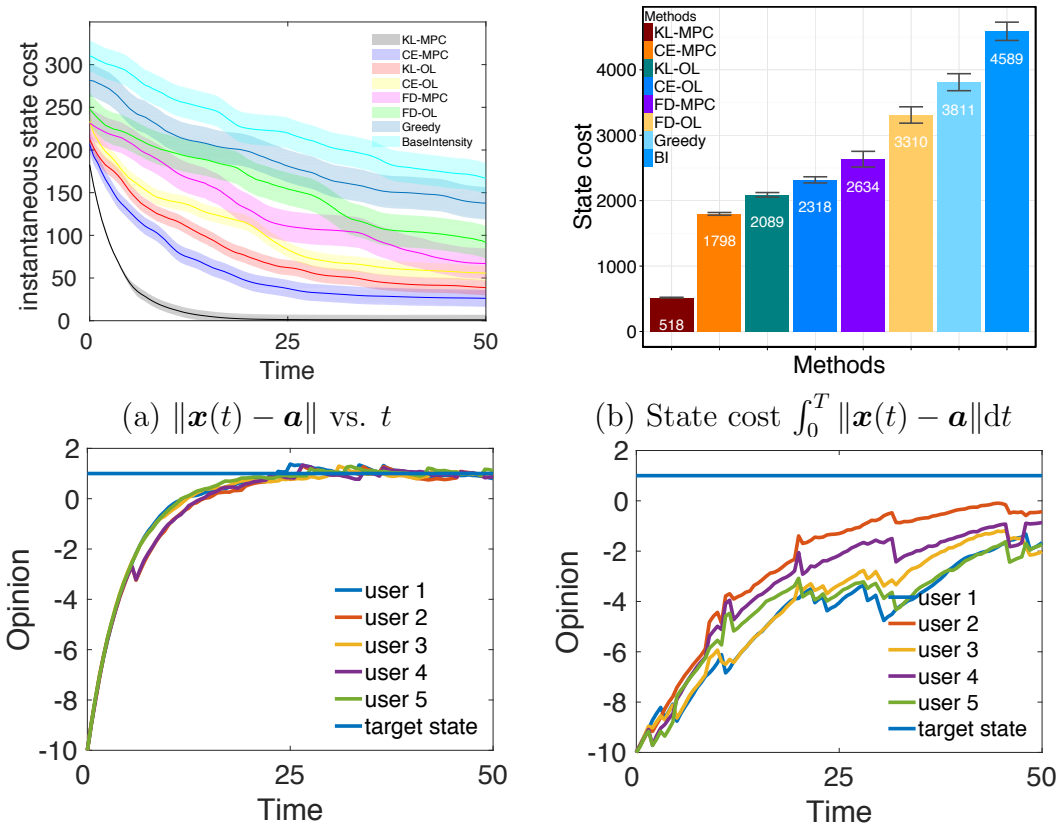
(c) $t = 10$ (d) $t = 25$

Figure 7.3: Controlled opinion dynamics of 1000 users. The initial opinions are uniformly sampled from $[-10, 10]$ and sorted, target opinion $\boldsymbol{a}$ is polarized with -5 and 10. (a) shows the opinion value per user over time. (b-d) are network snapshots of the opinion polarity of 50 sub-users. Yellow/blue means positive/negative.

at the beginning. Since the goal is to influence everyone to be positive, (a) shows that if the user tweets positive opinion, the control will increase its intensity to influence others positively. On the contrary, (b) shows that if the user's opinion is negative, his intensity will be controlled to be small. (c) and (d) show scenarios near the terminal time. Since the system is around the target state, the policy is small and the original and controlled intensity are similar for positive and negative users.

7.6.2 Experiments on Smart Broadcasting

We evaluate on a real-world Twitter dataset [4], which contains 280,000 users with 550,000 tweets/retweets. We first learn the parameters of the point processes that capture each user's posting behavior by maximizing the likelihood function of data [137]. For each broadcaster, we track down all followers and record all the tweets they posted



(a) $\|\boldsymbol{x}(t) - \boldsymbol{a}\|$ vs. t

(b) State cost $\int_0^T \|\boldsymbol{x}(t) - \boldsymbol{a}\| \mathrm{d}t$

(c) KL-MPC trajectory

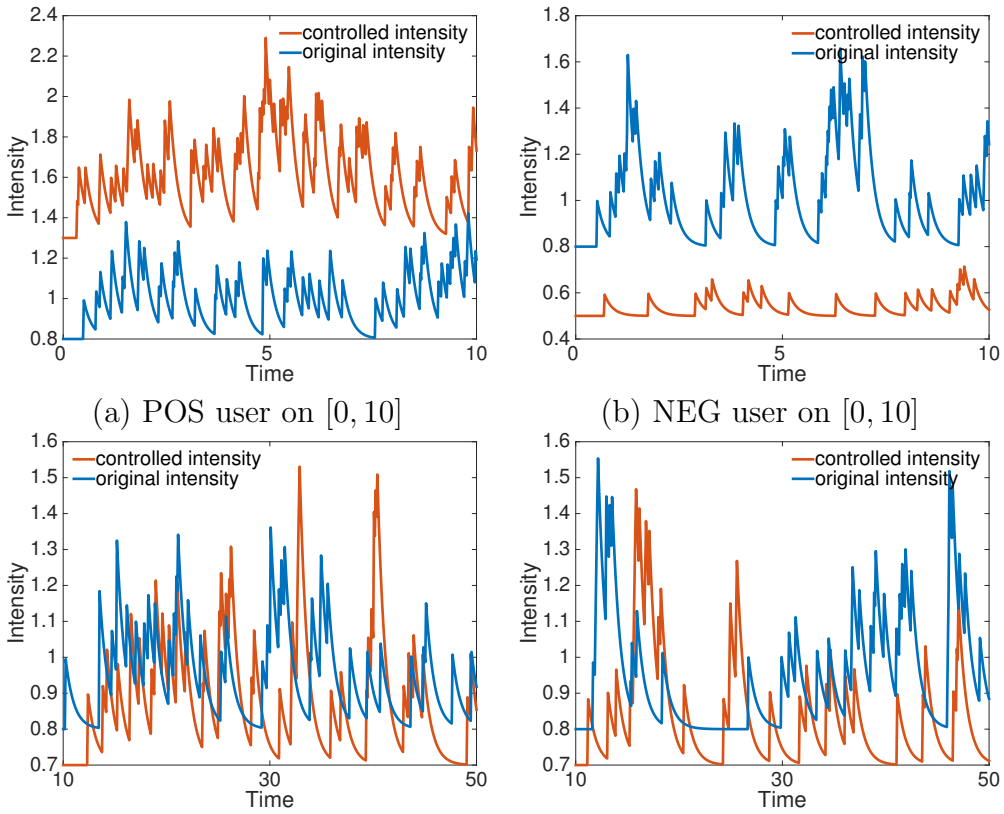
(d) CE-MPC trajectory

Figure 7.4: Experiments on least square guiding. (a) Instantaneous cost vs. time. Line is the mean and pale region is the variance; (b) state cost; (c,d) sample opinion trajectories of five users.

and reconstruct followers' timelines by collecting all the tweets by people they follow.

We use two evaluation schemes. First, similar to the synthetic case, with learned parameters, we simulate posting events on $[0, 10]$ and conduct control over the simulated dynamics with the cost tradeoff parameter as $\gamma = 10$. The time window is divided into ten timestamps. We repeat this simulation procedure ten times.

The second and more interesting scheme is to carry the policy in a real platform. Since it is very challenging to do so, we mimic it using held-out data. We partition the data into ten intervals and use one interval for training and others for testing. Each method essentially predicts which interval has smaller cost, by measuring the optimal position computed from that method to real position. Specifically, for each



(a) POS user on $[0, 10]$ (b) NEG user on $[0, 10]$

(c) POS user on $[10, 50]$ (d) NEG user on $[10, 50]$

Figure 7.5: Intensity comparison. (a-d) Opinion guiding experiment: visualization for users with positive (POS) and negative (NEG) opinions during different periods. (e) Smart broadcasting: visualization for one randomly picked broadcaster and his competitors.

broadcaster, the procedure is as follows: (i) Estimate model parameters using data in interval 1. (ii) Compute the optimal policy and obtain the broadcaster's optimal position x_i^* in each other interval i. Then sort intervals according to $|x_i - x_i^*|$. (iii) Sort intervals according to the actual value of x_i. (iv) Compute *prediction accuracy* by dividing the number of pairs with consistent ordering in step 2 and step 3 by total number of pairs. We report the accuracy over ten runs by choosing each different interval for training once.

Figure 7.7(a) compares the average rank of the broadcaster of different methods. We compute the average rank by dividing the state cost by window length, and average over all broadcasters. Kl-Mpc achieves the lowest average rank and is 4×

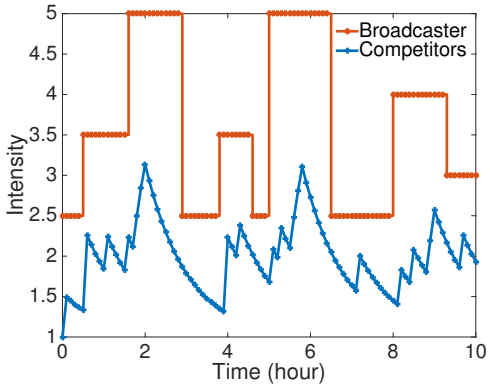
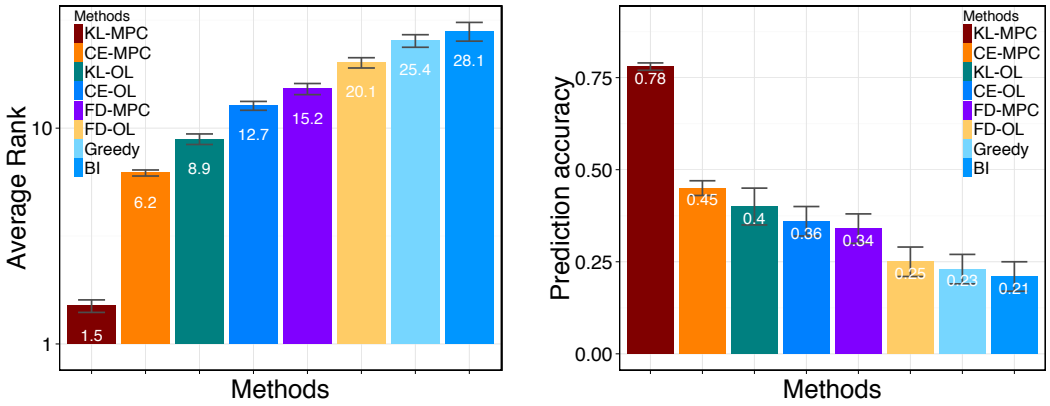


Figure 7.6: Smart broadcasting: visualization for one randomly picked broadcaster and his competitors.



(a) State cost (average rank) (b) prediction accuracy

Figure 7.7: Real world experiment with two evaluation schemes.

lower than the Ce-Mpc. Specifically, it achieves the rank around 1.5 at each time, which is nearly the ideal scenario where the broadcaster always remains on top-1. Figure 7.7(b) further shows that our method performs the best: it achieves more than 0.3+ improvement over Ce-Mpc; hence our method has 30% more of the total realizations correctly. Accurate prediction means that if applying our control policy to the users, we will achieve the objective much better than alternative methods.

Figure 7.6 compares the controlled intensity of one broadcaster with the uncontrolled intensity of his competitors. It shows that Kl-Mpc increases his intensity when that of other competitors is large, and decreases his intensity when competitor's intensity is small. For example, around timestamp 2 and 4, competitors have

large intensities; hence to remain on top, this broadcaster needs to double his intensity to create more posts. Moreover, on $[6.5, 8]$, others are not active and this broadcaster keeps a low intensity. His behavior is adaptive since our control cost ensures the broadcaster not to deviate too much from his original intensity.

7.7 Summary

In this chapter, we have presented a generic framework to control the stochastic intensity function of a general point process, such that a nonlinear SDE driven by the point process is steered towards a target state. We exploit the measure-theoretic view of the stochastic intensity control problem, derive an analytical form of the optimal measure, and compute the optimal policy using a KL divergence objective. We provide a scalable algorithm with superior performance in diverse social problems. There are many interesting venues for future work. For example, we can apply our method to other interesting problems, such as influence and activity maximization [35, 111].

CHAPTER 8
CONCLUSIONS

With the goal to make the digital platforms more useful and engaging for online users and entire society, this dissertation provides a generic framework that consists of expressive models for users' temporal behaviors, scalable predictive algorithms for users' macroscopic activity levels, and efficient optimal control policies for guiding users' behaviors. Our framework models the asynchronous and interdependent event data, which often require much more complex models and learning algorithms far beyond the existing epoch based methods. The design of our framework adheres to the following three steps: (1) we first develop expressive models and learning algorithms to capture various patterns of user's behaviors and their features; (2) based on the learned models, we next develop scalable algorithms for large scale inferences on users' behaviors, (3) finally, we consider the "closed loop" setting, and design feedback control policies to guide users' behaviors by utilizing the learned model as well as the predicted behaviors in the future. Specifically, we make the following contributions:

Model:

- We propose a novel Isotonic-Hawkes process model, which can better capture users' various behaviors in service platforms. We further develop a theoretically-guaranteed learning algorithm to learn both the link function and the parameters of our model.

- We develop a co-evolutionary feature embedding framework to capture the evolution of different entities' feature in continuous-time evolving graphs.

- We design a generic feature embedding framework that learns a general rep-

resentation of users and items, which innovatively connects recurrent neural networks with point processes.

Prediction:

- We develop an efficient time-sensitive recommendation algorithm, which is able to (1) recommend the proper item to the user at a particular time, and (2) predict the next returning of a user to the service.
- We design a novel framework to compute an unbiased estimator of the probability mass function of general point processes. It requires significantly less number of samples compared with the state of the art Monte Carlo method.

Control:

- We design a generic framework to reformulate point process based user activity models into stochastic differential equations, which allows us to bring in and extend tools from stochastic optimal control theory to address the closed loop activity guiding problem.
- We develop a novel measure theoretic view of the intensity control problem and design a scalable model predictive control algorithm, which enjoys superior performance in diverse social applications.

We believe these contributions can open several interesting research directions towards understand users' temporal and strategic behaviors. My long-term research aim is to develop rigorous foundations for the design of service platforms by using the large-scale temporal behavior data to advance our understanding of users' behavior, and then apply this knowledge to improve the various service platforms that now support our world. Having recently obtained the massive data and the computational ability to process it, we are clearly only at the start of this road. In the following, I will describe several future directions that I am passionate to pursue.

Developing a computational understanding of users' objectives. My work [8, 9] focus on designing policies to achieve an objective, but sometimes we are facing the inverse problem: "How to learn the objective of a user from observed temporal strategic behaviors?". This is an important problem in recommender systems, healthcare, and economics. In recommender systems, users make decisions on the recommended items, and receives utilities after the choice. We can treat users as online learning algorithms, parameterize the utility by a deep neural network, and design a parameter learning mechanism that links the observed choices with his utility. Similarly, in health data analytics, it is valuable to accurately understand doctor's true objectives and find the optimal treatments.

Brining automation in economic decision making. One of the most important problem in economic decision making is to measure causation. To make a better policy in a complex economy, it is vital to understand the mechanism that drives the human behavior. I have developed models and learning algorithms to understand advertisers' bidding behaviors in sponsored search auctions, and I am excited to apply this knowledge to other applications, such as measuring the effectiveness of advertising strategies, and designing of incentives for desirable sales and education outcomes.

Running online experiments. Online experimentation is an ever-increasing part of the machine learning, computational social science, and economics toolkit. I plan to continue collaborating with industry partners to run experiments that are both scientifically illuminating and practically impactful. There are many intriguing and hard problems to deal with when running randomized experiments online, and I also plan to contribute methodological tools and practices in working with online participants to help advance the modern practice of scientific experimentation.

Appendices

APPENDIX A
PROOF OF THEOREMS IN CHAPTER 2

A.1 Proof of Theorem 6

Theorem 6. *Suppose* $\mathbb{E}[N_i|\mathcal{H}_{t_i}] = \int_0^{t_i} g^*(w^* \cdot x_t)dt$, *where* g^* *is monotonic increasing, 1-Lipschitz and* $\|w^*\| \leq W$. *Then with probability at least* $1-\delta$, *there exist some iteration* $k < O\Big(\big(\frac{Wn}{\log(Wn/\delta)}\big)^{1/3}\Big)$ *such that*

$$\varepsilon(\hat{g}^k, \hat{w}^k) \leq O\bigg(\Big(\frac{W^2 \log(Wn/\delta)}{n}\Big)^{1/3}\bigg).$$

Notations. We define some extra notations. First we rewrite the integral as $\int_0^{t_i} g^*(w^* \cdot x_t)dt = \sum_{j \in \mathcal{S}_i} a_{ij} g^*(w^* \cdot x_j)$. Set $y_i^* = g^*(w^* \cdot x_i)$ to be the expected value of each y_i. Let $\bar{N}_i$ be the expected value of N_i. Then we have $\bar{N}_i = \sum_{j \in \mathcal{S}_i} a_{ij} y_j^*$. Clearly we do not have access to $\bar{N}_i$. However, consider a hypothetical call to the algorithm with input $\{(x_i, \bar{N}_i)\}_{i=1}^n$ and suppose it returns $\bar{g}^k$. In this case, we define $\bar{y}_i^k = \bar{g}^k(\bar{w}^k \cdot x_i)$. Next we begin the proof and introduce Lemma 3-5.

