Methods for Event Time Series Prediction and Anomaly Detection

by

Siqi Liu

Submitted to the Graduate Faculty of

the Department of Computer Science in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF COMPUTER SCIENCE

This dissertation was presented

by

Siqi Liu

It was defended on

June 22, 2020

and approved by

Milos Hauskrecht, Department of Computer Science, University of Pittsburgh

Adriana Kovashka, Department of Computer Science, University of Pittsburgh

Diane Litman, Department of Computer Science, University of Pittsburgh

Barnabas Poczos, Machine Learning Department, Carnegie Mellon University

Dissertation Director: Milos Hauskrecht, Department of Computer Science, University of Pittsburgh

Methods for Event Time Series Prediction and Anomaly Detection

Siqi Liu, PhD

University of Pittsburgh, 2020

Event time series are sequences of events occurring in continuous time. They arise in many real-world problems and may represent, for example, posts in social media, administrations of medications to patients, or adverse events, such as episodes of atrial fibrillation or earthquakes. In this work, we study and develop methods for prediction and anomaly detection on event time series. We study two general approaches. The first approach converts event time series to regular time series of counts via time discretization. We develop methods relying on (a) nonparametric time series decomposition and (b) dynamic linear models for regular time series. The second approach models the events in continuous time directly. We develop methods relying on point processes. For prediction, we develop a new model based on point processes to combine the advantages of existing models. It is flexible enough to capture complex dependency structures between events, while not sacrificing applicability in common scenarios. For anomaly detection, we develop methods that can detect new types of anomalies in continuous time and that show advantages compared to time discretization.

Table of Contents

List of Tables

List of Figures

Preface

First of all, I would like to thank my advisor, Dr. Milos Hauskrecht. This dissertation would not be possible without his academic guidance and the financial support from the funding he received. I would also like to thank all the other professors in my dissertation committee: Dr. Adriana Kovashka, Dr. Diane Litman, and Dr. Barnabas Poczos. They provided many insightful comments and suggestions that helped me to improve the dissertation.

I had the fortune to collaborate with several people outside our department during my PhD study, including Dr. Adam Wright, Dr. Dean F. Sittig, Dr. Gilles Clermont, Dr. Gregory Cooper, and Dr. Shyam Visweswaran. I would like to thank all of them for the help and inspiration they provided.

I am grateful for the continuous financial support provided by the grants from the National Institutes of Health (NIH) and the fellowships and assistantships from the Department of Computer Science, previously at the Dietrich School of Arts and Sciences and now at the School of Computing and Information, University of Pittsburgh. Also, I would like to thank the staff at our department and schools for their help and support, especially Keena Walker, who is always responsive and tries to help as much as she can.

I am grateful for all the professors teaching the interesting courses I took both inside and outside our department. They helped to widen my horizons and lay the foundation of my research. I also appreciate the help I received from my friends and fellow PhD students and would like to thank all of them, especially the students from our lab, with whom I had many fun moments and interesting discussions.

Last but not least, I would like to thank my parents, relatives, and old friends for all kinds of support and encouragement they gave to me at different times during the entire course of pursing my PhD.

1.0 Introduction

In the past decades, artificial intelligence (AI) has gained increasing attention from both academia and industry. Machine learning, which enables automated systems to improve their performance through experience, is at the core of modern AI systems. The experience is typically collected as data. Due to theoretical simplicity, machine learning research has traditionally focused on data that consist of instances that are assumed to independent (from each other) and come from a fixed probability distribution. That is, they are independent and identically distributed (IID). However, with success in learning models from IID data, researchers have begun to explore more challenging data types that exhibit various dependences and relations. Time series have been one of the most important data types.

In everyday life, time is an essential component of the information we perceive from the world. In fact, time is always in the data we collect, and the IID assumption is only a simplification of the more complex real world we observe. Different from IID data, in a time series, a timestamp is associated with each observation. Usually, the timestamp records when the observation has been made.

By bringing in the dimension of time, we can get many interesting and useful insights that would otherwise be impossible to get. For example, trends are the overall changes of the time series over time. They can be increasing or decreasing, and they are important in problems like forecasting stock prices. For instance, if we ignore them and treat the price of a stock over the past few weeks as IID, then a reasonable prediction of the price tomorrow could be based on the mean of part or all of the past data. But if we consider the trends (e.g., the price has been increasing for the past few weeks), then our prediction could be a linear or nonlinear extrapolation of the past data, which should be more reasonable and accurate in most cases.

In Figure 1, we show an example of time series. These are the numbers of bike rentals that occurred every day in the Bay area [1]. Each data point represent the observed counts for a day, while the location of the point on the X-axis corresponds to the time of the observation.

[1] `https://www.kaggle.com/benhamner/sf-bay-area-bike-share`

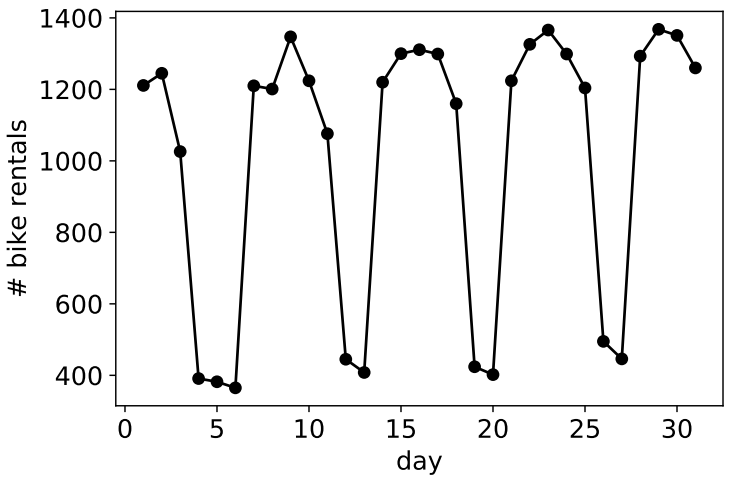


Figure 1: Bay area bike rental daily counts. Each point in the figure represents the number of bike trips in a day. Time is restricted to July 1 to July 31 in 2014.

By arranging the data according to the time, we can see some patterns in the data across the time that would otherwise be invisible.

In most cases, time series are observed at regular time intervals as we saw in the previous example, but some of these observations are actually aggregations of lower level observations. Again, consider the bike rental count time series in Figure 1. Each point is the total number of bike rentals in a day, which is the aggregation of all the bike rentals that happened in that day. If we record each individual bike rental with its time, then we get a different view of the same process with more details as shown in Figure 2. In the figure, each point (and its stem) represent an individual event of bike rental. The location of the point on the X-axis corresponds to the exact time of that event. We call this type of time series *event time series* or *event sequences*.

The main difference between event time series and regular time series is in the role of the times. In regular time series, the times are treated as indices that help to order the sequence of values of the target variable observed at different times. In event time series, the times are one of the targets, if not the only one. In many cases, we do not have any values associated with each event and only observe the occurrence of the event. The task is to learn the time dependencies between the events, to infer the states based on the occurrences of the events,

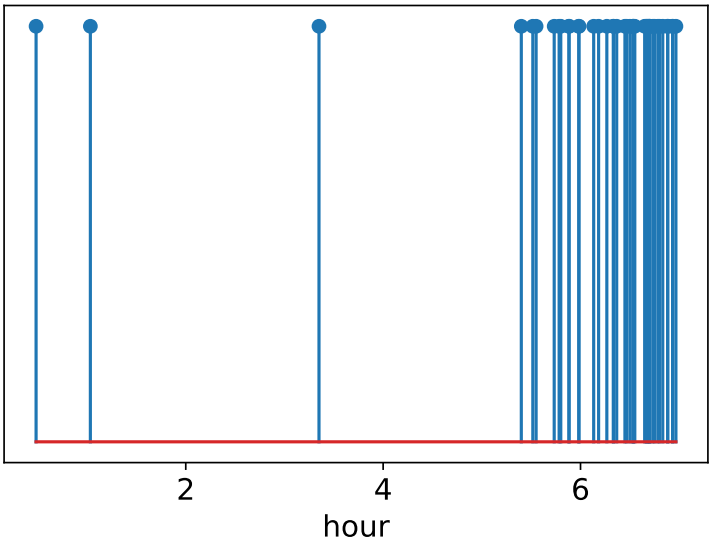


Figure 2: Bay area bike rental events. Each stem in the figure represents the event of a trip beginning in a bike share system. Time is restricted to 0:00 to 7:00 on July 1, 2014.

or to make predictions of the times of future events. Another difference between regular time series and event time series is the regularity of the times. The times in a regular time series are *regular*, meaning the time intervals between consecutive observations are always of the same length. In contrast, the times of an event time series are typically *irregular*, meaning the time intervals between consecutive observations have different lengths. In fact, the irregularity in time is assumed to be part of the randomness in most probabilistic models for event time series.

In real life, events are abundant, and being able to predict their occurrences can be very valuable. For example, in automated driving, by monitoring the condition of the car and the surroundings, it would be very useful if we can reliably predict whether a traffic accident is likely to happen in the foreseeable future, and in case it is, we can take measures to prevent it beforehand. In clinical care, by monitoring the condition of the patient, it would be very helpful if we can reliably predict whether the patient will experience an adverse event in the near future, e.g., an episode of atrial fibrillation, and in the case they will, we can alert the physicians to mitigate the negative effects of the events beforehand or at least notify

them to anticipate the event. Both the car accidents and the adverse events for a patient can be represented as events in continuous time and therefore form an event time series. The occurrences of the events themselves are the main target we wish to precisely model and predict.

Meanwhile, by modeling event time series, we can learn the patterns of the occurrences of the events in time. The patterns reflect the expected behaviors of the underlying systems that generate the events. However, in reality, there could be abnormal interruptions to these patterns that appear differently from the expected behaviors. Algorithmic detection of these abnormal behaviors or *anomalies* can be very useful. For example, the events of the administration of a medication are expected to be dependent on the recent medical condition of the patient. However, there could be cases that the medication is not administered when the condition requires it, or that it is administered when it should not. By learning a model on the pattern of the expected behaviors, we can potentially detect these anomalous behaviors automatically and send timely alerts to the health care provider. Therefore, algorithmic detection of anomalies in event time series can be really important and useful.

1.1 Problems and Approaches

In this dissertation, we study two general problems in event time series: predictive modeling and anomaly detection. We claim that these two problems are highly related, as a predictive model can help us develop anomaly detection methods. This point is revisited as one of our hypotheses in Section 1.2 and demonstrated in our methods for anomaly detection in event time series in later chapters. We briefly introduce both problems in the following sections.

1.1.1 Predictive Modeling of Event Time Series

We first highlight the questions we might wish to answer related to predictive modeling. We use the bike rental events as an example. Suppose we have observed these events for a

period of time. In terms of predictive modeling, we may wish to answer different types of questions as follows.

Q1 When will the next bike rental happen?

Q2 How many bike rentals will happen in the next 1, 12, or 24 hours?

Q3 Will there be any bike rental in the next 1, 12, or 24 hours?

To answer these questions, we may model the event time series differently depending on the requirements. In general, time series and their models can be categorized into three types:

- **Regular time series.** This is the type of time series that has been most widely studied. The observations are made at regular time points, e.g., every day, where the intervals between consecutive time points always have the same length. The times of the observations act as the indices of the data. In this case, the focus of predictive modeling is on capturing the dependencies between the observed values of the time series and forecasting the future values based on the dependencies and the history.
- **Irregular time series.** This is the type of time series that is closely related to the previous case. The difference is that the observations are made at irregular time points, such that the intervals between time points may have different lengths. The times of the observations still act as the indices of the data, although the indices are not regular anymore. The focus of predictive modeling is still the same as in regular time series, but the irregular indices can create difficulties that need to be addressed.
- **Event time series.** This is the type of time series we study. They represent discrete events happening in continuous time. Since the times of the events are also irregular, they may look similar to irregular time series. However, fundamentally they are different, because for event time series, the irregular time points themselves are the observations with randomness that we intend to model, while in irregular time series, they only serve as the indices of the data. The focus of predictive modeling is on capturing the dependencies between the events according to their times and forecasting the times of future events based on the dependencies and the history.

For event time series, the occurrences of events can depend on previous events or other context information. For example, for bike rental events, they may depend on the recent rental events, the time of the day, the day of the week, current weather condition, and so on. For medication administration events, the action of giving a specific medication may depend on the previous administration of the same or other medications, or on the patient's underlying health condition reflected by various values in the medical records. It is important for predictive models to be able to capture these dependencies. If we model these dependencies directly in continuous time, we can answer questions of all types (**Q1**, **Q2**, and **Q3**).

Despite the clear distinction, there is also a strong connection between event time series and regular time series. As we saw in Figure 1 and 2, the same data can be presented as either type of time series. The transformation between them is *time discretization*. To convert an event time series to a regular time series, we can discretize the time by dividing the time line into consecutive bins with equal length and counting the number of events in each bin. The counts will form a regular time series, or more specifically, a *count time series*. This is the second way to model event time series, which is modeling the corresponding count time series after transformation. In this case, we can answer questions of type **Q2**, and **Q3** but not type **Q1**, due to the loss of information.

Finally, we can take the abstraction in time discretization one step further. Instead of counting the number of events within each bin, we can binarize the data as an indicator of whether *any* events occurred within the bin. This is an even higher level of abstraction compared with count time series, but it can help to simplify the data, if we only care about whether the events occurred or not. In this case, we can still answer questions of type **Q3** but not type **Q1** or **Q2**.

1.1.2 Anomaly Detection in Event Time Series

Next, we highlight the questions we might wish to answer related to anomaly detection using the bike rental events as an example. Given the observations of the events for a period of time, in terms of anomaly detection, we may wish to answer different types of questions as follows.