Analysis roadmap. To prove Theorem 6, we establish several lemmas. The heart of the proof is Lemma 3, in which we show a property of the learned parameters $\hat{w}^k$ at iteration k. That is, the squared distance $\|\hat{w}^k - w^*\|^2$ between $\hat{w}^k$ and the true direction w^* decreases at each iteration at a rate which depends on $\varepsilon(\hat{g}^k, \hat{w}^k)$ and some other additive error terms η_1 and η_2, which can be bounded respectively:

$$\|\hat{w}^k - w^*\|^2 - \|\hat{w}^{k+1} - w^*\|^2 \geq C_2 \varepsilon(\hat{g}^k, \hat{w}^k) - C_1(\eta_1 + \eta_2) \tag{A.1}$$

Lemma 4 bounds $\eta_1 = O\Big((K + \sqrt{4K^2 + 8k^2})\big(\log(\frac{1}{\delta})\big)^{1/2}\Big)$ using martingale concentration inequality.

Lemma 5 bounds $\eta_2 = O\Big(\big(\frac{W^2\log(Wn/\delta)}{n}\big)^{1/3}\Big)$. It relates $\hat{y}_j^k$ (the value we can actually compute) and $\bar{y}_j^k$ (the value we could compute if we had $\bar{N}_i$). $\bar{y}_j^k$ and $\hat{y}_j^k$ will show up when we decouple $\|\hat{w}^k - w^*\|^2 - \|\hat{w}^{k+1} - w^*\|^2$.

Finally, we plug in the values of η_1 and η_2 to Lemma 3. Then we conduct telescoping sum of (A.1) and show there is at most $O\Big(W/(\eta_1+\eta_2)\Big)$ iterations before the error $\varepsilon(\hat{g}^k, \hat{w}^k)$ is less than $O(\eta_1+\eta_2)$. Since η_2 is the dominant term compared with η_1, we replace η_1 by η_2 in the final results. This completes the proof.

Now we introduce Lemma 3-5 as follows.

Lemma 3. *Suppose that* $\|w^k - w\| \le W$, $\|x_i\| \le 1$, $\sqrt{c} \le \sum_{j\in\mathcal{S}_i} a_{ij} \le \sqrt{C}, \forall i \in [n], j \in [n]$ *and* $y_j \le M, \forall j \in [n]$, *and*

$$|\frac{1}{n}\sum_{i=1}^{n}(N_i - \bar{N}_i)| \le \eta_1, \;\; \frac{1}{n}\sum_{i=1}^{n}\sum_{j\in\mathcal{S}_i} a_{ij}|\hat{y}_j^k - \bar{y}_j^k| \le \eta_2$$

then the following formula holds:

$$\|\hat{w}^k - w^*\|^2 - \|\hat{w}^{k+1} - w^*\|^2 \ge C_2\varepsilon(\hat{g}^k, \hat{w}^k) - C_1(\eta_1+\eta_2) \tag{A.2}$$

where $C_1 = \max\{5CW, 4M\sqrt{c} + 2CW\}$, $C_2 = 2c - C$.

The complete proof of Lemma 3 is in Appendix C.

Lemma 4 (Martingale Concentration Inequality)**.** *Suppose* $dM(t) \le K$, $V(t) \le k$ *for all* $t > 0$ *and some* $K, k \ge 0$. *With probability at least* $1-\delta$, *it holds that*

$$\frac{1}{n}\sum_{i=1}^{n}|N_i - \bar{N}_i| \le O\left((K + \sqrt{4K^2+8k^2})\big(\log(1/\delta)\big)^{1/2}\right).$$

Note $N_i - \bar{N}_i = M_i$, which is the martingale at time t_i. A continuous martingale is a stochastic process such that $\mathbb{E}[M_t|\{M_\tau, \tau \le s\}] = M_s$. It means the conditional expectation of an observation at time t is equal to the observation at time s, given

all the observations up to time $s \leq t$. $V(t)$ is the variation process. It is shown in [54] that $V(t) = \Lambda(t) = \int_0^t \lambda(s)ds$, which is the compensator for point process $N(t)$. The martingale serves as the noise term in point processes (similar to Gaussian noise in regression) and can be bounded using the Bernstein-type concentration inequality. The proof is in Appendix 4.

Lemma 5. *With probability at least* $1 - \delta$*, it holds for any* k *that*

$$\frac{1}{n}\sum_{j=1}^{n} |\hat{y}_j^k - \bar{y}_j^k| \leq O\bigg(\Big(\frac{W^2 \log(Wn/\delta)}{n}\Big)^{1/3}\bigg).$$

Lemma 5 relates $\hat{y}_j^k$ (the value we can actually compute) to $\bar{y}_j^k$ (the value we could compute if we had the conditional means of N_j). The proof of this lemma uses the covering number technique and can be found in [51].

Proof of Theorem 6. With Lemma 3, we can conduct telescoping sum. There can be two cases: either $\varepsilon(\hat{g}^k, \hat{w}^k) \leq 3C_1(\eta_1 + \eta_2)/C_2$ or $\varepsilon(\hat{g}^k, \hat{w}^k) \geq 3C_1(\eta_1 + \eta_2)/C_2$. If it is the first case, then we are done. If it is the second case, then we have:

$$\|w^k - w\|^2 - \|w^{k+1} - w\|^2 \geq C_1(\eta_1 + \eta_2)$$

Since $\|w^{k+1} - w\|^2 \geq 0$, and $\|w^0 - w\|^2 \leq 2W^2$, by telescoping sum, at iteration K, we have:

$$2W^2 \geq \|w^0 - w\|^2 - \|w^K - w\|^2 \geq KC_1(\eta_1 + \eta_2)$$

Set $K = 2W^2/C_1(\eta_1 + \eta_2)$, if $k > K$, then the above inequality does not hold, which means $\varepsilon(\hat{g}^k, \hat{w}^k) \geq 3C_1(\eta_1 + \eta_2)/C_2$ does not hold. Hence there can be at most $2W^2/C_1(\eta_1 + \eta_2) = O(W/(\eta_1 + \eta_2))$ iterations before $\varepsilon(\hat{g}^k, \hat{w}^k) \leq 3C_1(\eta_1 + \eta_2)/C_2$.

The remaining step is to bound η_1 and η_2. We use Lemma 4 to bound η_1 and use Lemma 5 to bound η_2. Clearly η_2 is the dominant term. Plugging the values of η_1 and η_2, we have the conclusion that there is some h^k such that

$$\varepsilon(\hat{g}^k, \hat{w}^k) \leq O\bigg((\frac{W^2 \log(Wn/\delta)}{n})^{1/3} \bigg)$$

A.2 Proof of Lemma 7

To prove Lemma 3, a key technique is the generalized calibration property. It generalizes that of isotonic regression in [50] since our objective function is more general. We first state Lemma 7 and then provide the proof.

Lemma 7 (Generalized Calibration Property)**.** *The solutions to Quadratic Problem in (2.7) is partitioned into disjoint blocks* $\{\mathcal{P}_l\}_{l=1}^m$*, and for each block* $\mathcal{P}_l$*:*

$$\sum_{i=1}^{n} (N_i - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^k) \sum_{j \in \mathcal{P}_l} a_{ij} = 0 \tag{A.3}$$

Proof. First we define a_{ij} such that

$$a_{ij} = \begin{cases} a_{ij} & \text{if } j \in \mathcal{S}_i \\ 0 & \text{else} \end{cases}$$

Hence we have

$$\sum_{j=1}^{n} a_{ij} = \sum_{j \in \mathcal{S}_i} a_{ij} \tag{A.4}$$

We can rewrite the objective function as:

$$f = \frac{1}{2} \sum_{i=1}^{n} (N_i - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^k)^2 = \frac{1}{2} \sum_{i=1}^{n} (N_i - \sum_{j=1}^{n} a_{ij} \hat{y}_j^k)^2$$

Set $\{\lambda_i\}_{i=1}^{n-1}$ to be the Lagrange multipliers. To update $\hat{y}_j^k$, we apply the KKT conditions to (2.7) and obtain the following formulas:

$$\frac{\partial f}{\partial \hat{y}_1^k} = \sum_{i=1}^{n}(N_i - \sum_{j=1}^{n} a_{ij}\hat{y}_j^k)a_{i1} + \lambda_1 = 0 \tag{A.5}$$

$$\frac{\partial f}{\partial \hat{y}_j^k} = \sum_{i=1}^{n}(N_i - \sum_{j=1}^{n} a_{ij}\hat{y}_j^k)a_{ij} + \lambda_j - \lambda_{j-1} = 0, \;\; 2 \le j \le n-1 \tag{A.6}$$

$$\frac{\partial f}{\partial \hat{y}_n^k} = \sum_{i=1}^{n}(N_i - \sum_{j=1}^{n} a_{ij}\hat{y}_j^k)a_{in} - \lambda_{n-1} = 0 \tag{A.7}$$

$$\lambda_j(\hat{y}_j^k - \hat{y}_{j+1}^k) = 0, \quad 1 \le j \le n-1 \tag{A.8}$$

$$\lambda_j \ge 0, \quad 1 \le j \le n-1 \tag{A.9}$$

Depending whether $\hat{y}_j^k$'s are equal, we can divide the subscript of $\hat{y}_j^k$ into disjoint sets $\{\mathcal{P}_l\}_{l=1}^{m}$ such that in each $\mathcal{P}_l$, the values of $\hat{y}_j^k$ are the same. Hence there exists $j_1 < j_2 < \cdots < j_{m-1} < n$, such that

$$\mathcal{P}_1 = \{1, \cdots, j_1\}, \mathcal{P}_2 = \{j_1 + 1, \cdots, j_2\}, \cdots, \mathcal{P}_m = \{j_{m-1} + 1, n\} \tag{A.10}$$

Figure A.1 illustrates an example when $m = 3$. in this case, $\mathcal{P}_1 = \{1, 2\}$, $\mathcal{P}_2 = \{3, 4\}$, and $\mathcal{P}_3 = \{5, 6\}$. Now we show the following equality holds for $l = 1, \cdots, m$ in three cases,

$$\sum_{i=1}^{n}(N_i - \sum_{j=1}^{n} a_{ij}\hat{y}_j^k) \sum_{j \in \mathcal{P}_l} a_{ij} = 0$$

Case 1: the first block. For $\mathcal{P}_1$, we sum up equations $\frac{\partial f}{\partial \hat{y}_j^k} = 0$ according to the index in $\mathcal{P}_1$. we have

$$\begin{cases} \sum_{i=1}^{n}(N_i - \sum_{j=1}^{n} a_{ij}\hat{y}_j^k) \sum_{j \in \mathcal{P}_1} a_{ij} + \lambda_{j_1} = 0 \\ \lambda_{j_1} = 0 \end{cases} \tag{A.11}$$

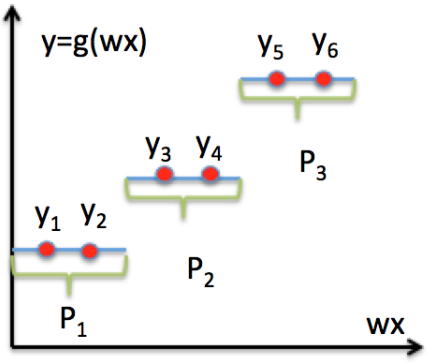


Figure A.1: Demonstration for the block partition in (A.10). $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ are the first, intermediate and last block respectively. In each block, $\hat{y}$ has the same value.

Since $\hat{y}^k_{j_1} \neq \hat{y}^k_{j_1+1}$, from (A.8) we have $\lambda_{j_1} = 0$.

Case 2: the intermediate blocks. For $2 \leq l \leq m-1$, in $\mathcal{P}_l$, we sum up equations $\frac{\partial f}{\partial \hat{y}^k_j} = 0$. Then we have

$$\begin{cases} \sum_{i=1}^n (N_i - \sum_{j=1}^n a_{ij}\hat{y}^k_j) \sum_{j\in\mathcal{P}_l} a_{ij} + \lambda_{j_l} - \lambda_{j_{l-1}} = 0 \\ \lambda_{j_l} = \lambda_{j_{l-1}} = 0 \end{cases} \tag{A.12}$$

Since $\hat{y}^k_{j_l} \neq \hat{y}^k_{j_l+1}$ and $\hat{y}^k_{j_{l-1}} \neq \hat{y}^k_{j_{l-1}+1}$, from (A.8) we have $\lambda_{j_l} = \lambda_{j_{l-1}} = 0$.

Case 3: the last block. For $\mathcal{P}_m$, similarly we have

$$\begin{cases} \sum_{i=1}^n (N_i - \sum_{j=1}^n a_{ij}\hat{y}^k_j) \sum_{j\in\mathcal{P}_m} a_{ij} - \lambda_{j_{m-1}} = 0 \\ \lambda_{j_{m-1}} = 0 \end{cases} \tag{A.13}$$

From (A.4), we have for all $l = 1, \cdots, m$

$$\sum_{i=1}^n (N_i - \sum_{j\in\mathcal{P}_l} a_{ij}\hat{y}^k_j) \sum_{j\in\mathcal{P}_l} a_{ij} = 0$$

This completes the proof.

A.3 Proof of Lemma 3

First, we have

$$\|\hat{w}^k - w^*\|^2 - \|\hat{w}^{k+1} - w^*\|^2 = 2(\hat{w}^{k+1} - \hat{w}^k) \cdot (w^* - \hat{w}^k) - \|\hat{w}^{k+1} - \hat{w}^k\|^2 \tag{A.14}$$

$$= \underbrace{\frac{2}{n}\sum_{i=1}^{n}(N_i - \sum_{j\in\mathcal{S}_i} a_{ij}\hat{y}_j^k)(\sum_{j\in\mathcal{S}_i} a_{ij}x_j \cdot (w^* - \hat{w}^k))}_{A} - \underbrace{\left\|\frac{1}{n}\sum_{i=1}^{n}(N_i - \sum_{j\in\mathcal{S}_i} a_{ij}\hat{y}_j^k)\sum_{j\in\mathcal{S}_i} a_{ij}x_j\right\|^2}_{B} \tag{A.15}$$

First we simplify A. Using the following equality:

$$N_i - \sum_{j\in\mathcal{S}_i} a_{ij}\hat{y}_j^k = N_i - \sum_{j\in\mathcal{S}_i} a_{ij}y_j^* + \sum_{j\in\mathcal{S}_i} a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k + \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k - \sum_{j\in\mathcal{S}_i} a_{ij}\hat{y}_j^k,$$

we can rewrite A into three parts:

$$A = \frac{2}{n}\sum_{i=1}^{n}(N_i - \sum_{j\in\mathcal{S}_i} a_{ij}y_j^*)(\sum_{j\in\mathcal{S}_i} a_{ij}x_j) \cdot (w^* - w^k) \tag{A.16}$$

$$+ \frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i} a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k)(\sum_{j\in\mathcal{S}_i} a_{ij}x_j \cdot (w^* - w^k)) \tag{A.17}$$

$$+ \frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k - \sum_{j\in\mathcal{S}_i} a_{ij}\hat{y}_j^k)(\sum_{j\in\mathcal{S}_i} a_{ij}x_j \cdot (w^* - w^k)) \tag{A.18}$$

The term (A.16) is at least $-2CW\eta_1$, the term (A.18) is at least $-2CW\eta_2$ since $|\sum_{j\in\mathcal{S}_i} a_{ij}(w - w^k) \cdot x_j| \leq \sqrt{C}W$ and assuming $C \geq 1$. We thus bound(A.17).

First define v, the inverse of g as

$$v(y) = \inf\{z \in \mathrm{dom}(g) | g(z) = y\}$$

Note that v is well defined since g is monotonic. We also split (A.17) into three parts,

$$\frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i}a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)(\sum_{j\in\mathcal{S}_i}a_{ij}x_j\cdot(w^* - \hat{w}^k))$$

$$= \frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i}a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i}a_{ij}v(\bar{y}_j^k) \tag{A.19}$$

$$- \frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i}a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i}a_{ij}\hat{w}^k\cdot x_j \tag{A.20}$$

$$+ \frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i}a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i}a_{ij}(w^*\cdot x_j - v(\bar{y}_j^k)) \tag{A.21}$$

As for (A.19), it is 0 by Lemma 7. To see this, remember that $\bar{N}_i = \sum_{j\in\mathcal{S}_i}a_{ij}y_i^*$ and $\bar{y}_j^k$ is the output of the algorithm in Eq. (2.7) with input $\{(\bar{w}^k\cdot x_i, \bar{N}_i)\}$. Apply Lemma 6 and we have the pools $\{\mathcal{P}_l\}_{l=1}^m$ and

$$\sum_{i=1}^{n}(\bar{N}_i - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{P}_l}\sum_{j\in\mathcal{S}_i}a_{ij} = 0$$

Define function v to be the inverse of g. v is defined as $v(y) = \inf\{z\in\mathrm{dom}(g)|g(z) = y\}$. Since g is monotonic, v is well-defined. Since all $\bar{y}_j^k$ in the same set $\mathcal{P}_l$ has the same value, then the value $v(\bar{y}_j^k)$ (the inverse mapping) is also the same. Hence

$$\sum_{i=1}^{n}(N_i - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i}a_{ij}v(\bar{y}_j^k) = 0$$

Now sum the above equation up for all sets $\mathcal{P}_l, l = 1,\cdots,m$, note that $\bigcup_{l=1}^m\mathcal{P} = \{1,\cdots,n\}$, we have

$$\sum_{i=1}^{n}(\bar{N}_i - \sum_{j\in\mathcal{S}_i}a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i}a_{ij}v(\bar{y}_j^k) = 0$$

As to (A.20), we show it is always no greater than 0. To see this, we first claim that

for any $\delta > 0$,

$$\sum_{i=1}^{n}(\bar{N}_i - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k)^2 \leq \sum_{i=1}^{n}(\bar{N}_i - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k - \delta(\sum_{j\in\mathcal{S}_i} a_{ij}x_j)\cdot\hat{w}^k)^2$$

This is because $\sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k$ minimizes the sum of squared difference w.r.t. $\bar{N}_i$ over all such sequences. Rewriting this as a difference of squares gives,

$$\sum_i \delta(\sum_{j\in\mathcal{S}_i} a_{ij}x_j)\cdot\hat{w}^k\left(2N_i - 2\sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k - \delta(\sum_{j\in\mathcal{S}_i} a_{ij}x_j)\cdot\hat{w}^k\right) \geq 0$$

Dividing both sides by $2\delta > 0$, we have

$$\sum_i (\sum_{j\in\mathcal{S}_i} a_{ij}x_j)\cdot\hat{w}^k\left(\bar{N}_i - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k - \frac{\delta}{2}(\sum_{j\in\mathcal{S}_i} a_{ij}x_j)\cdot\hat{w}^k\right) \geq 0$$

Setting $\delta \to 0$, by continuity we obtain

$$\frac{2}{n}\sum_{i=1}^{n}(\bar{N}_i - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i} a_{ij}\hat{w}^k\cdot x_j \geq 0$$

Hence we have (A.20) always no greater than 0.