Q4 Given that it has been 4.36 hours since the previous bike rental, is there anything abnormal?

Q5 Given the pattern of the previous bike rentals, is the last bike rental too soon that may indicate something is wrong?

Q6 Is the number of bike rentals yesterday abnormally high?

Q7 Does the high numbers of bike rentals in the last month compared to the previous months indicate something has changed?

Q8 Is it normal that there are any/no bike rentals in the last hour?

In general, anomaly detection [11, 2] aims to identify data instances that are unusual when compared to other instances in data. It has been applied in variety of areas to identify rare or novel instances or patterns, such as fraud detection [27], network intrusion detection [32], disease outbreak detection [110], and medical error detection [40].

Anomaly detection in time series has also been an important topic [36]. Here, we categorize anomalies in time series into two types:

- **Temporary outliers.** These are abnormal observations that only last for an instant, after which the time series return to normal.
- **Change-points.** These are long-term changes in the data that show up as consecutive observations that are very different from previous observations and last for a large or indefinite amount of time.

Figure 3 shows examples of both a temporary outlier and a change-point on a time series. The temporary outlier is an individual observation with abnormally low value compared with previous patterns in the data, but it does not have any effect on the observations after it, while the change-point marks the time point after which all the observations have increased values compared with before.

We can use any one of the three types of models of event time series with different levels of abstraction as the model to study anomaly detection in event time series. If we model event time series without any time discretization, we have access to the detailed information of each event and the different lengths of intervals between the events, so we can answer questions of all types (**Q4**, **Q5**, **Q6**, **Q7**, and **Q8**). If we convert event time series to count

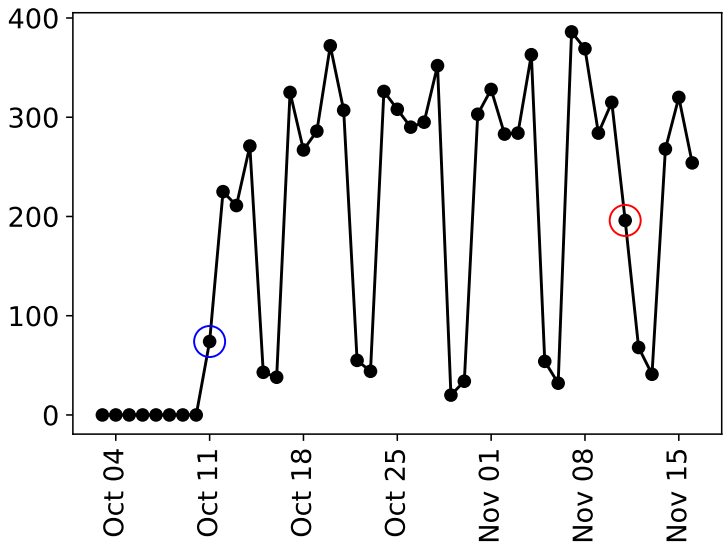


Figure 3: An example time series containing both a temporary outlier and a change-point. October 11 is a change-point. November 11 is a temporary outlier.

time series with time discretization, we lose the detailed information related to each event and the intervals between events, but we still have the numbers of the events in each fixed regular time interval, so we can still answer questions of type **Q6**, **Q7**, and **Q8** but not type **Q4** or **Q5**. Finally, if we convert event time series into time series of indicators of whether any events occurred within regular time intervals, then we can only answer questions of type **Q8**.

1.1.3 Approaches

To study the problems of predictive modeling and anomaly detection in event time series, we follow two general approaches based on the connection between event time series and count time series.

- **Discretized-time approach.** In this approach, we first convert event time series to count time series and then solve the problems in the discretized time domain.
- **Continuous-time approach.** In this approach, we directly solve the problems in the original continuous time domain.

We briefly state our motivations for following the discretized-time approach. First, regular time series have been studied more widely in the literature, so we expect to take advantage of the existing research and build our solutions on top of the existing works. Second, in practice, there could be cases when collecting the timing of each event is impossible, and only available data are the summarized counts of the events over regular period of time, so only discretized-time approach can be applied.

Meanwhile, we also have motivations for following the continuous-time approach. First, by utilizing the detailed information collected in event time series, we expect to solve the problems from a novel aspect that is not possible in the discretized-time approach, e.g., predicting the occurrences of individual events. Second, due to the relatively limited research on event time series compared to regular time series, we expect to find more open questions to answer following this approach.

1.2 Contributions

Following the discretized-time or continuous-time approach for predictive modeling or anomaly detection, we can have four different combinations. We summarize our contributions from these four aspects.

- **Modeling count time series.** Because regular time series modeling is a widely-studied problem, we focus on applying and adapting existing models of time series to the problem of anomaly detection. In Chapter 3 and Chapter 4, we try to build models that can account for properties of time series, such as seasonality, that are usually not considered for anomaly detection in existing methods, although they are common in real data and can have significant impact on the performance of anomaly detection.
- **Anomaly detection in count time series.** We study two main types of anomalies: (temporary) outliers and change-points. In Chapter 3 and Chapter 4, we develop new anomaly detection methods based on better models of the time series that account for properties of the time series such as trends and seasonality. With extensive experiments,

we show that by using these improved models we can detect temporary outliers and change-points with better performance.

- **Modeling event time series.** The existing models for event time series are limited in different ways. Some of these models are easy to apply in practice but not very flexible, while the others are more flexible but harder to apply. In Chapter 5, we develop a new model for event time series that is more flexible than the former and more applicable than the latter, and therefore combines the benefits from both types of models.
- **Anomaly detection in event time series.** Anomaly detection in continuous time is an area that has barely been studied. We attempt to solve new problems in this area with practical implications. In Chapter 6, we define two new types of anomalies that can be detected in continuous time and develop new methods to detect them. We provide theoretical justifications for the proposed methods, prove performance guarantees, and show the effectiveness of the methods using experiments. In Chapter 7, we develop methods for change-point detection in event time series using the continuous-time approach and compare the continuous-time approach with the discretized-time approach. We discover interesting theoretical properties of these two approaches and compare their performance using experiments.

We formalize the four hypotheses that we evaluate in the dissertation as follows.

H1 By using more accurate models, we can detect anomalies more accurately in time series (Chapter 3 and 4).

H2 By combining nonparametric and regressive models for point processes, we can build more flexible and applicable models for event time series (Chapter 5).

H3 There are new types of anomalies that we can detect in continuous time for event time series (Chapter 6).

H4 There are anomalies that we can detect in continuous time with better accuracy than via time discretization for event time series (Chapter 7).

2.0 Background

In this chapter, we introduce preliminary background for our problems and review related works in the literature. Overall it consists of two parts. First, we introduce and review works related to models of time series, including regular time series and event time series. Second, we introduce and review works related to anomaly detection in general and specifically in time series.