As to (A.21), by 1-Lipschitz property of g, the first term can be bounded as

$$\begin{aligned}
&\frac{2}{n}\sum_{i=1}^{n}(\sum_{j\in\mathcal{S}_i} a_{ij}y_j^* - \sum_{j\in\mathcal{S}_i} a_{ij}\bar{y}_j^k)\sum_{j\in\mathcal{S}_i} a_{ij}(v(y_j^*) - v(\bar{y}_j^k)) \\
&\geq \frac{2}{n}\sum_{j=1}^{n} c(y_j^* - \bar{y}_j^k)(v(y_j^*) - v(\bar{y}_j^k)) \\
&\geq \frac{2}{n}\sum_{j=1}^{n} c(y_j^* - \bar{y}_j^k)^2 = 2c\varepsilon(\bar{g}^k, \bar{w}^k)
\end{aligned} \tag{A.22}$$

Plugging to the definition of A, we get

$$\boxed{A \geq 2c\varepsilon(\bar{g}^k, \bar{w}^k) - 2CW(\eta_1 + \eta_2)} \tag{A.23}$$

Next we bound B. First rewrite B as:

$$
\begin{aligned}
B &= \left\| \frac{1}{n} \sum_{i=1}^{n} (N_i - \sum_{j \in \mathcal{S}_i} a_{ij} y_j^* + \sum_{j \in \mathcal{S}_i} a_{ij} y_j^* - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^k) \sum_{j \in \mathcal{S}_i} a_{ij} x_j \right\|^2 \\
&\leq \left\| \frac{1}{n} \sum_{i=1}^{n} (N_i - \sum_{j \in \mathcal{S}_i} a_{ij} y_j^*) \sum_{j \in \mathcal{S}_i} a_{ij} x_j \right\|^2 \qquad \text{(A.24)} \\
&+ 2 \left\| \frac{1}{n} \sum_{i=1}^{n} (N_i - \sum_{j \in \mathcal{S}_i} a_{ij} y_j^*) \sum_{j \in \mathcal{S}_i} a_{ij} x_i \right\| \times \left\| \frac{1}{n} \sum_{i=1}^{n} (\sum_{j \in \mathcal{S}_i} a_{ij} y_j^* - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^k) \sum_{j \in \mathcal{S}_i} a_{ij} x_j \right\| \qquad \text{(A.25)} \\
&+ \left\| \frac{1}{n} \sum_{i=1}^{n} (\sum_{j \in \mathcal{S}_i} a_{ij} y_j^* - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^k) \sum_{j \in \mathcal{S}_i} a_{ij} x_j \right\|^2 \qquad \text{(A.26)}
\end{aligned}
$$

From the condition in Lemma 3, we have

$$
\| \frac{1}{n} \sum_{i=1}^{n} (N_i - \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \mathcal{S}_i} a_{ij} y_j^*) \sum_{j \in \mathcal{S}_i} a_{ij} x_j \|^2 \leq C \eta_1^2 \qquad \text{(A.27)}
$$

Use Jensen's inequality and consider the upper bound C for $\| \sum_{j \in \mathcal{S}_i} a_{ij} x_i \|^2$, we show that

$$
\left\| \frac{1}{n} \sum_{i=1}^{n} (\sum_{j \in \mathcal{S}_i} a_{ij} y_j^* - \sum_{j \in \mathcal{S}_i} a_{ij} \hat{y}_j^k) \sum_{j \in \mathcal{S}_i} a_{ij} x_i \right\|^2 \leq C \times \frac{1}{n} \sum_{i=1}^{n} (y_j^* - \hat{y}_j^k)^2 = C \varepsilon(\hat{g}^k, \hat{w}^k) \qquad \text{(A.28)}
$$

Combining (A.27) and (A.28) into (A.24), (A.25), (A.26), assuming $\eta_1 \leq 1$, $C \geq 1$, we have

$$
B \leq C \eta_1^2 + 2C \eta_1 \sqrt{\varepsilon(\hat{g}^k, \hat{w}^k)} + C \varepsilon(\hat{g}^k, \hat{w}^k) \leq C \varepsilon(\hat{g}^k, \hat{w}^k) + 3C \eta_1 \qquad \text{(A.29)}
$$

Hence the we have:

$$
\boxed{B \leq C \varepsilon(\hat{g}^k, \hat{w}^k) + 3C \eta_1} \qquad \text{(A.30)}
$$

Combining the bound for A in (A.23) and the bound for B in (A.30) into (A.15), we

get

$$\boxed{\|\hat{w}^k - \hat{w}\|^2 - \|\hat{w}^{k+1} - \hat{w}\|^2 \geq 2c\varepsilon(\bar{g}^k, \bar{w}^k) - C\varepsilon(\hat{g}^k, \hat{w}^k) - CW(5\eta_1 + 2\eta_2)} \quad \text{(A.31)}$$

To finish the proof, we establish the relationship between $\varepsilon(\bar{g}^k, \bar{w}^k)$ and $\varepsilon(\hat{g}^k, \hat{w}^k)$ as follows: we claim that the difference between $\varepsilon(\bar{g}^k, \bar{w}^k)$ and $\varepsilon(\hat{g}^k, \hat{w}^k)$ can be lower bounded:

$$\boxed{\varepsilon(\bar{g}^k, \bar{w}^k) - \varepsilon(\hat{g}^k, \hat{w}^k) \geq -2M\eta_2/\sqrt{c}} \quad \text{(A.32)}$$

To see this, we have:

$$\begin{aligned}
\varepsilon(\bar{g}^k, \bar{w}^k) &= \frac{1}{n}\sum_{j=1}^{n}(\bar{y}_j^k - y_j^*)^2 \\
&= \frac{1}{n}\sum_{j=1}^{n}(\bar{y}_j^k - \hat{y}_j^k + \hat{y}_j^k - y_j^*)^2 \\
&= \frac{1}{n}\sum_{j=1}^{n}(\hat{y}_j^k - y_j^*)^2 + \frac{1}{n}\sum_{j=1}^{n}(\bar{y}_j^k - \hat{y}_j^k)(\bar{y}_j^k + \hat{y}_j^k - 2y_j^*) \\
&= \varepsilon(\hat{g}^k, \hat{w}^k) + \frac{1}{n}\sum_{j=1}^{n}(\bar{y}_j^k - \hat{y}_j^k)(\bar{y}_j^k + \hat{y}_j^k - 2y_j^*)
\end{aligned}$$

and we have $|\bar{y}_i^k + \hat{y}_i^k - 2y_i^*| \leq 2M$. Plugging this and the following inequality leads to (A.32).

$$\frac{1}{n}\sum_{j=1}^{n}|\hat{y}_j^k - \bar{y}_j^k| \leq \frac{1}{n}\sum_{j=1}^{n}\sum_{j\in\mathcal{S}_i} a_{ij}/\sqrt{c}|\hat{y}_j^k - \bar{y}_j^k| \leq \eta_2/\sqrt{c}$$

Combine (A.32) and (A.31), we have

$$\begin{aligned}
\|w^k - w\|^2 - \|w^{k+1} - w\|^2 &\geq (2c - C)\varepsilon(\hat{g}^k, \hat{w}^k) - 4M\sqrt{c}\eta_2 - CW(5\eta_1 + 2\eta_2) \\
&\geq C_2\varepsilon(\hat{g}^k, \hat{w}^k) - C_1(\eta_1 + \eta_2)
\end{aligned}$$

where $C_1 = \max\{5CW, 4M\sqrt{c} + 2CW\}$, $C_2 = (2c - C)$, this completes the proof.

A.4 Proof of Lemma 4

We have $N_i - \bar{N}_i = M_i$, which is the martingale at time t_i. The martingale serves as the noise term in point processes (similar to Gaussian noise in regression) and can be bounded using the Bernstein-type concentration inequality. First, we have the following martingale inequality [54, 150]: for each ϵ and some t, we have

$$\mathbb{P}[|M(t)| > \epsilon] \leq \exp\bigg(- \frac{\epsilon^2}{2(k^2 + \epsilon K)} \bigg)$$

In our case, for each i, we have $N_i = \Lambda(t_i) + M(t_i)$, where $\Lambda(t)$ is the compensator and $M(t)$ is the zero-mean martingale. Also we have $\bar{N}_i = \mathbb{E}(N_i) = \Lambda(t_i)$. Hence $N_i - \bar{N}_i = M(t_i) = M_i$. Now we set $\delta = \mathbb{P}[|M(t)| > \epsilon]$, then with probability at least $1 - \delta$, $|M(t)| \leq \epsilon$. Set $\delta = \exp\bigg(- \frac{\epsilon^2}{2(k^2 + \epsilon K)} \bigg)$, then we have the equation

$$\epsilon^2 - 2K \log(\frac{1}{\delta})\epsilon - 2k^2 \log(\frac{1}{\delta}) = 0$$

Hence

$$\begin{aligned}
\epsilon &= \frac{2K \log(\frac{1}{\delta}) + \sqrt{4K^2 \big(\log(\frac{1}{\delta}) \big)^2 + 8k^2 \log(\frac{1}{\delta})}}{2} \\
&\leq K \log(\frac{1}{\delta}) + \sqrt{4K^2 + 8k^2} \bigg(\log(\frac{1}{\delta}) \bigg)^{1/2} \\
&\leq (K + \sqrt{4K^2 + 8k^2}) \bigg(\log(\frac{1}{\delta}) \bigg)^{1/2}
\end{aligned}$$

Here we have used the fact that $\big(\log(1/\delta) \big)^2 \leq \log(1/\delta) \leq \sqrt{\log(1/\delta)}$. We can obtain that

$$\epsilon = O\left((K + \sqrt{4K^2 + 8k^2}) \big(\log(\frac{1}{\delta}) \big)^{1/2} \right)$$

Hence we have

$$\frac{1}{n}\sum_{i=1}^{n}|N_i - \bar{N}_i| = \frac{1}{n}\sum_{i=1}^{n}|M_i| \leq O\left((K + \sqrt{4K^2 + 8k^2})\big(\log(\frac{1}{\delta})\big)^{1/2}\right)$$

APPENDIX B

PROOF OF THEOREMS IN CHAPTER 5

B.1 Proof of Theorem 8

Theorem 8 (Mass Transport Equation for Point Processes)**.** *Let* $\lambda(t) := \lambda(t|\mathcal{H}_{t^-})$ *be the conditional intensity function of the point process* $N(t)$ *and* $\tilde{\phi}(x,t) := \mathbb{P}[N(t) = x|\mathcal{H}_{t^-}]$ *be its conditional probability mass function; then* $\tilde{\phi}(x,t)$ *satisfies the following differential-difference equation:*

$$\tilde{\phi}_t(x,t) := \frac{\partial \tilde{\phi}(x,t)}{\partial t} = \begin{cases} -\lambda(t)\tilde{\phi}(x,t) + \lambda(t)\tilde{\phi}(x-1,t) & \text{if } x = 1,2,3,\cdots \\ -\lambda(t)\tilde{\phi}(x,t) & \text{if } x = 0 \end{cases} \tag{B.1}$$

Proof. For the simplicity of notation, we define a functional operator $\mathcal{F}[\tilde{\phi}]$ as follows:

$$\mathcal{F}[\tilde{\phi}] = -\lambda(t)\tilde{\phi}(x,t) + \lambda(t)\tilde{\phi}(x-1,t)\mathbb{I}[x \geqslant 1],$$

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

Our goal is to prove $\mathcal{F}[\tilde{\phi}] = \tilde{\phi}_t$. For the simplicity of notation, we define the inner product [151] between functions $f(x)$ and $g(x)$ as the summation of the product of of $f(x)$ and $g(x)$, where $x \in \mathbb{N}$:

$$(f,g) = \sum_{x=0}^{\infty} f(x)g(x)$$

To prove the equality $\tilde{\phi}_t = \mathcal{F}[\tilde{\phi}]$, we will prove that the equality $(v, \tilde{\phi}_t) = (v, \mathcal{F}[\tilde{\phi}])$ holds for any test function $v(x)$. Then the equality $\tilde{\phi}_t = \mathcal{F}[\tilde{\phi}]$ follows from the famous Fundamental Lemma of Calculus of Variations [107]. To show the above equality, we

start by computing $(v, \tilde{\phi}_t)$.

Computing $(v, \tilde{\phi}_t)$. According to the definition of expectation and the fact that $\tilde{\phi}(x,t)$ is the conditional probability mass, we have

$$\mathbb{E}[v(N(t))|\mathcal{H}_{t^-}] = \sum\nolimits_{x=0}^{\infty} v(x)\mathbb{P}[N(t)=x|\mathcal{H}_{t^-}] = \sum\nolimits_{x=0}^{\infty} v(x)\tilde{\phi}(x,t) = (v, \tilde{\phi}).$$

Taking the gradient with respect to t yields

$$\frac{\partial \mathbb{E}[v(N(t))|\mathcal{H}_{t^-}]}{\partial t} = \sum\nolimits_{x=0}^{\infty} v(x)\tilde{\phi}_t(x,t) = (v, \tilde{\phi}_t). \tag{B.2}$$

Next, we obtain another expression for $(v, \tilde{\phi}_t)$. First we show the following property of $\mathrm{d}v(N(t))$

$$\mathrm{d}v(N(t)) = \big(v(N(t)+1) - v(N(t))\big)\mathrm{d}N(t) \tag{B.3}$$

In fact, from the definition of the differential operator d, we have the following property:

$$\mathrm{d}v(N(t)) := v\big(N(t+\mathrm{d}t)\big) - v\big(N(t)\big) = v\big(N(t)+\mathrm{d}N(t)\big) - v\big(N(t)\big)$$

Since $\mathrm{d}N(t) = \{0,1\}$, if $\mathrm{d}N(t) = 0$, we have $\mathrm{d}v(N(t)) = 0$; otherwise, we have $\mathrm{d}v(N(t)) = v(N(t)+1) - v(N(t))$. For both cases, equation (B.3) holds.

Next, we integrate both sides of (B.3) on $[0,t]$ and express $v(N(t))$ as follows:

$$v(N(t)) = v(N(0)) + \int_0^t \big(v(N(t)+1) - v(N(t))\big)\mathrm{d}N(t) \tag{B.4}$$

Given $\mathcal{H}_{t^-}$, we take the conditional expectation of (B.4) and obtain the following expression:

$$\mathbb{E}[v(N(t))|\mathcal{H}_{t^-}] = v(N(0)) + \mathbb{E}\Big[\int_0^t \big(v(N(t)+1)) - v(N(t))\big)\lambda(t)\mathrm{d}t\Big|\mathcal{H}_{t^-}\Big] \tag{B.5}$$

Now we differentiate both sides of (B.5) with respect to time t and obtain the following expression:

$$\begin{aligned}\frac{\partial \mathbb{E}[v(N(t))|\mathcal{H}_{t^-}]}{\partial t} &= \mathbb{E}\Big[\frac{\partial}{\partial t}\int_{t_0}^{t}\Big(\mathcal{B}[v](N(s))\Big)\mathrm{d}s\Big|\mathcal{H}(t^-)\Big] \\ &= \mathbb{E}\Big[\mathcal{B}[v](N(t))\Big|\mathcal{H}_{t^-}\Big] \\ &= \sum\nolimits_{x=0}^{\infty}\mathcal{B}[v](x(t))\tilde{\phi}(x,t) \\ &= (\mathcal{B}[v],\tilde{\phi}) \end{aligned} \tag{B.6}$$

where $\mathcal{B}[v]$ is another functional operator defines as

$$\mathcal{B}[v]\big(N(t)\big) = \big(v(N(t)+1) - v(N(t))\big)\lambda(t) \tag{B.7}$$

Since (B.6) and (B.2) are equivalent, we have:

$$(v,\tilde{\phi}_t) = (\mathcal{B}[v],\tilde{\phi})$$

Now we have finished the first part of the proof. In the second part, our goal is to move the operator $\mathcal{B}$ from test function v to the conditional probability mass function ϕ and prove $(\mathcal{B}[v],\tilde{\phi}) = (v,\mathcal{F}[\tilde{\phi}])$. We start by computing $(\mathcal{B}[v],\tilde{\phi})$ as follows.

Computing $(\mathcal{B}[v],\tilde{\phi})$. We define a new post-jump variable as $y = x+1$, and conduct a *change of variable* from x to $y = x+1$ in $(\mathcal{B}[v],\tilde{\phi})$. Specifically, we express $(\mathcal{B}[v],\tilde{\phi})$ as follows

$$\begin{aligned}\sum_{x=0}^{\infty}\big(v(x+1)-v(x)\big)\lambda(t)\tilde{\phi}(x,t) &= \sum_{x=0}^{\infty}v(x+1)\lambda(t)\tilde{\phi}(x,t) - \sum_{x=0}^{\infty}v(x)\lambda(t)\tilde{\phi}(x,t) \\ &= \sum_{y=1}^{\infty}v(y)\lambda(t)\tilde{\phi}(y-1,t) - \sum_{x=0}^{\infty}v(x)\lambda(t)\tilde{\phi}(x,t)\end{aligned} \tag{B.8}$$

Next, we use an indicator function and let the value of y to start from 0 in the first term of (B.8):

$$\begin{aligned}\sum_{y=1}^{\infty} v(y)\lambda(t)\tilde{\phi}(y-1,t) &= \sum_{y=0}^{\infty} v(y)\lambda(t)\tilde{\phi}(y-1,t)\mathbb{I}[y \geqslant 1] \\ &= \Big(v(y), \lambda(t))\tilde{\phi}(y-1,t)\mathbb{I}[y \geqslant 1]\Big) \end{aligned} \tag{B.9}$$

Now we substitute (B.9) back to (B.8) and obtain the following equation:

$$\begin{aligned}\sum_{x=0}^{\infty} \big(v(x+1)-v(x)\big)\lambda(t)\tilde{\phi}(x,t) &= \Big(v(y), \lambda(t))\tilde{\phi}(y-1,t)\mathbb{I}[y \geqslant 1]\Big) - \Big(v(x), \lambda(t)\tilde{\phi}(x,t)\Big) \\ &= \Big(v(x), \lambda(t))\tilde{\phi}(x-1,t)\mathbb{I}[x \geqslant 1]\Big) - \Big(v(x), \lambda(t)\tilde{\phi}(x,t)\Big) \\ &= (v, \mathcal{F}[\tilde{\phi}]) \end{aligned} \tag{B.10}$$

Hence, for an arbitrary function $v(x)$, we have shown the following equality:

$$(v, \tilde{\phi}_t) = (\mathcal{B}[v], \tilde{\phi}) = (v, \mathcal{F}[\tilde{\phi}]).$$

This yields $\tilde{\phi}_t = \mathcal{F}[\tilde{\phi}]$ and the proof is now complete.

B.2 Proof of Theorem 7

Theorem 7. *For time $t > 0$ and an arbitrary function f, we have:*

$$\mathbb{VAR}[g(\mathcal{H}_{t^-})] < \mathbb{VAR}[f(N(t))] \tag{B.11}$$

Proof. The proof contains two steps. We first compute the expected value of the conditional variance $\mathbb{E}\Big[\mathbb{VAR}\Big[f(N(t))|\mathcal{H}_{t^-}\Big]\Big]$, and next compute the variance of the conditional expected value $\mathbb{VAR}\Big[g(\mathcal{H}_{t^-})\Big]$.

(i) Expected value of the conditional variance. Since $\mathbb{VAR}[f(N(t))|\mathcal{H}_{t^-}]$ is a ran-

dom variable, we can compute its expected value. Using the definition of variance, *i.e.*, $\mathbb{VAR}[f(N(t))|\mathcal{H}_{t^-}] = \mathbb{E}[f(N(t))^2|\mathcal{H}_{t^-}] - [\mathbb{E}[f(N(t))|\mathcal{H}_{t^-}]]^2$, we have

$$\mathbb{E}\Big[\mathbb{VAR}\Big[f(N(t))|\mathcal{H}_{t^-}\Big]\Big] = \mathbb{E}\Big[\mathbb{E}\Big[f(N(t))^2|\mathcal{H}_{t^-}\Big]\Big] - \mathbb{E}\Big[\Big[f(N(t))|\mathcal{H}_{t^-}\Big]^2\Big] \tag{B.12}$$

$$= \mathbb{E}[f(N(t))^2] - \mathbb{E}\Big[\Big[\mathbb{E}[f(N(t))|\mathcal{H}_{t^-}]\Big]^2\Big] \tag{B.13}$$

(ii) Variance of the conditional expected value. We express $\mathbb{VAR}\Big[g(\mathcal{H}_{t^-})\Big]$ as follows

$$\mathbb{VAR}\Big[g(\mathcal{H}_{t^-})\Big] = \mathbb{VAR}\Big[\mathbb{E}\Big[f(N(t))|\mathcal{H}(t)\Big]\Big] \tag{B.14}$$

$$= \mathbb{E}\Big[\mathbb{E}\Big[f(N(t))|\mathcal{H}_{t^-}\Big]^2\Big] - \Big[\mathbb{E}\Big[\mathbb{E}[f(N(t))|\mathcal{H}_{t^-}]\Big]\Big]^2 \tag{B.15}$$

$$= \mathbb{E}\Big[\mathbb{E}\Big[f(N(t))|\mathcal{H}_{t^-}\Big]^2\Big] - \mathbb{E}[f(N(t))]^2 \tag{B.16}$$

Combining (B.13) and (B.16) yields the following equation:

$$\mathbb{VAR}[g(\mathcal{H}_{t^-})] + \mathbb{E}\Big[\mathbb{VAR}\big[f(N(t))|\mathcal{H}_{t^-}\big]\Big] = \mathbb{VAR}[N(t)]$$

Next, we show that the inequality in our theorem is strict. According to the definition of counting process, we have $N(0) = 0$. Moreover, we are only interested in the scenarios where the number of events are positive, *i.e.*, $N(t) > 0$ for future time $t > 0$. Since the point process $N(t)$ is right continuous and not a predictable process [106], we obtain the fact that conditioning on $\mathcal{H}_{t^-}$, there is a stochastic jump at time t and the value of $f(N(t))$ is random and not a constant. Hence the conditional variance $\mathbb{VAR}\big[f(N(t))|\mathcal{H}_{t^-}\big]$ is positive and we have $\mathbb{E}\Big[\mathbb{VAR}\big[f(N(t))|\mathcal{H}_{t^-}\big]\Big] > 0$. The proof is now complete.

APPENDIX C

PROOF OF THEOREMS IN CHAPTER 6

C.1 Proof of Theorem 11

Theorem 11 (Transformation Framework)**.** *The equivalent SDE form of the user activity model is:*

$$\begin{aligned} \mathrm{d}\lambda_i(t) &= \mathrm{d}\eta_i + \omega_1\big(\eta_i - \lambda_i\big)\mathrm{d}t + \sum\nolimits_j \beta_{ij}\mathrm{d}N_j(t) \\ \mathrm{d}x_i(t) &= \mathrm{d}b_i + \omega_2\big(b_i - x_i(t)\big)\mathrm{d}t + \sum\nolimits_j \alpha_{ij}h(x_j)\mathrm{d}N_j(t) \end{aligned} \tag{C.1}$$

Proof. Given any two function $f(t)$ and $g(t)$, we first define a convolution operator $\star$ as follows

$$f(t) \star g(t) = \int_0^t f(t-s)g(s)\mathrm{d}s \tag{C.2}$$

Therefore, the user activity model for $x_i(t)$ can be expressed as

$$x_i(t) = \underset{\text{base}}{b_i(t)} + \underbrace{\sum\nolimits_{j=1}^{U} \alpha_{ij}\kappa_{\omega_2}(t) \star \big(h(x_j(t))\mathrm{d}N_j(t)\big)}_{\text{social neighbor influence}} \tag{C.3}$$

Before we apply the differential operator d to (C.3), we also need the following two properties:

- $\mathrm{d}\kappa_{\omega_2}(t) = -\omega_2\kappa_{\omega_2}(t)\mathrm{d}t$ for $t \geqslant 0$ and $\kappa_{\omega_2}(0) = 1$.
- The differential of the convolution of two functions is expressed as: $\mathrm{d}(f \star g) = f(0)g + g \star \mathrm{d}f$.

With the above two properties, we set $f = \kappa_{\omega_2}(t)$ and $g = \sum_j \alpha_{ij}h(x_j)\mathrm{d}N_j(t)$, and

take the differential of $x_i(t)$ in (C.3) as follows

$$\mathrm{d}x_i(t) = \mathrm{d}b_i(t) + \mathrm{d}(f \star g) \tag{C.4}$$

$$= \mathrm{d}b_i(t) + \sum\nolimits_{j=1}^{U} \alpha_{ij} h(x_j) \mathrm{d}N_j(t) - \omega_2 \Big(\sum\nolimits_{j=1}^{U} \alpha_{ij} k_{\omega_2}(t) \star (h(x_j) \cdot \mathrm{d}N_j(t)) \Big) \mathrm{d}t \tag{C.5}$$

$$= \mathrm{d}b_i(t) + \sum\nolimits_{j=1}^{U} \alpha_{ij} h(x_j) \mathrm{d}N_j(t) - \omega_2 (x_i(t) - b_i(t)) \mathrm{d}t \tag{C.6}$$

$$= \mathrm{d}b_i(t) + \omega_2 (b_i(t) - x_i(t)) \mathrm{d}t + \sum\nolimits_{j=1}^{U} \alpha_{ij} h(x_j(t)) \mathrm{d}N_j(t) \tag{C.7}$$

This completes the proof for the SDE formulation of $x_i(t)$.

Similarly, we can express the intensity function using the convolution operator as follows

$$\lambda_i(t) = \underset{\text{base}}{\eta_i(t)} + \underbrace{\sum\nolimits_{j=1}^{U} \beta_{ij} \kappa_{\omega_1}(t) \star \mathrm{d}N_j(t)}_{\text{social neighbor influence}} \tag{C.8}$$

Then we set $f = \kappa_{\omega_1}(t)$, $g = \sum_j \beta_{ij} \mathrm{d}N_j(t)$, and can show the following equation:

$$\mathrm{d}\lambda_i(t) = \mathrm{d}\eta_i(t) + \omega_1 \big(\eta_i(t) - \lambda_i(t) \big) \mathrm{d}t + \sum\nolimits_j \beta_{ij} \mathrm{d}N_j(t) \tag{C.9}$$

This completes the proof.

C.2 Proof of Theorem 13

Theorem 13 (Generalized Ito's Lemma)**.** *Given the SDE in (6.5), let $V(\boldsymbol{x}, t)$ be twice-differentiable in $\boldsymbol{x}$ and once in t; then we have:*

$$\mathrm{d}V = \Big\{ V_t + \frac{1}{2} \mathrm{tr}(V_{\boldsymbol{xx}} \boldsymbol{g}\boldsymbol{g}^\top) + V_{\boldsymbol{x}}^\top (\boldsymbol{f} + \boldsymbol{u}) \Big\} \mathrm{d}t + V_{\boldsymbol{x}}^\top \boldsymbol{g} \mathrm{d}\boldsymbol{w} + \big(V(\boldsymbol{x} + \boldsymbol{h}, t) - V(\boldsymbol{x}, t) \big) \mathrm{d}\boldsymbol{N}(t) \tag{C.10}$$

Proof. To prove the theorem, we will first provide some background and useful

formulas as follows [123].

$$(\mathrm{d}t)^2 = 0, \mathrm{d}t\mathrm{d}\boldsymbol{N}(t) = 0, \mathrm{d}t\mathrm{d}\boldsymbol{w}(t) = 0, \mathrm{d}\boldsymbol{w}(t)\mathrm{d}\boldsymbol{N}(t) = 0, \mathrm{d}\boldsymbol{w}(t)\mathrm{d}\boldsymbol{w}(t)^\top = \mathrm{d}t\boldsymbol{I} \quad \text{(C.11)}$$

All the above equations hold in the *mean square limit* sense. The mean square limit definition enables us to extend the calculus rules for deterministic functions and properly define stochastic calculus rules such as stochastic differential and stochastic integration for stochastic processes.

We first restate the SDE in (6.5) as follows

$$\begin{aligned}\mathrm{d}\boldsymbol{x} &= \big(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}\big)\mathrm{d}t + \boldsymbol{g}(\boldsymbol{x})\mathrm{d}\boldsymbol{w}(t) + \boldsymbol{h}(\boldsymbol{x})\mathrm{d}\boldsymbol{N}(t) \\ &= \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{h}(\boldsymbol{x})\mathrm{d}\boldsymbol{N}(t),\end{aligned}$$

where $\boldsymbol{F}(\boldsymbol{x})$ denotes the continuous part of the SDE and is define as

$$\boldsymbol{F}(\boldsymbol{x}) = \big(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}\big)\mathrm{d}t + \boldsymbol{g}(\boldsymbol{x})\mathrm{d}\boldsymbol{w}(t)$$

Note that the term $\boldsymbol{h}\mathrm{d}\boldsymbol{N}(t)$ denotes the discontinuous part of the SDE. For the simplicity of notation, we set $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}$ and $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}$ and omit the dependency on $\boldsymbol{x}$.

Next, we expand $\mathrm{d}V$ according to its definition as follows

$$\mathrm{d}V(\boldsymbol{x}, t) = V(\boldsymbol{x}(t + \mathrm{d}t), t + \mathrm{d}t) - V(\boldsymbol{x}, t)$$

With the definition $\boldsymbol{x}(t + \mathrm{d}t) = \boldsymbol{x}(t) + \mathrm{d}\boldsymbol{x}$, we can expand $V(\boldsymbol{x}(t + \mathrm{d}t), t + \mathrm{d}t)$ using

Taylor expansion on variable t as follows

$$V(\boldsymbol{x}(t+\mathrm{d}t), t+\mathrm{d}t) = V(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}, t+\mathrm{d}t) \tag{C.12}$$

$$= V(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}, t) + V_t(\boldsymbol{x}, t)\mathrm{d}t \tag{C.13}$$

Next, we expand $V(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}, t)$ as follows

$$V(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}, t)$$

$$= V(\boldsymbol{x}+\boldsymbol{F}+\boldsymbol{h}\mathrm{d}\boldsymbol{N}(t), t) \tag{C.14}$$

$$= \Big(V(\boldsymbol{x}+\boldsymbol{F}+\boldsymbol{h}, t) - V(\boldsymbol{x}+\boldsymbol{F}, t)\Big)\mathrm{d}\boldsymbol{N}(t) + V(\boldsymbol{x}+\boldsymbol{F}, t) \tag{C.15}$$

$$= \Bigg[\underbrace{V(\boldsymbol{x}+\boldsymbol{h}, t) + V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h})^\top \boldsymbol{F} + \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}(\boldsymbol{x}+\boldsymbol{h})\boldsymbol{F}^\top}_{\text{Taylor expansion 1}} \tag{C.16}$$

$$- \Big(\underbrace{V(\boldsymbol{x}, t) + V_{\boldsymbol{x}}^\top \boldsymbol{F} + \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}\boldsymbol{F}^\top}_{\text{Taylor expansion 2}}\Big)\Bigg]\mathrm{d}\boldsymbol{N}(t) \tag{C.17}$$

$$+ \underbrace{V(\boldsymbol{x}, t) + V_{\boldsymbol{x}}^\top \boldsymbol{F} + \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}\boldsymbol{F}^\top}_{\text{Taylor expansion 2}} \tag{C.18}$$

$$= \Big(V(\boldsymbol{x}+\boldsymbol{h}, t) - V(\boldsymbol{x}, t)\Big)\mathrm{d}\boldsymbol{N}(t) + \Big(V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h}) - V_{\boldsymbol{x}}\Big)^\top \boldsymbol{F}\mathrm{d}\boldsymbol{N}(t) \tag{C.19}$$

$$+ V(\boldsymbol{x}, t) + V_{\boldsymbol{x}}^\top \boldsymbol{F} + \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}\boldsymbol{F}^\top + \Big(\frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}(\boldsymbol{x}+\boldsymbol{h})\boldsymbol{F}^\top - \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}(\boldsymbol{x})\boldsymbol{F}^\top\Big)\mathrm{d}\boldsymbol{N}(t) \tag{C.20}$$

Next, we show the reasoning from (C.14) to (C.18).

First, we derive a stochastic calculus rule for the point process. Specifically, since $\mathrm{d}\boldsymbol{N}(t) \in \{0, 1\}$, there are two cases for (C.14): If a jump happens, *i.e.*, $\mathrm{d}\boldsymbol{N}(t) = 1$, (C.14) is equivalent to $V(\boldsymbol{x}+\boldsymbol{F}(\boldsymbol{x})+\boldsymbol{h}(\boldsymbol{x}), t)$; otherwise, we have $\mathrm{d}\boldsymbol{N}(t) = 0$ and (C.14) is equivalent to $V(\boldsymbol{x}+\boldsymbol{F}(\boldsymbol{x}), t)$. Hence (C.15) is equivalent to (C.14). This stochastic rule essentially takes $\mathrm{d}\boldsymbol{N}(t)$ from inside the value function V to the outside.

Second, from (C.15) to (C.18), we have used the following Taylor expansions.