2.1 Predictive Modeling of Time Series

2.1.1 Regular Time Series Models

Although predictive modeling of regular time series is not our focus, we will see that many anomaly detection algorithms are based on specific models of the time series. Since one approach of detecting anomalies in event time series is by converting them to regular time series, these models are also related to anomaly detection in event time series. We give a brief introduction to these models in this section. For more details, Box et al. [9] and Shumway and Stoffer [96] provide good references.

2.1.1.1 ARIMA Models Autoregressive (AR) models are simple, intuitive, and popular models for time series, where the recent history of the observations are used to predict the future through regression. Let $\{x_t | t \in \mathbb{Z}\}$ denote the time series observed at discrete time t. Then an AR model of order p assumes

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + w_t, \tag{2.1}$$

where $w_t \sim N(0, \sigma_w^2)$ and ϕ_i are parameters.

Moving average (MA) models are another popular type of models using regression on the past noises/errors, instead of observations as in AR models, to predict the future. An MA model of order q assumes

$$x_t = \sum_{j=1}^{q} \theta_j w_{t-i} + w_t, \tag{2.2}$$

where $w_t \sim N(0, \sigma_w^2)$ and θ_j are parameters.

Autoregressive moving average (ARMA) models are combinations of AR models and MA models. An ARMA model of order (p, q) has the following form:

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-i} + w_t. \tag{2.3}$$

In the above model definitions, we have assumed zero mean for x_t for simplicity, but it is not hard to model nonzero means by adding another parameter to the RHS of the equations.

Although these models can represent a wide variety of time series, none of them handle common nonstationarity such as trends and seasonality. Trends are long-term changes (increase/decrease) in the mean of the observations. Seasonality is the periodic (e.g., weekly) changes in the mean. When a time series has any type of nonstationarity, the distribution of data may change over time. Nonstationarity is very common in real data due to the ever-changing nature of the real world.

Autoregressive integrated moving average (ARIMA) models extend ARMA models, such that trends and seasonality can be modeled. Essentially, we take difference of observations at different times with lags. For example, with a lag of 1, we calculate $x_t - x_{t-1}$. Then, we model the resulted differences with an ARMA model. Depending on the lag of the difference, we can model both trend and seasonality, given that we know the period of the seasonality.

However, ARIMA models (specifically differencing) are problematic for time series with outliers or change-points. For example, suppose there is an outlier at time t with an extremely large value, while all the other points are normal. Assume we are taking differences with a lag of 1. Then the point at time $t + 1$ after differencing will be extremely small and becomes an "artificial outlier". By taking multiple and/or higher order differences (i.e. compounded differences), these artificial outliers can spread even further. How to detect anomalies in nonstationary time series is a nontrivial question to answer.

2.1.1.2 State-Space Models State-space models are another type of models for time series. Different from ARIMA models, latent (or hidden) states are introduced to model the underlying dynamics of the time series. The observations are assumed to be drawn from distributions determined by the latent states. The most widely used and studied state-space models are dynamic linear models (DLMs) [37] or linear dynamical systems (LDSs) [49], where the state transition equation and the emission equation are both linear.

Specifically, let $\{y_t \in \mathbb{R}^p : t = 1, 2, \ldots\}$ be the observed time series and $\{x_t \in \mathbb{R}^d : t = 1, 2, \ldots\}$ be the latent states. A DLM is defined as

$$\begin{aligned} y_t &= F_t x_t + v_t, \quad v_t \sim N(0, V_t). \\ x_t &= G_t x_{t-1} + w_t, \quad w_t \sim N(0, W_t), \end{aligned} \tag{2.4}$$

where F_t and G_t are matrices that respectively model the emission of the observation y_t from the current hidden state x_t, and the transition of the latent states over time. Zero-mean Gaussian noises v_t and w_t are added to both equations with covariance matrices V_t and W_t. The former accounts for the fact that our observations of the time series can be noisy. The latter accounts for possible (unexpected) innovations in the state transition that are not captured by G_t.

Here, the model is defined in its most general form. However, in many cases, F_t, G_t, V_t, and W_t will be time independent, so they become F, G, V, and W. To complete the definition, we also need to define the latent state at the beginning ($t = 0$), $x_0 \sim N(m_0, C_0)$, where m_0 and C_0 are the mean and covariance of x_0 respectively. Figure 4 shows the graphical representation of a DLM with 3 time steps. The arrows in the graph indicate the dependencies between the latent states and the observations.

Researchers have extended state-space models in different ways. One common way is to allow the parameters of the model, such as F and G, which are usually fixed and learned from data, to change by switching to different values over time [30, 84, 83, 86]. The second way of extending the models is to allow non-Gaussian observations [54, 109]. The third way of extending the models is to replace the linear transition and emission equations with nonlinear functions [90, 103]. Finally, practical approaches have been developed to fill the gap for applications of state-space models, including regularization based algorithms for learning the

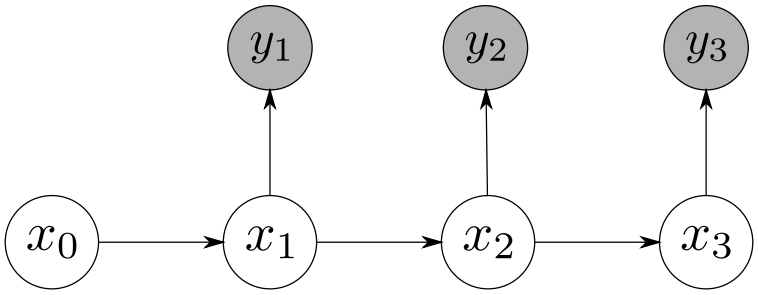


Figure 4: The graphical representation of a DLM with 3 time steps. $x_0, \ldots, x_3$ are the latent states. $y_1, \ldots, y_3$ are the observations. The arrows indicate the dependencies between the latent states and the observations.

model [69, 72], handling irregularly sampled observations [74, 70], and learning models that can adapt to time series generated from different individuals [71, 73].

2.1.1.3 Binary Event Prediction Models So far we have reviewed models for regular time series, which can be used to model event time series after converting them into count time series through time discretization. However, we can take one step further in the abstraction and convert the counts resulted from time discretization to binary indicators of whether there are any events within each time interval.

Given the simplified view from this approach, the key issue to address is how to effectively summarize the history observed in the data up to a given time point, the result of which can be fed into a model to predict whether there will be any events in the next time interval. More formally, the event prediction problem is modeled as $P(x_{t+1}|\mathcal{H}_{0:t})$, where x_{t+1} is the observation at time $t+1$ of the binary regular time series converted from the event time series, and $\mathcal{H}_{0:t}$ is a summary of past observations up to time t relevant for the prediction. These models often require summarizing the history of past observations that are defined by complex multivariate time series.

Such models have been, for example, used to support event detection and prediction in clinical time series [39, 40, 41]. Valko and Hauskrecht [104], Hauskrecht et al. [39] define a fixed set of feature templates placed on individual time series. Batal et al. [5, 6] define and

select pattern-based features summarizing the history by mining so-called recent temporal patterns. Lee and Hauskrecht [58, 59, 60] use various temporal mechanisms based on recurrent neural networks to summarize the history of observations.

2.1.2 Event Time Series Models

Event time series or event sequences (we use these two terms interchangeably) are sequences of timestamps recording events observed over time. A distinctive feature of event time series compared with regular time series is that timestamps are irregular and in a continuous domain instead of regular and in an essentially discrete domain. Another feature is that there is randomness in the times of the events, and the main goal of modeling event time series is to model the distribution of the event times taking account of the randomness.

Point processes [18, 19] are probabilistic models for points randomly distributed over a domain. Although they have also been widely used for modeling random point patterns in both spatial and temporal domains, here we focus on the cases when the domain is the time ($\mathbb{R}_+$). In these cases, they are also called temporal point processes to distinguish them from spatial point processes. We will not make this distinction and focus only on temporal point processes unless there is an ambiguity in the context.

In the most general form, an event time series will contain not only timestamps, but also additional information associated with each time. For example, we can collect data about patients' visits to a hospital. For each visit, besides its time, we can also collect the patient's gender and/or the reason for the visit. The data can be denoted as a sequence of tuples $\{(t_i, v_i)|i \in \mathbb{N}\}$, where t_i is the time of the ith point, and v_i is the corresponding information associated to the point. This additional information is called marks in the context of point processes, and this type of data (models) are sometimes called marked event time series (marked point processes) to distinguish them from time-only event time series (point processes), where $v_i = \emptyset$.

One special case of the marked event time series that is worth mentioning due to its wide availability is when the marks are discrete labels, where we can assume $v_i \in \mathbb{N}$. In this case, instead of viewing the labels as marks associated to each time, we can alternatively view the

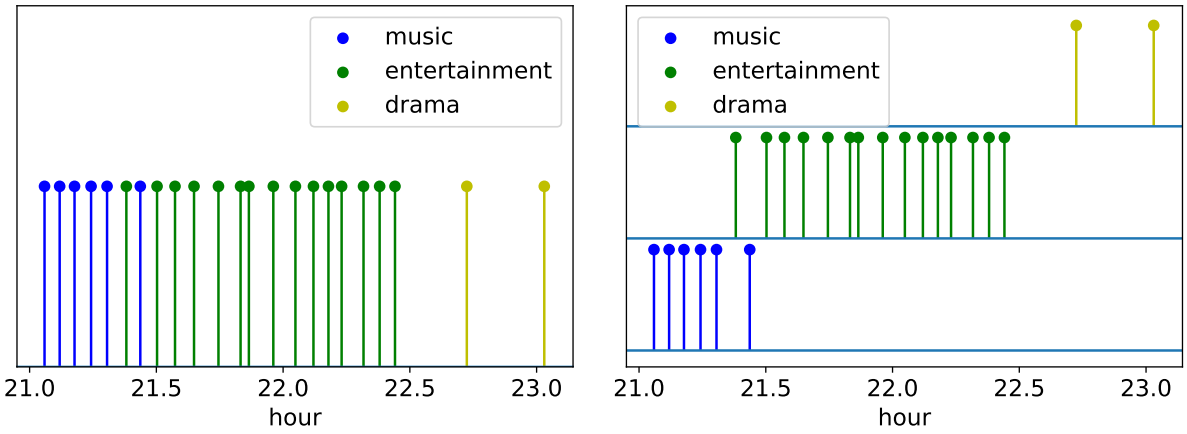


Figure 5: A marked event time series (left) and its equivalent multivariate representation (right). The data is extracted from the IPTV dataset [77] for one user on February 8, 2012 from 21:00 to 24:00. Each stem represents the event that the user starts watching a TV program of a specific type (music, entertainment, or drama).

event time series as a compound of multiple event time series, one for each label. Therefore, this specific type of data (models) are also called multivariate or multidimensional event time series (point processes).

Figure 5 shows an example of a marked event time series with its equivalent multivariate representation. It is extracted from the IPTV dataset [77] for one user. Each stem in the figure represents the event that the user starts watching a TV program. The type of the program is associated with the event as the mark (or label). If we treat each type of program as a separate sequence of events, we get multiple sequences of events, each with a single type, which form a multivariate (or multidimensional) event time series.

The key component of a temporal point process is its conditional intensity function (CIF). In fact, the CIF uniquely defines the temporal point process. The CIF specifies the instantaneous rates of the occurrences of events at any time t, given the history up to time t. Let $\lambda(t)$ denote the CIF. Then

$$\lambda(t)dt = \mathbb{E}\left[N([t, t+dt))|\mathcal{H}_{t^-}\right], \tag{2.5}$$

where $\mathcal{H}_{t^-}$ is the history of the events up to (excluding) time t, and $N(\cdot)$ counts the number of points in an interval. The intensity is assumed to be always positive. For an event time series $y = \{t_i\}_{i=1}^n$, its log-likelihood given the CIF is

$$\ln p(y|\lambda) = \sum_{i=1}^{n} \ln \lambda(t_i) - \int_{\mathcal{T}} \lambda(s) ds, \tag{2.6}$$

where $\mathcal{T}$ is the time range within which the event time series is observed.

The simplest type of point processes are homogeneous Poisson processes, where the intensity function is just a constant and does not depend on the history. That is $\lambda(t) = \lambda_0$ for all t and some $\lambda_0 > 0$. By allowing the intensity function to change over time but still independent of the history, we get inhomogeneous Poisson processes, which are generalizations over the homogeneous Poisson processes.

If we take this generalization one step further, we can assume $\lambda(t)$ depend on the history either *implicitly* or *explicitly*. The models we will discuss in the next few sections will fall into either one of these categories.

2.1.2.1 Gaussian Process Modulated Point Processes Gaussian processes (GPs) are probabilistic models for functions [92]. One way of modeling the dependencies between points in event time series is to assume the data are generated from an inhomogeneous Poisson process, and then model the intensity function of the Poisson processes as random functions drawn from a GP prior with some transformation to make sure it is positive. That is

$$\begin{aligned} f &\sim \mathcal{GP}(\mu, \Sigma), \\ \lambda(t) &= g(f(t)), \\ y &\sim \mathcal{PP}(\lambda), \end{aligned}$$

where μ and Σ are the mean and covariance functions of the GP, g is a model-specific transformation that ensures the model to be well-defined, and $y = \{t_i\}_{i=1}^n$ is the event time series we observe. When we observe a sequence of events, we can infer the posterior distribution of the intensity function given the observed data. Moreover, if we wish to make predictions, we can first predict the intensity function in the future conditioned on the data,

and then infer the values in which we are interested (e.g., expected number of events within a window) using the predicted intensity function.

This can be viewed as a latent-state model, where the latent state is the random function drawn from the GP. Given the latent state, the intensity function and therefore the distribution of the points are uniquely defined. Furthermore, given the latent state, the distributions of any nonoverlapping regions in the domain (time) are independent with each other, due to the Poisson process assumption. However, the marginal distributions (without conditioning on the latent state) are not independent.