Taylor expansion 1. For $V(\boldsymbol{x}+\boldsymbol{F}+\boldsymbol{h},t)$, we expand it around $V(\boldsymbol{x}+\boldsymbol{h},t)$ on the $\boldsymbol{x}$-dimension

$$V(\boldsymbol{x}+\boldsymbol{F}+\boldsymbol{h},t) = V(\boldsymbol{x}+\boldsymbol{h},t) + V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h})^{\top}\boldsymbol{F} + \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{x}\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h})\boldsymbol{F}^{\top}$$

Taylor expansion 2. For $V(\boldsymbol{x}+\boldsymbol{F},t)$, we expand it around $V(\boldsymbol{x},t)$ along the $\boldsymbol{x}$ dimension

$$V(\boldsymbol{x}+\boldsymbol{F},t) = V(\boldsymbol{x},t) + V_{\boldsymbol{x}}^{\top}\boldsymbol{F} + \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{x}\boldsymbol{x}}\boldsymbol{F}^{\top}$$

Next, we simplify each term in (C.19) and (C.20). We keep the first term and expand the second term, $\Big(V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h}) - V_{\boldsymbol{x}}\Big)^{\top}\boldsymbol{F}\mathrm{d}\boldsymbol{N}(t)$ as follows

$$\begin{aligned}\Big(V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h}) - V_{\boldsymbol{x}}\Big)^{\top}\boldsymbol{F}\mathrm{d}\boldsymbol{N}(t) &= \Big(V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h}) - V_{\boldsymbol{x}}\Big)^{\top}((\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t + \boldsymbol{g}\mathrm{d}\boldsymbol{w}(t))\mathrm{d}\boldsymbol{N}(t) \\ &= \Big(V_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{h}) - V_{\boldsymbol{x}}\Big)^{\top}\Big((\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t\mathrm{d}\boldsymbol{N}(t) + \boldsymbol{g}\mathrm{d}\boldsymbol{w}(t)\mathrm{d}\boldsymbol{N}(t)\Big), \\ &= 0 \qquad \text{(C.21)}\end{aligned}$$

where we have used the equations: $\mathrm{d}t\mathrm{d}\boldsymbol{N}(t) = 0$ and $\mathrm{d}\boldsymbol{w}(t)\mathrm{d}\boldsymbol{N}(t) = 0$ in the Ito mean square limit sense from (C.11).

We keep the third term and expand the fourth term $V_{\boldsymbol{x}}^{\top}\boldsymbol{F}$ as

$$V_{\boldsymbol{x}}^{\top}\boldsymbol{F} = V_{\boldsymbol{x}}^{\top}(\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t + V_{\boldsymbol{x}}^{\top}\boldsymbol{g}\mathrm{d}\boldsymbol{w}(t) \qquad \text{(C.22)}$$

The fifth term $\frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}\boldsymbol{F}^\top$ is expanded as follows

$$\begin{aligned}
&\frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}\boldsymbol{F}^\top \\
&= \frac{1}{2}\Big((\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t + \boldsymbol{g}\mathrm{d}\boldsymbol{w}(t)\Big)V_{\boldsymbol{xx}}\Big((\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t + \boldsymbol{g}\mathrm{d}\boldsymbol{w}(t)\Big)^\top \\
&= \frac{1}{2}\Big((\boldsymbol{f}+\boldsymbol{u})V_{\boldsymbol{xx}}(\boldsymbol{f}+\boldsymbol{u})^\top(\mathrm{d}t)^2 + 2(\boldsymbol{f}+\boldsymbol{u})\mathrm{d}tV_{\boldsymbol{xx}}(\boldsymbol{g}\mathrm{d}\boldsymbol{w}(t))^\top + (\boldsymbol{g}\mathrm{d}\boldsymbol{w}(t))V_{\boldsymbol{xx}}(\boldsymbol{g}\mathrm{d}\boldsymbol{w}(t))^\top\Big) \\
&= \frac{1}{2}\Big(0 + 0 + \mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top)\mathrm{d}t\Big) \\
&= \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top)\mathrm{d}t, \qquad \text{(C.23)}
\end{aligned}$$

where we have used the property that $(\mathrm{d}t)^2 = 0$, $\mathrm{d}t\mathrm{d}\boldsymbol{w} = 0$, and $\mathrm{d}\boldsymbol{w}(t)\mathrm{d}\boldsymbol{w}(t)^\top = \mathrm{d}t\boldsymbol{I}$ from (C.11).

Finally, the last term is expressed as

$$\begin{aligned}
&\Big(\frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}(\boldsymbol{x}+\boldsymbol{h})\boldsymbol{F}^\top - \frac{1}{2}\boldsymbol{F}V_{\boldsymbol{xx}}(\boldsymbol{x})\boldsymbol{F}^\top\Big)\mathrm{d}\boldsymbol{N}(t) \\
&= \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}(\boldsymbol{x}+\boldsymbol{h})\boldsymbol{g}\boldsymbol{g}^\top)\mathrm{d}t\mathrm{d}\boldsymbol{N}(t) - \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top)\mathrm{d}t\mathrm{d}\boldsymbol{N}(t) = 0 - 0 = 0 \qquad \text{(C.24)}
\end{aligned}$$

Substituting equation (C.21), (C.22), (C.23), and (C.24) to equation (C.19) and (C.20), we have:

$$\begin{aligned}
V(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}, t) =& \Big(V(\boldsymbol{x}+\boldsymbol{h}, t) - V(\boldsymbol{x}, t)\Big)\mathrm{d}\boldsymbol{N}(t) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t + V_{\boldsymbol{x}}^\top\boldsymbol{g}\mathrm{d}\boldsymbol{w}(t) \\
&+ V(\boldsymbol{x}, t) + \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top)\mathrm{d}t \qquad \text{(C.25)}
\end{aligned}$$

Plugging (C.25) to (C.13), we have:

$$\begin{aligned}
V(\boldsymbol{x}(t+\mathrm{d}t), t+\mathrm{d}t) = &\Big(V(\boldsymbol{x}+\boldsymbol{h}, t) - V(\boldsymbol{x}, t)\Big)\mathrm{d}\boldsymbol{N}(t) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u})\mathrm{d}t + V_{\boldsymbol{x}}^\top\boldsymbol{g}\mathrm{d}\boldsymbol{w}(t) \\
&+ V(\boldsymbol{x}, t) + \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top)\mathrm{d}t + V_t(\boldsymbol{x}, t)\mathrm{d}t
\end{aligned}$$

Hence after simplification, we obtain (C.10) and finishes the proof:

$$\begin{aligned}
\mathrm{d}V &= V(\boldsymbol{x}(t+\mathrm{d}t), t+\mathrm{d}t) - V(\boldsymbol{x}(t), t) \\
&= \Big\{V_t + \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u})\Big\}\mathrm{d}t + V_{\boldsymbol{x}}^\top \boldsymbol{g}\mathrm{d}\boldsymbol{w} + \big(V(\boldsymbol{x}+\boldsymbol{h}, t) - V(\boldsymbol{x}, t)\big)\mathrm{d}\boldsymbol{N}(t)
\end{aligned}$$

C.3 Proof of Theorem 14

Theorem 14. *The HJB equation for the user activity guiding problem in (6.7) is*

$$\begin{aligned}
-V_t = \min_{\boldsymbol{u}} \Big[\mathcal{L} + \frac{1}{2}\mathrm{tr}\big(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top\big) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u}) \\
+ \sum\nolimits_{j=1}^{U} \lambda_j(t)\big(V(\boldsymbol{x}+\boldsymbol{h}_j(\boldsymbol{x}), t) - V(\boldsymbol{x}, t)\big)\Big]
\end{aligned}$$

where $\boldsymbol{h}_j(\boldsymbol{x})$ is the j-th column of $\boldsymbol{h}(\boldsymbol{x})$.

Proof. First we express the value function V as follows

$$V(\boldsymbol{x}, t) = \min_{\boldsymbol{u}} \mathbb{E}\Big[V(\boldsymbol{x}(t+\mathrm{d}t), t+\mathrm{d}t) + \int_t^{t+\mathrm{d}t} \mathcal{L}\ \mathrm{d}\tau\Big] \tag{C.26}$$

$$= \min_{\boldsymbol{u}} \mathbb{E}\Big[V(\boldsymbol{x}, t) + \mathrm{d}V + \mathcal{L}\ \mathrm{d}t\Big] \tag{C.27}$$

$$\begin{aligned}
= \min_{\boldsymbol{u}} \mathbb{E}\Big[V(\boldsymbol{x}, t) + \Big\{V_t + \frac{1}{2}\mathrm{tr}(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u})\Big\}\mathrm{d}t \\
+ V_{\boldsymbol{x}}^\top \boldsymbol{g}\mathrm{d}\boldsymbol{w} + \big(V(\boldsymbol{x}+\boldsymbol{h}, t) - V(\boldsymbol{x}, t)\big)\mathrm{d}\boldsymbol{N}(t) + \mathcal{L}\ \mathrm{d}t\Big]
\end{aligned} \tag{C.28}$$

$$\begin{aligned}
= \min_{\boldsymbol{u}} \Big[V(\boldsymbol{x}, t) + \Big\{V_t + \mathcal{L} + \frac{1}{2}\mathrm{tr}\big(V_{\boldsymbol{xx}}\boldsymbol{g}\boldsymbol{g}^\top\big) + V_{\boldsymbol{x}}^\top(\boldsymbol{f}+\boldsymbol{u})\Big\}\mathrm{d}t \\
+ \sum\nolimits_{j=1}^{U} \lambda_j(t)\big(V(\boldsymbol{x}+\boldsymbol{h}_j(\boldsymbol{x}), t) - V(\boldsymbol{x}, t)\big)\mathrm{d}t,\Big]
\end{aligned} \tag{C.29}$$

where (C.27) to (C.28) follows from Theorem 13, and (C.28) to (C.29) follows from the properties of Wiener processes and point processes, *i.e.*, $\mathbb{E}[\mathrm{d}\boldsymbol{w}] = 0$ and $\mathbb{E}[\mathrm{d}\boldsymbol{N}(t)] = \boldsymbol{\lambda}(t)\mathrm{d}t$.

Finally, cancelling $V(\boldsymbol{x}, t)$ on both sides of (C.29) and dividing both sides by $\mathrm{d}t$ yields the HJB equation.

C.4 Proof of Proposition 15

For the quadratic cost case (the opinion least square guiding problem), we have: $\phi = \frac{1}{2}\|\boldsymbol{x}(T) - \boldsymbol{a}\|^2$, $\mathcal{L} = \frac{1}{2}\|\boldsymbol{x}(t) - \boldsymbol{a}\|^2 + \frac{\rho}{2}\|\boldsymbol{u}(t)\|^2$. Since the instantaneous cost $\mathcal{L}$ is quadratic in $\boldsymbol{x}$ and $\boldsymbol{u}$, and terminal cost ϕ is quadratic in $\boldsymbol{x}$, if the control $\boldsymbol{u}$ is a linear function of $\boldsymbol{x}$, then the value function V must be quadratic in $\boldsymbol{x}$, since it is the optimal value of the summation of quadratic functions.

Moreover, the fact that $\boldsymbol{u}$ is linear in $\boldsymbol{x}$ is because our SDE model for user activities is linear in both $\boldsymbol{x}$ and $\boldsymbol{u}$. Since $V(T) = \phi(T)$ is quadratic, as illustrated in [123], one can show by induction that when computing the value of V backward in time, $\boldsymbol{u}$ is always linear in $\boldsymbol{x}$.

Similarly, one can show that the value function V is linear in the state $\boldsymbol{x}$ for the linear cost case (opinion maximization problem), where $\phi = -\sum_u x_u(T)$, $\mathcal{L} = -\sum_u x_u(t) + \frac{\rho}{2}\|\boldsymbol{u}(t)\|^2$.

C.5 Derivation of the Optimal Control Policy for Least Square Opinion Guiding

In this section, we derive the optimal control policy for the opinion SDE defined in (6.6) with the least square opinion guiding cost. First, we restate the controlled SDE in (6.6) as follows.

$$\mathrm{d}x_i(t) = \big(b_i + u_i(\boldsymbol{x}, t) - x_i(t)\big)\mathrm{d}t + \theta \mathrm{d}w_i(t) + \sum\nolimits_{j=1}^{U} \alpha_{ij} x_j(t) \mathrm{d}N_j(t)$$

Putting it in the vector form, we have:

$$\mathrm{d}\boldsymbol{x}(t) = \big(\boldsymbol{b} - \boldsymbol{x} + \boldsymbol{u}\big)\mathrm{d}t + \theta \mathrm{d}\boldsymbol{w}(t) + \boldsymbol{h}(\boldsymbol{x})\mathrm{d}\boldsymbol{N}(t)$$

where the j-th column of $\boldsymbol{h}(\boldsymbol{x})$ captures how much influence that x_j has on all other users and is defined as $\boldsymbol{h}_j(\boldsymbol{x}) = \boldsymbol{B}^j\boldsymbol{x}$, where the matrix $\boldsymbol{B}^j \in \Re^{U\times U}$ and has the j-th column to be $(\alpha_{1j}, \cdots, \alpha_{Uj})^\top$ and zero elsewhere.

We substitute $f = \boldsymbol{b} - \boldsymbol{x}(t) + \boldsymbol{u}(,t)$, $\boldsymbol{g} = \theta$ and $\boldsymbol{h}$ to (6.10) and obtain the HJB equation as

$$-\frac{\partial V}{\partial t} = \min_{\boldsymbol{u}} \Big\{ \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) + \frac{\theta^2}{2}\mathrm{tr}\big(V_{\boldsymbol{xx}}(\boldsymbol{x}, t)\big) + V_{\boldsymbol{x}}(\boldsymbol{x}, t)^\top(\boldsymbol{b} - \boldsymbol{x}(t) + \boldsymbol{u}(t)) + \sum_{j=1}^{U} \lambda_j(t)\big(V(\boldsymbol{x} + \boldsymbol{h}_j(\boldsymbol{x}), t) - V(\boldsymbol{x}, t)\big) \Big\} \tag{C.30}$$

For the least square guiding problem, the instantaneous cost and terminal cost are defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{a}\|^2 + \frac{1}{2}\rho\|\boldsymbol{u}\|^2, \qquad \phi(T) = \frac{1}{2}\|\boldsymbol{x}(T) - \boldsymbol{a}\|^2$$

Hence we assume that value function V is quadratic in $\boldsymbol{x}$ with unknown coefficients $\boldsymbol{v}_1(t) \in \Re^U$, $\boldsymbol{v}_{11}(t) \in \Re^{U\times U}$ and $v_0(t) \in \Re$:

$$V(\boldsymbol{x}, t) = v_0(t) + \boldsymbol{v}_1(t)^\top\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{v}_{11}(t)\boldsymbol{x} \tag{C.31}$$

To find the optimal control, we substitute (C.31) to HJB equation and take the gradient of the right-hand side of the HJB equation (C.30) with respect to $\boldsymbol{u}$ and set it to $\mathbf{0}$. This yields the optimal feedback control policy:

$$\boldsymbol{u}^*(\boldsymbol{x}, t) = -\frac{1}{\rho}V_{\boldsymbol{x}} = -\frac{1}{\rho}\Big(\boldsymbol{v}_1(t) + \boldsymbol{v}_{11}(t)\boldsymbol{x}\Big) \tag{C.32}$$

Substitute $\boldsymbol{u}^*$ in (C.32) to the HJB equation, we first compute the four terms on the right side of the HJB equation. Note that the minimization is reached when $\boldsymbol{u} = \boldsymbol{u}^*$.

In the following derivations, we will use the property that $\boldsymbol{v}_{11} = \boldsymbol{v}_{11}^\top$ and $\boldsymbol{a}^\top\boldsymbol{b} =$

$\boldsymbol{b}^\top \boldsymbol{a}$ for any vector $\boldsymbol{a}$ and $\boldsymbol{b}$.

The first term is:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}^*, t) &= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{a} + \frac{1}{2}\rho \boldsymbol{u}^{*\top} \boldsymbol{u}^* \\
&= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{a} + \frac{1}{2\rho}(\boldsymbol{v}_1 + \boldsymbol{v}_{11}\boldsymbol{x})^\top (\boldsymbol{v}_1 + \boldsymbol{v}_{11}\boldsymbol{x}) \\
&= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{a} + \frac{1}{2\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1 + \frac{1}{\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_{11}\boldsymbol{x} + \frac{1}{2\rho}\boldsymbol{x}^\top \boldsymbol{v}_{11}\boldsymbol{v}_{11}\boldsymbol{x} \\
&= \underbrace{\frac{1}{2\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1}_{\text{scalar}} + \underbrace{\boldsymbol{x}^\top (\frac{1}{\rho}\boldsymbol{v}_{11}\boldsymbol{v}_1 - \boldsymbol{a})}_{\text{linear}} + \underbrace{\frac{1}{2}\boldsymbol{x}^\top (\frac{1}{\rho}\boldsymbol{v}_{11}\boldsymbol{v}_{11} + \boldsymbol{I})\boldsymbol{x}}_{\text{quadratic}}
\end{aligned}$$

Note that in line 1 of the expansion of $\mathcal{L}$, we dropped the constant term $\frac{1}{2}\boldsymbol{a}^\top \boldsymbol{a}$.

The second term is a scalar: $\mathrm{tr}\big(V_{\boldsymbol{xx}}(\boldsymbol{x}, t) = \frac{\theta^2}{2}\mathrm{tr}(\boldsymbol{v}_{11})$. The third term is

$$\begin{aligned}
&V_{\boldsymbol{x}}^\top (\boldsymbol{b} - \boldsymbol{x} + \boldsymbol{u}^*) \\
&= (\boldsymbol{v}_1 + \boldsymbol{v}_{11}\boldsymbol{x})^\top (\boldsymbol{b} - \boldsymbol{x} - \boldsymbol{u}^*) = (\boldsymbol{v}_1 + \boldsymbol{v}_{11}\boldsymbol{x})^\top (\boldsymbol{b} - \boldsymbol{x} - \frac{1}{\rho}(\boldsymbol{v}_1 + \boldsymbol{v}_{11}\boldsymbol{x})) \\
&= (\boldsymbol{v}_1^\top \boldsymbol{b} - \frac{1}{\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1) - (\boldsymbol{v}_1^\top \boldsymbol{x} + \frac{1}{\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_{11}\boldsymbol{x} + \frac{1}{\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_{11}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{v}_{11}\boldsymbol{x}) - \boldsymbol{x}^\top \boldsymbol{v}_{11}^\top \boldsymbol{x} - \frac{1}{\rho}\boldsymbol{x}^\top \boldsymbol{v}_{11}^\top \boldsymbol{v}_{11}\boldsymbol{x} \\
&= \underbrace{(\boldsymbol{v}_1^\top \boldsymbol{b} - \frac{1}{\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1)}_{\text{scalar}} - \underbrace{\boldsymbol{x}^\top (\boldsymbol{v}_1 + \frac{2}{\rho}\boldsymbol{v}_{11}\boldsymbol{v}_1 - \boldsymbol{v}_{11}\boldsymbol{b})}_{\text{linear}} - \underbrace{\frac{1}{2}\boldsymbol{x}^\top (2\boldsymbol{v}_{11} + \frac{2}{\rho}\boldsymbol{v}_{11}\boldsymbol{v}_{11})\boldsymbol{x}}_{\text{quadratic}}
\end{aligned}$$

The fourth term is

$$\begin{aligned}
&\sum_{j=1}^{U} \boldsymbol{\lambda}_j(t)(V(\boldsymbol{x} + \boldsymbol{h}_j(\boldsymbol{x}), t) - V(\boldsymbol{x}, t)) \\
&= \sum_{j=1}^{U} \boldsymbol{\lambda}_j(t)(\boldsymbol{v}_1^\top \boldsymbol{B}^j \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{B}^{j\top} \boldsymbol{v}_{11}\boldsymbol{B}^j \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top 2\boldsymbol{v}_{11}\boldsymbol{B}^j \boldsymbol{x}) \\
&= \underbrace{\boldsymbol{x}^\top \boldsymbol{\Lambda}^\top \boldsymbol{v}_1}_{\text{linear}} + \underbrace{\frac{1}{2}\boldsymbol{x}^\top (\sum_{j=1}^{U} \lambda_j \boldsymbol{B}^{j\top} \boldsymbol{v}_{11}\boldsymbol{B}^j + 2\boldsymbol{v}_{11}\boldsymbol{\Lambda})\boldsymbol{x}}_{\text{quadratic}}
\end{aligned}$$

where $\mathbf{\Lambda}(t) = \sum_{j=1}^{U} \lambda_j(t)\boldsymbol{B}^j$. Next, we compute the left side of HJB equation as:

$$-V_t = -v_0'(t) - \boldsymbol{x}^\top \boldsymbol{v}_1'(t) - \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{v}_{11}'(t)\boldsymbol{x}$$

By comparing the coefficients for the scalar, linear and quadratic terms in both left-hand-side and right-hand-side of the HJB equation, we obtain three ODEs as follows.

First, only consider all the coefficients quadratic in $\boldsymbol{x}$:

$$-\boldsymbol{v}_{11}'(t) = \boldsymbol{I} + 2\boldsymbol{v}_{11}(t)(-1 + \mathbf{\Lambda}(t)) + \sum_{j=1}^{U} \lambda_j(t)\boldsymbol{B}^{j\top}\boldsymbol{v}_{11}(t)\boldsymbol{B}^j - \frac{1}{\rho}\boldsymbol{v}_{11}(t)\boldsymbol{v}_{11}(t)$$

Second, consider the linear term:

$$-\boldsymbol{v}_1'(t) = -\boldsymbol{a} + (-1 + \mathbf{\Lambda}^\top(t) - \frac{1}{\rho}\boldsymbol{v}_{11}(t))\boldsymbol{v}_1(t) + \boldsymbol{v}_{11}(t)\boldsymbol{b}$$

Third, consider the scalar term:

$$-v_0'(t) = \boldsymbol{b}^\top \boldsymbol{v}_1(t) + \frac{\theta^2}{2}\mathrm{tr}(\boldsymbol{v}_{11}(t)) - \frac{1}{2\rho}\boldsymbol{v}_1^\top(t)\boldsymbol{v}_1(t)$$

Finally, we compute the terminal condition for the three ODEs by $V(\boldsymbol{x}(T), T) = \phi(\boldsymbol{x}(T), T)$:

$$V(X(T), T) = v_0(T) + \boldsymbol{x}(T)^\top \boldsymbol{v}_1(T) + \frac{1}{2}\boldsymbol{x}(T)^\top \boldsymbol{v}_{11}(T)\boldsymbol{x}(T)$$
$$\phi(\boldsymbol{x}(T), T) = -\boldsymbol{x}(T)^\top \boldsymbol{a} + \frac{1}{2}\boldsymbol{x}(T)^\top \boldsymbol{x}(T)$$

Hence $v_0(T) = 0$, $\boldsymbol{v}_1(T) = -\boldsymbol{a}$ and $\boldsymbol{v}_{11} = \boldsymbol{I}$. Note here we drop the constant term $\frac{1}{2}\boldsymbol{a}^\top \boldsymbol{a}$ in terminal cost ϕ.

Finally, we just need to use Algorithm 7 to solve these ODEs to obtain $\boldsymbol{v}_{11}(t)$ and $\boldsymbol{v}_1(t)$. Substituting $\boldsymbol{v}_{11}, \boldsymbol{v}_1$ to (C.32) leads to the optimal control policy.

C.6 Derivation of the Optimal Control Policy for Opinion Influence Maximization

In this section, we solve the opinion influence maximization problem. The solving scheme is similar to the least square opinion shaping cost, but the derivation is different due to different cost functions.

First, we choose $\omega = 1$ and restate the controlled opinion SDE in (6.6) as

$$\mathrm{d}x_i(t) = \big(b_i + u_i(\boldsymbol{x}, t) - x_i(t)\big)\mathrm{d}t + \theta \mathrm{d}w_i(t) + \sum\nolimits_{j=1}^{U} \alpha_{ij} x_j(t) \mathrm{d}N_j(t)$$

Putting it in the vector form, we have:

$$\mathrm{d}\boldsymbol{x}(t) = \big(\boldsymbol{b} - \boldsymbol{x} + \boldsymbol{u}\big)\mathrm{d}t + \theta \mathrm{d}\boldsymbol{w}(t) + \boldsymbol{h}(\boldsymbol{x})\mathrm{d}\boldsymbol{N}(t)$$

where the j-th column of $\boldsymbol{h}(\boldsymbol{x})$ captures how much influence that x_j has on all other users and is defined as $\boldsymbol{h}_j(\boldsymbol{x}) = \boldsymbol{B}^j \boldsymbol{x}$, where the matrix $\boldsymbol{B}^j \in \Re^{U \times U}$ and has the j-th column to be $(\alpha_{1j}, \cdots, \alpha_{Uj})^\top$ and zero elsewhere. We substitute $f = \boldsymbol{b} - \boldsymbol{x}$, $\boldsymbol{g} = \theta$ and $\boldsymbol{h}$ to (6.10) and obtain the HJB equation as follows

$$\begin{aligned} -\frac{\partial V}{\partial t} = \min_{\boldsymbol{u}} \Big\{ \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) + \frac{\theta^2}{2} \mathrm{tr}\big(V_{\boldsymbol{xx}}(\boldsymbol{x}, t)\big) + V_{\boldsymbol{x}}(\boldsymbol{x}, t)^\top (\boldsymbol{b} - \boldsymbol{x}(t) + \boldsymbol{u}(t)) \\ + \sum_{j=1}^{U} \lambda_j(t) \big(V(\boldsymbol{x} + \boldsymbol{h}_j(\boldsymbol{x}), t) - V(\boldsymbol{x}, t)\big) \Big\} \end{aligned} \tag{C.33}$$

For opinion influence maximization, we define the cost as follows. Suppose the goal is to maximize the opinion influence at each time on $[0, T]$, the instantaneous cost $\mathcal{L}$ is defined as:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) = -\sum_{j=1}^{U} x_i(t) + \frac{1}{2}\|\boldsymbol{u}(t)\|^2 = -\boldsymbol{x}(t)^\top \boldsymbol{1} + \frac{1}{2}\|\boldsymbol{u}(t)\|^2$$

where $\mathbf{1}$ is the column vector with each entry to be one. For the terminal cost, we have: $\phi(T) = -\boldsymbol{x}(T)^\top \mathbf{1}$.

Following the similar reasoning as the least square opinion guiding problem. Since the terminal cost ϕ is linear in the state $\boldsymbol{x}$, the value function must be linear in $\boldsymbol{x}$, since it is the optimal value of a linear function. Hence we set the value function $V(\boldsymbol{x}, t)$ to be a linear function in $\boldsymbol{x}$ with *unknown coefficients* $\boldsymbol{v}_1(t) \in \Re^U$ and $v_0(t) \in \Re$:

$$V(\boldsymbol{x}, t) = v_0(t) + \boldsymbol{v}_1(t)^\top \boldsymbol{x} \tag{C.34}$$

To find the optimal control, we substitute (C.34) to (C.33) and take the gradient of the right-hand-side of (C.33) with respect to $\boldsymbol{u}$ and set it to $\mathbf{0}$. This yields the optimal control policy:

$$\boldsymbol{u}^*(t) = -\frac{1}{\rho}\boldsymbol{V}_{\boldsymbol{x}} = -\frac{1}{\rho}\boldsymbol{v}_1(t) \tag{C.35}$$

Next, we just need to compute $\boldsymbol{v}_1(t)$ to find $\boldsymbol{u}^*$. Substitute $\boldsymbol{u}^*$ in (C.35) to the HJB equation, we will compute the four terms on the right side of the HJB equation and derive the ODEs by comparing the coefficients. Note that the minimization is reached when $\boldsymbol{u} = \boldsymbol{u}^*$.

First, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}^*, t)$ is expanded as:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}^*, t) = -\boldsymbol{x}^\top \mathbf{1} + \frac{1}{2}\|\boldsymbol{u}^*\|^2 = \underbrace{\frac{1}{2\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1}_{\text{scalar}} - \underbrace{\boldsymbol{x}^\top \mathbf{1}}_{\text{linear}}$$

Since V is linear in $\boldsymbol{x}$, $V_{\boldsymbol{xx}} = 0$. The third term is:

$$V_{\boldsymbol{x}}^\top(\boldsymbol{b} - \boldsymbol{x} + \boldsymbol{u}^*) = \boldsymbol{v}_1^\top(\boldsymbol{b} - \boldsymbol{x} - \frac{1}{\rho}\boldsymbol{v}_1) = \underbrace{\boldsymbol{v}^\top \boldsymbol{b} - \frac{1}{\rho}\boldsymbol{v}^\top \boldsymbol{v}_1}_{\text{scalar}} - \underbrace{\boldsymbol{x}^\top \boldsymbol{v}_1}_{\text{linear}}$$

The fourth term is:

$$\sum_{j=1}^{U} \lambda_j(t)(V(\boldsymbol{x}+\boldsymbol{h}_j(\boldsymbol{x}),t)-V(\boldsymbol{x},t)) = \sum_{j=1}^{U} \lambda_j(t)\boldsymbol{v}_1^\top \boldsymbol{h}_j(\boldsymbol{x}) = \underbrace{\boldsymbol{x}^\top \boldsymbol{\Lambda}^\top \boldsymbol{v}_1}_{\text{linear}}$$

where $\boldsymbol{\Lambda}(t) = \sum_{j=1}^{U} \lambda_j(t)\boldsymbol{B}^j$. Next, we compute the left-hand-side of HJB equation as:

$$-V_t = -v_0'(t) - \boldsymbol{x}^\top \boldsymbol{v}_1'(t) \tag{C.36}$$

Then by comparing the coefficients for the scalar and linear terms in both left side and right side of the HJB equation, we obtain two ODEs.

First, only consider all the coefficients linear in $\boldsymbol{x}$:

$$\boldsymbol{v}_1'(t) = \mathbf{1} + \boldsymbol{v}_1(t) - \boldsymbol{\Lambda}^\top \boldsymbol{v}_1(t) \tag{C.37}$$

Second, consider the linear term:

$$v_0'(t) = -\frac{1}{2\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1 - \boldsymbol{v}_1^\top \boldsymbol{b} + \frac{1}{\rho}\boldsymbol{v}_1^\top \boldsymbol{v}_1 = -\boldsymbol{v}_1(t)^\top \boldsymbol{b} + \frac{1}{2\rho}\boldsymbol{v}_1(t)^\top \boldsymbol{v}_1(t)$$

Hence we just need to solve the ODEs (C.36) to obtain $\boldsymbol{v}_1$ and then compute the optimal control $\boldsymbol{u}^*(t)$ from (C.35).

Finally we derive the terminal conditions for the above two ordinary differential equations. First, $V(T) = \phi(T) = -\boldsymbol{x}(T)^\top \mathbf{1}$ holds from the definition of the value function. Moreover, from the function form of V, we have $= v_0(T) + \boldsymbol{x}^\top \boldsymbol{v}_1(T)$. Hence by comparing the coefficients, we have $v_0(T) = 0$ and $\boldsymbol{v}_1(T) = -\mathbf{1}$.

With the above terminal condition and (C.37), we will use Algorithm 7 to solve for $\boldsymbol{v}_1(t)$ and obtain the optimal control policy.

APPENDIX D

PROOF OF THEOREMS IN CHAPTER 7

D.1 Derivation of the Optimal Measure

The problem of finding the optimal measure is as follows:

$$\min_{\mathbb{Q}} \Big[\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] + \gamma \mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P})\Big], \text{ s.t. } \int \mathrm{d}\mathbb{Q} = 1 \tag{D.1}$$

The minimum in (D.1) is attained at optimal measure $\mathbb{Q}^*$ given by:

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} = \frac{\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]} \tag{D.2}$$

Next, we show the derivations of (D.2), which contain two parts. First, we will show the following inequality:

$$\gamma \log\Big(\mathbb{E}_{\mathbb{P}}\Big[\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\Big]\Big) \leqslant \Big[\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] + \gamma \mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P})\Big] \tag{D.3}$$

The second part is to show the minimum of the above inequality is reached at (D.2).

To prove the first part, we first express $\mathbb{E}_{\mathbb{P}}$ in the left-hand-side of (D.3) as a function of the expectation $\mathbb{E}_{\mathbb{Q}}$. More specifically, we have:

$$\log\Big(\mathbb{E}_{\mathbb{P}}\Big[\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\Big]\Big) = \log\Big(\int \exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\mathrm{d}\mathbb{P}\Big) \tag{D.4}$$

$$= \log\Big(\int \exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\mathrm{d}\mathbb{Q}\Big) \tag{D.5}$$

$$\geqslant \int \log\Big(\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\Big)\mathrm{d}\mathbb{Q} \tag{D.6}$$

where (D.6) is due to the Jensen's inequality that puts the log operator inside the

integral. The measure $\mathbb{P}$ is absolute continuous with respect to $\mathbb{Q}$, hence the derivative $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ exists.

Moreover, using the property that $\log(ab) = \log a + \log b$ and $\log(1/a) = -\log a$, the right-hand-side of the above inequality can be written as:

$$\begin{aligned}\int \log\Big(\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\Big)\mathrm{d}\mathbb{Q} &= \int\Big(-\frac{1}{\gamma}S(\boldsymbol{x}) + \log\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\Big)\mathrm{d}\mathbb{Q} \\ &= \int -\frac{1}{\gamma}S(\boldsymbol{x})\mathrm{d}\mathbb{Q} + \int \log\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\mathrm{d}\mathbb{Q} \\ &= \int -\frac{1}{\gamma}S(\boldsymbol{x})\mathrm{d}\mathbb{Q} - \int \log\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\mathrm{d}\mathbb{Q} \\ &= -\frac{1}{\gamma}\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] - \mathbb{D}_{\mathbb{KL}}(\mathbb{Q}||\mathbb{P}) \qquad \text{(D.7)}\end{aligned}$$

Hence, combining (D.6) and (D.7), we have:

$$\log\Big(\mathbb{E}_{\mathbb{P}}\Big[\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\Big]\Big) \geqslant -\frac{1}{\gamma}\mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] - \mathbb{D}_{\mathbb{KL}}(\mathbb{Q}||\mathbb{P}) \qquad \text{(D.8)}$$

Finally, since $\gamma > 0$, multiply both sides of (D.8) by $-\gamma$ yields:

$$-\gamma\log\Big(\mathbb{E}_{\mathbb{P}}\Big[\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\Big]\Big) \leqslant \mathbb{E}_{\mathbb{Q}}[S(\boldsymbol{x})] + \gamma\mathbb{D}_{\mathbb{KL}}(\mathbb{Q}||\mathbb{P}) \qquad \text{(D.9)}$$

This finishes the proof of (D.3), the first part of the theorem. Next, we will show the minimum is reached at $\mathbb{Q}^*$ given by (D.2).