Using the latent random function enables us to model the dependencies between the points. These models are sometimes called Cox processes or doubly-stochastic Poisson processes, due to the fact that the intensity function itself is stochastic. Adams et al. [1] propose an MCMC inference algorithm for a Sigmoidal Gaussian Cox process (SGCP), where the GP is mapped through a logistic function to the intensity function. Rao and Teh [91] propose an MCMC inference algorithm based on thinning for a generalization of SGCP, GP modulated renewal process, where the logistic mapping from the GP to the intensity function is augmented (multiplied) by a time dependent function (which used to be a constant in SGCP). Lasko [56] proposes to use the exponential transformation of a GP as the intensity function. In this way, there is no upper bound on the intensity function, so, the author argues, it can model bursty events, while SGCP cannot. However, due to this change, the thinning algorithm in the previous MCMC inference methods no longer works. The author instead proposes to use direct numerical integration in the MCMC inference. Samo and Roberts [93] improve the efficiency of inference in GP-modulated point processes with the exponential transformation by using inducing points and marginalizing out the log-intensity term in the log-likelihood at each event, while overall still using MCMC plus numerical integration.

In contrast to the previous works, where sampling algorithms are used for inference, Lloyd et al. [75] propose a variational inference algorithm for GP modulated point processes. The key difference is that the intensity function is assumed to be a *square* transformation of a GP. This enables analytical evaluation for the integration of the intensity function, which was always a problem in these models and "forced" the previous methods to use thinning or numerical integration with MCMC sampling. Kim [53] combines Markov jump processes

(MJPs) with GPs to model nonstationary (the author calls it semi-stationary) point processes. MJPs and GPs are both used as models for the intensity function in inhomogeneous Poisson processes. However, MJPs are piecewise constant with discontinuous jumps, while GPs are continuous across the whole domain without any discontinuity. By combining them, the resulted intensity function are piecewise continuous, which is more general than piecewise constant, and can have discontinuous jumps. This can be viewed as a nonstationary version of the previous GP modulated point processes. Instead of having one GP drive the point process, multiple GPs are switched on and off over time. A variational inference algorithm is developed based on [75].

All the aforementioned methods are for univariate event time series, where each sample is a single sequence of points. On the other hand, there are works addressing the problem of modeling multivariate event time series, where each sample is multiple sequences of points. As we discussed in the previous section, multivariate event time series are equivalent to marked event time series with discrete marks. A key problem here is to model the dependencies across the sequences in addition to the dependencies within the sequences. Gunter et al. [35] propose an extension of SGCP [1] to multivariate point processes. To model the dependencies between sequences, the authors propose to model the intensity function of each variate as a convolution of a set of GPs, which are shared across all variates. A tractable MCMC algorithm is developed for inference. Lloyd et al. [76] extend [75] to multivariate point processes, and call their model Latent Point Process Allocation (LPPA). In LPPA, the intensity function of each variate is assumed to be a positive weighted sum of squared GPs, and the GPs are shared across all variates. Ding et al. [20] further extend LPPA, where each intensity function is an *infinite*, instead of *finite*, weighted sum of squared GPs. The weights are assumed to be drawn from a Dirichlet process. This resolves the problem of choosing an appropriate number of hidden GPs beforehand in LPPA.

A big limitation of GP-modulated point processes is that they are learned sequence by sequence. Each sequence has its own latent state, which is independent from the other sequences. What we learn from the training data are only the hyperparameters. In the prediction stage, we still have to infer the latent state from scratch whenever we have a new sequence. This prevents training on a sample from the population and applying the learned

model on unseen examples directly. In the existing works, the models are all trained and tested on the same sequences across time (trained on the past and tested on the future). Although this is still useful, it is a significant limitation compared with the models we describe in the next section.

2.1.2.2 Hawkes Processes Another way of modeling dependencies between events is to explicitly put the dependency structure in the model. Hawkes processes [42] are the most widely used models in this category. A Hawkes process can model both *self-exciting* and *mutually exciting*. The former refers to the cases when events happened in the past can temporarily increase the rate of the events of the *same type*. The latter refers to the cases when events happened in the past can temporarily increase the rates of the events of *other types*.

The CIF of a Hawkes process is a sum of the influences from the points in the past. Let (t_i, u_i) denote the time and type (mark) of the ith event. Then the CIF for events of type u_i is

$$\lambda_{u_i}(t) = \mu_{u_i} + \sum_{t_j<t} \phi_{u_i u_j}(t - t_j), \tag{2.7}$$

where $\mu_{u_i} > 0$ is the baseline intensity, and each $\phi(\cdot) \geq 0$ is called a triggering kernel. A concrete example would be $\phi_{u_i u_j}(t) = A_{u_i u_j} \exp(-\beta t)$, where $A_{u_i u_j} \geq 0$, and $\beta > 0$. The triggering kernel captures the influences of the past events of type u_j to the future events of type u_i. The significance of the influence is controlled by $A_{u_i u_j}$, while how long it lasts depends on β. As the signs of these parameters indicate, the influences are assumed to be excitatory and temporary, that is the occurrences of events can increase the rates of other events for a limited amount of time.

Hawkes processes have been used for modeling the interactions between entities such as humans, countries, or websites. Blundell et al. [7] models reciprocating relationships by combining a Chinese restaurant process (CRP) with a multivariate Hawkes process, where each variate corresponds to actions from one group to the other, and the individuals are assigned to groups through the CRP. The intensity function is explicitly assumed to reciprocating, that is actions from group A to B are excited by actions from group B to A.

Zhou et al. [119] assume sparsity and low-rank in the social interactions and therefore in the *infectivity matrix* of the Hawkes process, which models the influences between the entities. An alternating direction method of multipliers (ADMM) is developed to solve the regularized maximum likelihood estimation problem.

One drawback of Hawkes processes is that its triggering kernels take a parametric form, which needs to be specified using prior knowledge. Researchers have made much effort in extending Hawkes processes to learn the triggering kernels nonparametrically. Zhou et al. [118] propose to learn the triggering kernels of a Hawkes process nonparametrically. Each triggering kernel is assumed to be a linear combination of a set of base kernels, and the base kernels are learned nonparametrically by discretization. They solve a penalized maximum likelihood optimization problem by transforming an infinite dimensional functional optimization problem to an ordinary differential equation problem. Eichler et al. [25] also use discretization of the triggering kernels, but their model learns the triggering kernels by solving a least-square problem. Xu et al. [114] propose to use a set of basis functions to approximate the triggering kernel nonparametrically, which does not require discretization, and their method shows better performance than the previous works.