To prove the second part, we will substitute (D.2) to the right-hand-side of (D.8)

to show that the infimum is reached with this $\mathbb{Q}^*$. More specifically,

$$
\begin{aligned}
&\mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] + \gamma \mathbb{D}_{\mathbb{KL}}(\mathbb{Q}^* || \mathbb{P}) && \text{(D.10)}\\
&= \mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] + \gamma \int \log \frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} \mathrm{d}\mathbb{Q}^* \\
&= \mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] + \gamma \int \log \frac{\exp(-\frac{1}{\gamma} S(\boldsymbol{x}))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma} S(\boldsymbol{x}))]} \mathrm{d}\mathbb{Q}^* \\
&= \mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] + \gamma \int -\frac{1}{\gamma} S(\boldsymbol{x}) \mathrm{d}\mathbb{Q}^* - \gamma \int \log \Big(\mathbb{E}_{\mathbb{P}} \Big[\exp\big(-\frac{1}{\gamma} S(\boldsymbol{x}) \big) \Big] \Big) \mathrm{d}\mathbb{Q}^* && \text{(D.11)}\\
&= \mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] - \int S(\boldsymbol{x}) \mathrm{d}\mathbb{Q}^* - \gamma \log \Big(\mathbb{E}_{\mathbb{P}} \Big[\exp\big(-\frac{1}{\gamma} S(\boldsymbol{x}) \big) \Big] \Big) \int \mathrm{d}\mathbb{Q}^* \\
&= \mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] - \mathbb{E}_{\mathbb{Q}^*}[S(\boldsymbol{x})] - \gamma \log \Big(\mathbb{E}_{\mathbb{P}} \Big[\exp\big(-\frac{1}{\gamma} S(\boldsymbol{x}) \big) \Big] \Big) && \text{(D.12)}\\
&= -\gamma \log \Big(\mathbb{E}_{\mathbb{P}} \Big[\exp\big(-\frac{1}{\gamma} S(\boldsymbol{x}) \big) \Big] \Big)
\end{aligned}
$$

where (D.11) is due to the property $\log(a/b) = \log a - \log b$ and (D.12) is because $\mathbb{Q}^*$ is a probability measure hence $\int \mathrm{d}\mathbb{Q}^* = 1$. Hence the infimum is reached and this finishes the proof of the second part.

D.2 Proof of Theorem 18

Theorem 18. *For the intensity control problem in (7.3), we have:* $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\boldsymbol{u})} = \exp\big(\mathcal{D}(\boldsymbol{u})\big)$, *where* $\mathcal{D}(\boldsymbol{u})$ *is expressed as*

$$
\sum_{i=1}^{M} \int_0^T \big(u_i(s) - 1\big) \lambda_i(s) \mathrm{d}s - \int_0^T \log\big(u_i(s)\big) \mathrm{d}N_i(s)
$$

Proof. Intuitively, the derivative $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$ means the relative density of probability distribution $\mathbb{P}$ with respect to $\mathbb{Q}$. The change of probability measure happens because the intensity of the point process that drives the SDE in (6.6) is changed from $\boldsymbol{\lambda}(t)$ to $\boldsymbol{\lambda}(\boldsymbol{u}, t)$ in (7.3). Hence $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$ describes the change of probability measure for point processes and is the *likelihood ratio* between the uncontrolled and

controlled point process [106]:

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\boldsymbol{u})} = \frac{\exp\big(\mathcal{L}(\boldsymbol{\lambda})\big)}{\exp\big(\mathcal{L}(\boldsymbol{\lambda}(\boldsymbol{u}))\big)} = \exp\big(\mathcal{D}(\boldsymbol{u})\big),$$

where $\mathcal{L}$ is the *log-likelihood* for the multi-dimension point process with $\mathcal{L}(\boldsymbol{\lambda}) = \sum_{i=1}^{M} \mathcal{L}(\lambda_i)$. It is defined as the summation of log-likelihood $\mathcal{L}(\lambda_i)$ of each dimension i, where $\mathcal{L}(\lambda_i)$ is defined as follows [54]:

$$\mathcal{L}(\lambda_i(t)) = \int_0^T \log(\lambda_i(t))\mathrm{d}N_i(t) - \int_0^T \lambda_i(t)\mathrm{d}t \tag{D.13}$$

where the operation $\int f(t)\mathrm{d}N(t)$ is defined as the summation of the value of function f at each event time: $\int f(t)\mathrm{d}N(t) := \sum_i f(t_i)$.

Hence, $\mathcal{D}(\boldsymbol{u})$ denotes the difference of the log-likelihood between these two point processes:

$$\begin{aligned}
&\mathcal{D}(\boldsymbol{u}) = \mathcal{L}(\boldsymbol{\lambda}(t)) - \mathcal{L}(\tilde{\boldsymbol{\lambda}}(\boldsymbol{u}(t), t)) \\
&= \sum_{i=1}^{M} \bigg(\int_0^T \Big(\tilde{\lambda}_i(u_i(s), s) - \lambda_i(s)\Big)\mathrm{d}s - \int_0^T \log\Big(\frac{\tilde{\lambda}_i(u_i(s), s)}{\lambda_i(s)}\Big)\mathrm{d}N_i(s)\bigg) \\
&= \sum_{i=1}^{M} \bigg(\int_0^T \Big(u_i(s)\lambda_i(s) - \lambda_i(s)\Big)\mathrm{d}s - \int_0^T \log\Big(u_i(s)\Big)\mathrm{d}N_i(s)\bigg) \qquad \text{(D.14)} \\
&= \sum_{i=1}^{M} \bigg(\int_0^T \Big(u_i(s) - 1\Big)\lambda_i(s)\mathrm{d}s - \int_0^T \log\Big(u_i(s)\Big)\mathrm{d}N_i(s)\bigg)
\end{aligned}$$

where M is the dimension of point process. (D.14) comes from the form of control in (7.3). $\lambda_i(t), N_i(t), u_i(t)$ denote the i-th dimension of $\boldsymbol{\lambda}(t), \boldsymbol{N}(t), \boldsymbol{u}(t)$.

D.3 Derivation of the Optimal Control Policy

In this section, we will show the derivation of the optimal control policy in (7.13). We will formulate our objective function based on the form of optimal measure $\mathbb{Q}^*$

in (7.9). More specifically, we find a control $\boldsymbol{u}$ which pushes the controlled measure $\mathbb{Q}(\boldsymbol{u})$, as close to the optimal measure as possible. This leads to minimizing the Kullback-Leibler (KL) distance:

$$\boldsymbol{u}^* = \underset{\boldsymbol{u}>0}{\operatorname{argmin}} \, \mathbb{D}_{KL}(\mathbb{Q}^* || \mathbb{Q}(\boldsymbol{u})) \tag{D.15}$$

This objective function is in sharp contrast to traditional methods that solve the optimal control problem by computing the solution the HJB PDE, which have severe limitations in scalability and feasibility to nonlinear jump diffusion SDEs.

Next we simplify the objective function. According to the definition of KL divergence and chain rule of derivatives, we have:

$$\mathbb{D}_{KL}(\mathbb{Q}^* || \mathbb{Q}(\boldsymbol{u})) = \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\boldsymbol{u})}\right)\right] = \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\boldsymbol{u})}\right)\right] \tag{D.16}$$

The derivative $\mathrm{d}\mathbb{Q}^*/\mathrm{d}\mathbb{P}$ is given in (D.2) and $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$ is given in Theorem 18. Hence, we then substitute $\mathrm{d}\mathbb{Q}^*/\mathrm{d}\mathbb{P}$ and $\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q}(\boldsymbol{u})$ to (D.16). After removing terms which are independent of $\boldsymbol{u}$, the objective function (D.15) is simplified as:

$$\boldsymbol{u}^* = \underset{\boldsymbol{u}>0}{\operatorname{argmin}} \, \mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\boldsymbol{u})]$$

Next we parameterize $\boldsymbol{u}(t)$ as a piecewise constant function on $[0, T]$ as follows.

$$\boldsymbol{u}(t) = \begin{cases} \vdots \\ \boldsymbol{u}^k \quad \text{for } t \in [k\Delta t, (k+1)\Delta t) \\ \vdots \end{cases}$$

More specifically, the k-th piece is defined on $[k\Delta t, (k+1)\Delta t)$ as $\boldsymbol{u}^k$, where $k =$

$0, \cdots, K-1$, $t_k = k\Delta t$ and $T = t_K$. Then we have:

$$\mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\boldsymbol{u})] = \sum_{i=1}^{M}\sum_{k=1}^{K}\left(\mathbb{E}_{\mathbb{Q}^*}\Big[\int_{t_k}^{t_{k+1}}(u_i^k - 1)\lambda_i(s)\mathrm{d}s\Big] - \mathbb{E}_{\mathbb{Q}^*}\Big[\int_{t_k}^{t_{k+1}}\log(u_i^k)\mathrm{d}N_i(s)\Big]\right) \tag{D.17}$$

where u_i^k is the i-th dimension of $\boldsymbol{u}^k$. To compute u_i^k, we can neglect the two summation terms in (D.17) and only focus on the parts that involves u_i^k. Then we move u_i^k outside of the expectation and discard any constant terms. This yields the function that only involves u_i^k:

$$f(u_i^k) = u_i^k\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}}\lambda_i(s)\mathrm{d}s\big] - \log(u_i^k)\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}}\mathrm{d}N_i(s)\big] \tag{D.18}$$

We can then show $f(u_i^k)$ is convex in u_i^k. More specifically, it is in the form of $f(x) = ax - \log(x)b$ with $a > 0, b > 0$ and $f''(x) > 0$. Finally, setting $f'(u_i^k) = 0$ yields u_i^{k*}:

$$u_i^{k*} = \frac{\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}}\mathrm{d}N_i(s)\big]}{\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}}\lambda_i(s)\mathrm{d}s\big]} \tag{D.19}$$

However, u_i^{k*} is still not computable since the expectation is taken under the optimal probability measure $\mathbb{Q}^*$. Since we only known the SDE of the uncontrolled dynamics and can only compute the expectation under $\mathbb{P}$, we need to change the expectation from $\mathbb{E}_{\mathbb{Q}^*}$ to $\mathbb{E}_{\mathbb{P}}$ to compute u_i^{k*}.

To do this, we will use the following lemma.

Lemma 19. *Let the probability measure $\mathbb{Q}^*$ be defined as $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} = \frac{\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]}$ in (7.9), and $g(\boldsymbol{x}) : \Omega \to \Re$ be any measurable function. Then we have:*

$$\mathbb{E}_{\mathbb{Q}^*}[g(\boldsymbol{x})] = \frac{\mathbb{E}_{\mathbb{P}}\Big[\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)g(\boldsymbol{x})\Big]}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]}$$

Proof. The proof of this lemma uses the idea of importance sampling as follows.

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}^*}[g(\boldsymbol{x})] &= \int g(\boldsymbol{x})\mathrm{d}\mathbb{Q}^* \\
&= \int g(\boldsymbol{x})\frac{\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\mathrm{d}\mathbb{P}}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]} \\
&= \frac{\int \Big(g(\boldsymbol{x})\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)\Big)\mathrm{d}\mathbb{P}}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]} \\
&= \frac{\mathbb{E}_{\mathbb{P}}\Big[\exp\big(-\frac{1}{\gamma}S(\boldsymbol{x})\big)g(\boldsymbol{x})\Big]}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))]}
\end{aligned}$$

Finally, applying Lemma 19 to (D.19) yields the following expression for the optimal policy:

$$u_i^{k*} = \frac{\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}}\mathrm{d}N_i(s)\big]}{\mathbb{E}_{\mathbb{Q}^*}\big[\int_{t_k}^{t_{k+1}}\lambda_i(s)\mathrm{d}s\big]} = \frac{\frac{\mathbb{E}_{\mathbb{P}}\Big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\int_{t_k}^{t_{k+1}}\mathrm{d}N_i(s)\Big]}{\mathbb{E}_{\mathbb{P}}\Big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\Big]}}{\frac{\mathbb{E}_{\mathbb{P}}\Big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\int_{t_k}^{t_{k+1}}\lambda_i(s)\mathrm{d}s\Big]}{\mathbb{E}_{\mathbb{P}}\Big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\Big]}} = \frac{\mathbb{E}_{\mathbb{P}}\big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\int_{t_k}^{t_{k+1}}\mathrm{d}N_i(s)\big]}{\mathbb{E}_{\mathbb{P}}\big[\exp(-\frac{1}{\gamma}S(\boldsymbol{x}))\int_{t_k}^{t_{k+1}}\lambda_i(s)\mathrm{d}s\big]} \tag{D.20}$$

The derivation of the optimal policy is now complete.

D.4 Derivation of the Control Cost

We will derive the control cost in (7.8), which comes naturally from the dynamics. According to the definition of the KL divergence, we have:

$$\mathbb{D}_{KL}(\mathbb{Q}||\mathbb{P}) := \mathbb{E}_{\mathbb{Q}}[\log(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}})] = \mathbb{E}_{\mathbb{Q}}[C(\boldsymbol{u})] \tag{D.21}$$

Hence, the next step is to compute the derivative $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}$. This derivative means the relative density of probability distribution $\mathbb{Q}$ with respect to $\mathbb{P}$. According to [106],

we have:

$$\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} = \exp\Big(\sum_i \int_0^T \log\Big(\frac{\tilde{\lambda}_i(u_i(t),t)}{\lambda_i(t)}\Big)\mathrm{d}N_i(u_i(t),t) - \int_0^T (\tilde{\lambda}_i(u_i(t),t) - \lambda_i(t))\mathrm{d}t\Big), \tag{D.22}$$

Using the relationship that $\lambda_i(u_i(t),t) = \lambda_i(t)u_i(t)$, we have:

$$\begin{aligned}
&\mathbb{E}_{\mathbb{Q}}[\log(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}})] && \text{(D.23)}\\
&= \mathbb{E}_{\mathbb{Q}}\Big[\sum_i \int_0^T \log\Big(\frac{\tilde{\lambda}_i(u_i(t),t)}{\lambda_i(t)}\Big)\mathrm{d}\tilde{N}_i(u_i(t),t) - \int_0^T (\tilde{\lambda}_i(u_i(t),t) - \lambda_i(t))\mathrm{d}t\Big] && \text{(D.24)}\\
&= \mathbb{E}_{\mathbb{Q}}\Big[\sum_i \int_0^T \log\Big(u_i(t)\Big)\mathrm{d}\tilde{N}_i(u_i(t),t) - \int_0^T \Big(1 - \frac{1}{u_i(t)}\Big)\tilde{\lambda}_i(u_i(t),t)\mathrm{d}t\Big] && \text{(D.25)}\\
&= \mathbb{E}_{\mathbb{Q}}\Big[\sum_i \int_0^T \log\Big(u_i(t)\Big)\tilde{\lambda}_i(u_i(t),t)\mathrm{d}t + \int_0^T \Big(1 - \frac{1}{u_i(t)}\Big)\tilde{\lambda}_i(u_i(t),t)\mathrm{d}t\Big] && \text{(D.26)}
\end{aligned}$$

Note that (D.25) to (D.26) follows from the Campbell theorem [60]. Therefore, the control cost is:

$$\begin{aligned}
C(\boldsymbol{u}) &= \int_0^T \sum\nolimits_i \Big(\log(u_i(t)) + \frac{1}{u_i(t)} - 1\Big)\tilde{\lambda}_i(u_i(t),t)\mathrm{d}t\\
&= \int_0^T \sum\nolimits_i \Big(\log(u_i(t)) + \frac{1}{u_i(t)} - 1\Big)u_i(t)\lambda_i(t)\mathrm{d}t
\end{aligned}$$

REFERENCES

[1] Y. Wang, N. Du, R. Trivedi, and L. Song, "Coevolutionary latent feature processes for continuous-time user-item interactions," in *NIPS*, 2016, pp. 4547–4555.

[2] Y. Wang, B. Xie, N. Du, and L. Song, "Isotonic hawkes processes," in *ICML*, 2016, pp. 2226–2234.

[3] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Deep coevolutionary network: Embedding user and item features for recommendation," *ArXiv preprint :1609.03675*, 2016.

[4] M. Farajtabar, Y. Wang, M. Gomez-Rodriguez, S. Li, H. Zha, and L. Song, "Coevolve: A joint point process model for information diffusion and network co-evolution," in *NIPS*, 2015, pp. 1954–1962.

[5] N. Du, Y. Wang, N. He, and L. Song, "Time sensitive recommendation from recurrent user activities," in *NIPS*, 2015, pp. 3492–3500.

[6] Y. Wang, X. Ye, H. Zha, and L. Song, "Predicting user activity level in point processes with mass transport equation," in *NIPS*, 2017.

[7] Y. Wang, X. Ye, H. Zhou, H. Zha, and L. Song, "Linking micro event history to macro prediction in point process models," in *AISTAT*, 2017, pp. 1375–1384.

[8] Y. Wang, G. Williams, E. Theodorou, and L. Song, "Variational policy for guiding point processes," in *ICML*, 2017.

[9] Y. Wang, E. Theodorou, A. Verma, and L. Song, "A stochastic differential equation framework for guiding online user activities in closed loop," in *AISTATS*, 2018.

[10] Y. Koren, "Collaborative filtering with temporal dynamics," in *KDD*, 2009.

[11] L. Charlin, R. Ranganath, J. McInerney, and D. M. Blei, "Dynamic poisson factorization," in *RecSys*, 2015.

[12] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, Dec. 1966.

[14] R. Kalman, "A new approach to linear filtering and prediction problems," vol. 82, pp. 35–45, 1960.

[15] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994.

[16] J. Janssen and N. Limnios, *Semi-Markov Models and Applications*. Kluwer Academic, 1999.

[17] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Conference on Knowledge and Information Management CKIM*, New Orleans, LA, USA: ACM, 2003, pp. 556–559.

[18] E. Adar and L. A. Adamic, "Tracking information epidemics in blogspace.," in *Web Intelligence*, 2005, pp. 207–214.

[19] W. de Nooy, "Networks of action and events over time. a multilevel discrete-time event history model for longitudinal network data.," *Social Networks*, vol. 33, no. 1, pp. 31–40, 2011.

[20] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: Membership, growth, and evolution," in *Proc. of KDD'06*, 2006.

[21] Y. Matsubara, Y. Sakurai, W. G. van Panhuis, and C. Faloutsos, "Funnel: Automatic mining of spatially coevolving epidemics," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, New York, New York, USA: ACM, 2014, pp. 105–114, ISBN: 978-1-4503-2956-9.

[22] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa, "Fast mining and forecasting of complex time-stamped events," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12, Beijing, China: ACM, 2012, pp. 271–279, ISBN: 978-1-4503-1462-6.

[23] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: Model and implications," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12, Beijing, China, 2012, pp. 6–14, ISBN: 978-1-4503-1462-6.

[24] T. A. B. Snijders, "Statistical methods for network dynamics," in *Proc of the Scientific Meeting of the Italian Statistical Society*, 2006.

[25] E. Lewis and G. Mohler, "A nonparametric em algorithm for multiscale hawkes processes," *Preprint*, pp. 1–16, 2011.

[26] G. O. Mohler, M. B. Short, P. J. Brantingham, F. P. Schoenberg, and G. E. Tita, "Self-exciting point process modeling of crime," *Journal of the American Statistical Association*, vol. 106, no. 493, 2011.

[27] M. Egesdal, C. Fathauer, K. Louie, J. Neuman, G. Mohler, and E. Lewis, "Statistical and stochastic modeling of gang rivalries in los angeles," *SIAM Undergraduate Research Online*, vol. 3, pp. 72–94, 2010.

[28] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha, "Scalable influence estimation in continuous-time diffusion networks," in *NIPS*, 2013.

[29] X. He, T. Rekatsinas, J. Foulds, L. Getoor, and Y. Liu, "Hawkestopic: A joint model for network inference and topic modeling from text-based cascades," in *ICML*, 2015, pp. 871–880.

[30] H. Xu, Y. Zhen, and H. Zha, "Trailer generation via a point process-based visual attractiveness model," in *Proceedings of the 24th International Conference on Artificial Intelligence*, AAAI Press, 2015, pp. 2198–2204.

[31] P. O. S. Vaz de Melo, C. Faloutsos, and A. F. A. Loureiro, "Human dynamics in large communication networks," in *SIAM Conference on Data Mining (SDM)*, 2011, pp. 879–968.

[32] P. O. S. Vaz de Melo, C. Faloutsos, R. Assunção, and A. Loureiro, "The self-feeding process: A unifying model for communication dynamics in the web," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13, Rio de Janeiro, Brazil: ACM, 2013, pp. 1319–1330, ISBN: 978-1-4503-2035-1.

[33] N. Masuda, T. Takaguchi, N. Sato, and K. Yano, "Self-exciting point process modeling of conversation event sequences," *Temporal Networks*, pp. 245–264, 2013.

[34] R. D. Malmgren, J. M. Hofman, L. A. N. Amaral, and D. J. Watts, "Characterizing individual communication patterns.," in *KDD*, ACM, 2009, pp. 607–616.

[35] M. Farajtabar, N. Du, M. Gomez-Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *NIPS*, 2014, pp. 2474–2482.

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

[37] P. Brémaud and L. Massoulié, "Stability of nonlinear hawkes processes," *The Annals of Probability*, pp. 1563–1588, 1996.

[38] L. Carstensen, A. Sandelin, O. Winther, and N. R. Hansen, "Multivariate hawkes process models of the occurrence of regulatory elements," *BMC bioinformatics*, vol. 11, no. 1, p. 456, 2010.

[39] N. R. Hansen, P. Reynaud-Bouret, V. Rivoirard, *et al.*, "Lasso and probabilistic inequalities for multivariate point processes," *Bernoulli*, vol. 21, no. 1, pp. 83–143, 2015.

[40] L. Zhu, "Large deviations for markovian nonlinear hawkes processes," *The Annals of Applied Probability*, 2015.

[41] L. Paninski, "Estimating entropy on m bins given fewer than m samples," *IEEE Transactions on Information Theory*, vol. 50, pp. 2200–2203, 2004.

[42] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *Journal of neurophysiology*, vol. 93, no. 2, pp. 1074–1089, 2005.

[43] R. Barlow, D. Bartholomew, J. M. Bremner, and H. D. Brunk, *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*. J. Wiley, 1972.

[44] T. Robertson, F. Wright, R. L. Dykstra, and T Robertson, *Order restricted statistical inference*. Wiley New York, 1988, vol. 229.

[45] P. Mair, K. Hornik, and J. de Leeuw, "Isotone optimization in r: Pool-adjacent-violators algorithm (pava) and active set methods," *Journal of statistical software*, vol. 32, no. 5, pp. 1–24, 2009.

[46] J. L. Horowitz and W. Härdle, "Direct semiparametric estimation of single-index models with discrete covariates," *Journal of the American Statistical Association*, vol. 91, no. 436, pp. 1632–1640, 1996.

[47] M. Hristache, A. Juditsky, and V. Spokoiny, "Direct estimation of the index coefficient in a single-index model," *Annals of Statistics*, 2001.

[48] P. A. Naik and C.-L. Tsai, "Isotonic single-index model for high-dimensional database marketing," *Computational statistics & data analysis*, 2004.

[49] M. Delecroix, M. Hristache, and V. Patilea, "On semiparametric m-estimation in single-index regression," *Journal of Statistical Planning and Inference*, 2006.

[50] A. T. Kalai and R. Sastry, "The isotron algorithm: High-dimensional isotonic regression.," in *COLT*, 2009.

[51] S. M. Kakade, V. Kanade, O. Shamir, and A. Kalai, "Efficient learning of generalized linear and single index models with isotonic regression," in *NIPS*, 2011.

[52] S. Acharyya and J. Ghosh, "Parameter estimation of generalized linear models without assuming their link function," in *AISTAT*, 2015.

[53] T. Ozaki, "Maximum likelihood estimation of hawkes' self-exciting point processes," *Annals of the Institute of Statistical Mathematics*, vol. 31, no. 1, pp. 145–155, 1979.

[54] O. Aalen, O. Borgan, and H. Gjessing, *Survival and event history analysis: A process point of view*. Springer, 2008.

[55] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," *Soviet Math. Docl.*, vol. 269, pp. 543–547, 1983.

[56] C.-H. Zhang, "Risk bounds in isotonic regression," *The Annals of Statistics*, vol. 30, no. 2, pp. 528–555, 2002.

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

[58] L. Li and H. Zha, "Learning parametric models for social infectivity in multi-dimensional hawkes processes," in *AAAI*, 2014.

[59] Y. Ogata, "On lewis' simulation method for point processes," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 23–31, 1981.

[60] D. Daley and D. Vere-Jones, *An introduction to the theory of point processes: Volume II: General theory and structure*. Springer, 2007, vol. 2.

[61] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[62] Y. Wang, R. Chen, J. Ghosh, J. C. Denny, A. Kho, Y. Chen, B. A. Malin, and J. Sun, "Rubik: Knowledge guided tensor factorization and completion for health data analytics," in *KDD*, 2015, pp. 1265–1274.

[63] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan, "Beyond clicks: Dwell time for personalization," in *RecSys*, 2014.

[64] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *ICML*, 2008.

[65] Y. Chen, D. Pavlov, and J. Canny, "Large-scale behavioral targeting," in *KDD*, J. Elder, F. Fogelman-Soulié, P. Flach, and M. J. Zaki, Eds., 2009.

[66] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *KDD*, J. Elder, F. Fogelman-Soulié, P. Flach, and M. Zaki, Eds., 2009.

[67] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Foundations and Trends in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[68] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, "Like like alike: Joint friendship and interest propagation in social networks," in *WWW*, 2011.

[69] Y. Wang and A. Pal, "Detecting emotions in social media: A constrained optimization approach," in *IJCAI*, 2015.

[70] S. Gultekin and J. Paisley, "A collaborative kalman filter for time-evolving dyadic processes," in *ICDM*, 2014, pp. 140–149.

[71] J. Z. J. L. Preeti Bhargava Thomas Phan, "Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data," in *WWW*, 2015.

[72] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Recsys*, 2010.

[73] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization.," in *SDM*, 2010, pp. 211–222.

[74] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical poisson factorization," *UAI*, 2015.

[75] B. Hidasi and D. Tikk, "General factorization framework for context-aware recommendations," *Data Mining and Knowledge Discovery*, pp. 1–30, 2015.

[76] X. Wang, R. Donaldson, C. Nell, P. Gorniak, M. Ester, and J. Bu, "Recommending groups to users using user-group engagement and time-dependent matrix factorization," in *AAAI*, 2016.

[77] S. Sastry, "Some np-complete problems in linear algebra," *Honors Projects*, 1990.

[78] L. Baltrunas and X. Amatriain, *Towards time-dependant recommendation based on implicit feedback*, 2009.

[79] K. Kapoor, K. Subbian, J. Srivastava, and P. Schrater, "Just in time recommendations: Modeling the dynamics of boredom in activity streams," in *WSDM*, 2015.

[80] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.

[81] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *KDD*, 2016.

[82] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 307–361, 2012.

[83] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2265–2273.

[84] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *ICML*, 2016.

[85] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv preprint arXiv:1412.6980*, 2014.

[86] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[87] G. Ballard, T. G. Kolda, A. Pinar, and C Seshadhri, "Diamond sampling for approximate maximum all-pairs dot-product (mad) search," in *Data Mining (ICDM), 2015 IEEE International Conference on*, IEEE, 2015, pp. 11–20.

[88] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone," *ISA Annual Convention*, 2013.

[89] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, and M. Ward, "Icews coded event data," 2017.

[90] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 809–816.

[91] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems*, 2013, pp. 926–934.

[92] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 601–610.

[93] Y. Lin, Z. Liu, M. Sun, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," 2015.

[94] J. Pan, V. Rao, P. Agarwal, and A. Gelfand, "Markov-modulated marked poisson processes for check-in data," in *ICML*, 2016, pp. 2244–2253.

[95] X. Tan, S. A. Naqvi, A. Y. Qi, K. A. Heller, and V. Rao, "Content-based modeling of reciprocal relationships using hawkes and gaussian processes," in *UAI*, 2016, pp. 726–734.

[96] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *ICML*, 2017.

[97] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes," in *AISTAT*, vol. 31, 2013, pp. 641–649.

[98] S. Gao, J. Ma, and Z. Chen, "Modeling and predicting retweeting dynamics on microblogging platforms," in *WSDM*, 2015.

[99] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *KDD*, 2015.

[100] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *Reviews of modern physics*, vol. 87, no. 3, p. 925, 2015.

[101] N. Du, L. Song, A. J. Smola, and M. Yuan, "Learning networks of heterogeneous influence," in *NIPS*, 2012.

[102] S.-H. Yang and H. Zha, "Mixture of mutually exciting processes for viral diffusion," in *ICML*, 2013, pp. 1–9.

[103] L. Yu, P. Cui, F. Wang, C. Song, and S. Yang, "From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics," in *ICDM*, 2015.

[104] J. Da Fonseca and R. Zaatour, "Hawkes process: Fast calibration, application to trade clustering, and diffusive limit," *Journal of Futures Markets*, vol. 34, no. 6, pp. 548–579, 2014.

[105] D. Blackwell, "Conditional expectation and unbiased sequential estimation," *The Annals of Mathematical Statistics*, pp. 105–110, 1947.

[106] P. Brémaud, "Point processes and queues.," 1981.

[107] I. M. Gelfand, R. A. Silverman, *et al.*, *Calculus of variations*. Courier Corporation, 2000.

[108] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[109] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[110] D. Antoniades and C. Dovrolis, "Co-evolutionary dynamics in social networks: A case study of twitter," *ArXiv preprint arXiv:1309.6001*, 2013.

[111] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, ACM, 2003, pp. 137–146.

[112] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *KDD*, 2016.

[113] W. Lian, R. Henao, V. Rao, J. E. Lucas, and L. Carin, "A multitask point process predictive model," in *ICML*, 2015, pp. 2030–2038.

[114] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems*, 2016.

[115] X. He and Y. Liu, "Not enough data?: Joint inferring multiple diffusion networks via network generation priors," in *WSDM*, 2017.

[116] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD*, 2009.

[117] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *KDD*, 2010.

[118] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *ICDM*, 2010.

[119] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan, "Influence maximization in social networks when negative opinions may emerge and propagate.," in *SDM*, 2012.

[120] R Boel and P Varaiya, "Optimal control of jump processes," *SIAM Journal on Control and Optimization*, vol. 15, no. 1, pp. 92–119, 1977.

[121] H. Pham, "Optimal stopping of controlled jump diffusion processes: A viscosity solution approach," in *Journal of Mathematical Systems, Estimation and Control*, Citeseer, 1998.

[122] B. K. Oksendal and A. Sulem, *Applied stochastic control of jump diffusions*. Springer, 2005, vol. 498.

[123] F. B. Hanson, *Applied stochastic processes and control for Jump-diffusions: Modeling, analysis, and computation*. Siam, 2007, vol. 13.

[124] D. Thalmeier, V. Gómez, and H. J. Kappen, "Action selection in growing state spaces: Control of network structure growth," *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 3, p. 034 006, 2016.

[125] R. S. Epanchin-Niell and J. E. Wilen, "Optimal spatial control of biological invasions," *Journal of Environmental Economics and Management*, vol. 63, no. 2, pp. 260–270, 2012.

[126] M. Ogura and V. M. Preciado, "Stability of spreading processes over time-varying large-scale networks," *IEEE Transactions on Network Science and Engineering*, vol. 3, no. 1, pp. 44–57, 2016.

[127] A. De, I. Valera, N. Ganguly, S. Bhattacharya, and M. G. Rodriguez, "Learning opinion dynamics in social networks," *ArXiv preprint arXiv:1506.05474*, 2015.

[128] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, 2012.

[129] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *ICML*, 2012.

[130] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006.

[131] S. M. Iacus, *Simulation and inference for stochastic differential equations: With R examples*. Springer Science & Business Media, 2009, vol. 1.

[132] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *KDD*, 2009.

[133] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, "The development and psychometric properties of liwc2015," *UT Faculty/Researcher Works*, 2015.

[134] W. Lian, V. Rao, B. Eriksson, and L. Carin, "Modeling correlated arrival events with latent semi-markov processes," in *ICML*, 2014, pp. 396–404.

[135] N. He, Z. Harchaoui, Y. Wang, and L. Song, "Fast and simple optimization for poisson likelihood models," *ArXiv preprint arXiv:1608.01264*, 2016.

[136] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Steering user behavior with badges," in *WWW*, 2013, pp. 95–106.

[137] M. Karimi, E. Tavakoli, M. Farajtabar, L. Song, and M. Gomez-Rodriguez, "Smart broadcasting: Do you want to be seen?" In *KDD*, 2016, pp. 1635–1644.

[138] E. Bayraktar and M. Ludkovski, "Liquidation in limit order books with controlled intensity," *Mathematical Finance*, vol. 24, no. 4, pp. 627–650, 2014.

[139] A. Zarezade, U. Upadhyay, H. R. Rabiee, and M. Gomez-Rodriguez, "Redqueen: An online algorithm for smart broadcasting in social networks," in *WSDM*, ACM, 2017, pp. 51–60.

[140] M. Ades, P. E. Caines, and R. P. Malhamé, "Stochastic optimal control under poisson-distributed observations," *Automatic Control, IEEE Transactions on*, vol. 45, no. 1, pp. 3–13, 2000.

[141] M. Lemmon, "Event-triggered feedback in control, estimation, and optimization," in *Networked Control Systems*, Springer, 2010, pp. 293–358.

[142] W. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, 2012, pp. 3270–3285.

[143] X. Meng, B. Wang, T. Chen, and M. Darouach, "Sensing and actuation strategies for event triggered stochastic optimal control," in *52nd IEEE Conference on Decision and Control*, IEEE, 2013, pp. 3097–3102.

[144] P. Dupuis and R. S. Ellis, *A weak convergence approach to the theory of large deviations*. John Wiley & Sons, 1997, vol. 902.

[145] E. A. Theodorou, "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations," *Entropy*, vol. 17, no. 5, pp. 3352–3375, 2015.

[146] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," UC Berkeley, Department of Statistics, Tech. Rep. 649, 2003.

[147] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1433–1440.

[148] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[149] E. Süli and D. F. Mayers, *An introduction to numerical analysis*. Cambridge university press, 2003.

[150] R. Liptser and A. N. Shiryayev, *Theory of martingales*. Springer Science & Business Media, 2012, vol. 49.

[151] R. M. Dudley, *Real analysis and probability*. Cambridge, UK: Cambridge University Press, 2002.

VITA

Yichen Wang is a PhD candidate the School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology. His research focuses on designing machine learning models and scalable algorithms for statistical analysis of temporal dynamic processes, with applications in improving the understanding of users' temporal and strategic behaviors, promoting users' engagement in service platforms, and time-sensitive recommendations. Yichen has received several recognitions for his research, including the 2017 IBM PhD Fellowship, the RecSys'16 Deep Learning Workshop Best Paper Award, and the 2016 Annual Outstanding Graduate Research Assistant Award of Georgia Tech.