There are also works trying to extend Hawkes processes to be more flexible other than learning nonparametric triggering kernels. Du et al. [23] propose Dirichlet-Hawkes process (DHP), where the *parameters* of the kernels are generated nonparametrically through a Dirichlet process for each point. Lee et al. [61] use a stochastic process to model the evolution of the excitations (weights) in a Hawkes process, so the weights become random variables instead of constant parameters. Restricting the kernels to the same exponential kernel (sharing one parameter) for efficiency, they propose a simulation algorithm and an inference algorithm based on a hybrid of MCMC algorithms. Wang et al. [106] define an isotonic Hawkes process, where the intensity function of the original Hawkes process is transformed through a monotonic discretized nonparametric link function. They assume the link function is piecewise-constant non-decreasing, and the jumps are only at the intensities of the observed points, so the integral can be computed analytically. The optimization problem is formed by moment matching instead of traditional maximum likelihood estimation and can be solved with the proposed algorithm having a proved convergence rate.

Although the above works have significantly extended the flexibility of Hawkes processes, these models still have some limitations preventing them from successfully capturing some common phenomena in real world. For example, in most of the Hawkes process variants, events in the past can only increase the rates of events happening (with or without delays). Although some of them can also model decreases, none of them, except for [25], can model a mix of both (e.g. an increase following a decrease). However, this is a common phenomena in real world. For example, giving a medication to a patient will increase the overall chance of giving the same medication due to the fact that the patient's medical condition can last for a long time and therefore repeated treatments are needed. But right after a medication being given, the chance of giving the same medication will decrease for a while (e.g., 24 hours), since there is a limit of how much medication can be taken within a period of time. Similar behaviors are also observed in our neural systems [25], where there is "a self-inhibition after the firing of a neuron".

2.1.2.3 Other Point Process Models Neural networks have also been studied as a way of modeling the intensity functions of point processes. Du et al. [24] develop a recurrent neural network (RNN) model for event time series. Event labels and inter-event times are used as the input at each step, and the logarithm of the intensity function between two consecutive events are assumed to be a linear function. The likelihood function has a closed form and is maximized through gradient ascent, although for prediction, numerical integration has to be invoked for the calculation of expected time. Mei and Eisner [80] combine Hawkes processes with long short term memory (LSTM [43]) and develop a continuous-time LSTM, where the memory cell has an exponential decay (with learnable parameters) between two consecutive events, while in a traditional LSTM, the memory cell is simply preserved between two consecutive inputs. The softplus transformation of the output is used as the intensity at each time t *continuously*, i.e., it is defined even between events when no input is provided. Due to the complex nonlinear transformations, the integral cannot be evaluated in closed form, and the authors resort to Monte-Carlo sampling for both training and prediction. Xiao et al. [112] define a Wasserstein distance for point processes and combine it with Generative Adversarial Networks [33] to train generative models. However, their model is limited to

univariate point processes.

Another type of point-process models are based on featurization of the history. Gunawardana et al. [34] propose piecewise-constant intensity models (PCIMs), assuming the intensity function is a piecewise-constant function of the past events. This function is modeled as a decision tree mapping features extracted with window-based functions from the past events (e.g., events with label u occurred more than τ times within a window of size w) to a constant intensity. The parameters and the structure of the tree can be learned. An importance sampling algorithm is developed for predicting the probability of a sequence of events happening in the future within a sequence of time intervals. Weiss and Page [108] apply the multiplicative-forest technique [107] to PCIMs, assuming the decision tree can be factorized into multiple trees with the final intensity being the product of the outputs of all the trees. Lian et al. [62] also assume a piecewise-constant intensity function, but extends it to multitask problems using a hierarchical model.

2.2 Anomaly Detection in Time Series

Anomaly detection [11] aims to identify data instances that are unusual when compared to other instances in data. It has been widely studied on both independent and identically distributed data and time series. We first briefly review general anomaly detection.

In general, anomalies are unusual instances or patterns in the data. The specific definition of anomalies can change slightly depending on the specific applications. In many cases, anomalies are data instances that deviate significantly from the majority of the data such that they might be generated differently than the normal data, in which case they may also be called *outliers* [44, 2]. In other cases, anomalies are new or unknown instances or patterns in the data that have not been seen before, in which case they may also be called *novelties* [78, 79]. Anomaly detection has diverse applications in different domains, such as fraud detection [27], network intrusion detection [32], disease outbreak detection [110], and medical error detection [40, 41, 39].

Basic anomaly detection methods aim to identify unusual instances among all the data.

Statistical anomaly detection methods [78] address the problem by defining and modeling the distribution of the normal instances $p(x)$, where x represents an instance, and identifying instances that fall into a low probability region. However, in some cases, whether an instance is an anomaly or not may depend on additional information provided as context. Conditional (or contextual) anomaly detection [98, 38, 39, 105] aims to identify unusual responses given the context. To illustrate conditional anomaly detection, consider a medication for which the dosage varies depending on the patient age. A low dosage may look like an anomaly with respect to the whole population, but it could be perfectly normal for a child. Similarly, a dosage that is normal for an adult should be considered as anomalous for a child. Statistically, conditional anomaly detection tries to identify instances that come from regions with a low conditional probability $p(y|z)$, where y is the response, and z defines the context. Depending on the form of y, conditional anomaly detection can be either univariate (the response is defined by one scalar random variable) or multivariate/multidimensional [45, 46, 47] (the response is defined by multiple, possibly dependent, random variables). Gaussian mixture models [98] or multilabel classification models [45] have been used to define the model $p(y|z)$.

2.2.1 Anomaly Detection in Regular Time Series

As event time series can be converted to regular time series through time discretization, we can solve some event time series anomaly detection problems based on anomaly detection in regular time series. Therefore, we review existing works related to anomaly detection in regular time series. Specifically, we focus on outlier detection and change-point detection in time series.

Outlier detection is treated as synonyms for anomaly detection in many existing works, where outliers are the data points that show unusual behaviors or patterns. On the other hand, change-points are the points when the behaviors of the data change, supposedly caused by the change in the underlying process generating the data. They can be viewed as a specific type of anomalies but have also received research on their own. Most outliers are usually temporary instead permanent. That is, the behavior of the time series will return to normal after a limited amount of time of being abnormal. In contrast, change-points mark

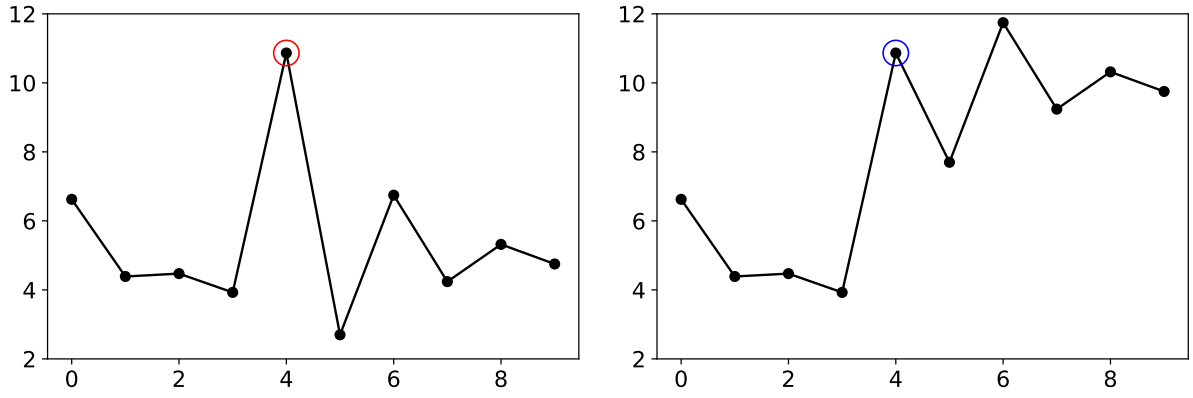


Figure 6: A simulated time series with a temporary outlier (left). A simulated time series with a change point (right). Both the temporary outlier (red circle) and the change-point (blue circle) occur at time 4.

the changes in the time series that can last for indefinite amount of time. Figure 6 shows examples for a temporary outlier and a change-point.

2.2.1.1 Outlier Detection in Regular Time Series In this section, we review existing works for outlier detection in regular time series. Over the years, different types of outliers have been defined and studied in the context of different time series models (e.g., AR and MA models) [29, 8, 101, 12]. The essential method of these works is the likelihood ratio test. Specifically, a model (e.g., AR) is assumed for the time series without outliers. Then, for a specific type of outliers, an alternative model with an outlier inserted is defined. These two models form two hypotheses

$$H_0 : \text{there is no outlier;} \qquad H_1 : \text{there is an outlier.}$$

Then a likelihood ratio test statistic [10] can be derived based on the likelihood of these two models to test whether there is an outlier.

Tsay [102] provides a good summary of previously studied outliers. Furthermore, the author extends previous work [12] and proposes two procedures to handle outliers in the context of ARMA models. The procedures are iterative and cycles among parameter estimation,

outlier detection, and outlier removal. Through cycles, the most significant outlier is removed one by one. Chen and Liu [13] propose to jointly estimate multiple outlier effects and model parameters, instead of removing outliers one by one as in the previous work, although their procedure still uses the previous procedure in the initial stage.

There are some common traits shared by the above methods. First, they are all retrospective, i.e., they assume the whole time series (including "future" for the outliers) is available, and they look back to find all outliers in the past. In reality, *online* detection, where we wish to detect outliers in the newly observed data as soon as they arrive (without access to the future data), might be of more interest, since it can be applied to monitoring systems to alert on outliers in real time. Second, most of the above methods assume the time series follow ARMA models (or their subsets). However, ARMA models cannot deal with some of the nonstationarity that is very common in practice, such as seasonality and trends. We note that both seasonality and trends can be addressed by adding (seasonal) differencing to the models, resulting in the (seasonal) autoregressive integrated moving average ((S)ARIMA) models [96, 9]. However, normal points differenced with outliers can become "artificial outliers" and result in false alarms, unless they are properly accounted for. Therefore, these methods by themselves are not suitable for online outlier detection in nonstationary time series.

More recently, Yamanishi and Takeuchi [116] develop an online algorithm for nonstationary time series. The authors assume the time series follow the AR models and introduce a sequential discounting algorithm to estimate its parameters and to make inference. However, as they assume AR, which is a subset of ARMA, their method suffers from some of the same problems as the above methods (e.g., not able to model time series with seasonality). Laptev et al. [55] propose an outlier detection framework for time series data that allows one to exclude outliers that may be explained by contextual variables. This is done by defining rules on these variables. For example, one may define a rule checking whether a day is a holiday and exclude all these days from consideration. A limitation of this approach is that it prevents us from detecting outliers on holidays that differ from typical holiday patterns. Another limitation is that building these rules requires human knowledge. Using statistical machine learning to learn probabilistic models conditioned on contextual variables from the data is a potential improvement, since we can then make decisions using "soft" probabilistic

scores instead of "hard" rules, and we only need to provide a list of contextual variables that *might* be useful and let the data tell us which are more useful than the others through learning.

2.2.1.2 Change-Point Detection in Regular Time Series In the statistics community, most change-point detection methods, similar to outlier detection methods, also focus on retrospective analysis (i.e., offline detection), where the complete data is available for analysis, and the methods try to find changes in the past. Both parametric (e.g., [94]) and nonparametric (e.g., [87]) methods have been developed. Because the complete data is available, it is common to have multiple change-points in one time series. For detecting multiple change-points, much work has been done to improve the computational efficiency (e.g., [52], [31]).

Besides the above works that focus on offline detection, there is some related research on online detection (e.g., CUSUM [85]). These methods perform tests in a sequential manner such that a change is detected as soon as possible. However, these methods are designed to detect changes with respect to a reference value provided beforehand. In cases when the data are nonstationary, it is hard to decide such a value. Furthermore, although they are able to output a score at each time step, every time a change is detected, i.e., the score is above/below a pre-specified threshold, the procedure (including the score) needs to be reset. Therefore, a different threshold on the score can drastically change the operation of the procedure, since it determines when the resets should happen.

More importantly, all the above methods, online or offline, assume the data are independent and identically distributed (IID). The assumption does not hold for time series in general (e.g., a time series generated from a simple AR(1) model). Even worse, if the data have nonstationarity, such as trends or seasonality, the method can get many false positives and/or false negatives due to the IID assumption.

In the data mining community, researchers have made efforts to address online change-point detection in time series. Yamanishi and Takeuchi [116] propose a unified framework for detecting both temporary outliers and change-points. However, their method is based on AR models, which limits the applicability of the method to many nonstationary time

series (e.g., time series with seasonality) in practice. Kawahara and Sugiyama [50] solve a change-point detection problem that is different from the traditional setting. Their goal is to detect whether there is a change between two consecutive time intervals. Within each time interval, samples are formed by sliding a small window and extracting the subsequences. Then the likelihood ratio between the null hypothesis (no change) and the alternative hypothesis is modeled as a linear combination of kernels defined on the subsequences. Besides solving a different problem, where the change-point is *fixed*, a drawback of their method is that the time intervals both before and after the change-point must be long enough to draw enough samples. Therefore, the delay of the detection is always bounded below by a large number, which is not preferable for online detection.

2.2.2 Anomaly Detection in Event Time Series

The main focus of research in event time series has been centered around the development of more flexible models to better fit the data and make more accurate predictions. Anomaly detection in event time series has not received much attention. A relatively related line of works are about learning from noisy data. Recently researchers have started to develop methods to deal with noisy data, such as incomplete data [115, 95, 81] and desynchronized data [100]. They assume the data has been corrupted by some source (e.g., censoring or noise), and the goal is to recover the original data and/or learn a model nonetheless.

In some cases, anomaly detection may be related to learning from missing data, but we note that the goals of anomaly detection and learning from missing data are completely different. For anomaly detection, the goal is to detect these outliers as accurately as possible, i.e., to distinguish them from normal data. For learning from missing data, the goal is to learn an accurate model of the data without being affected by the noisy or missing data much. Also, the settings in these two problems can be different. In anomaly detection, it is most interesting and common to have an online setting, where we only have access to the history not the the future when we try to decide whether there is any anomaly at the current time. In contrast, for learning from noisy or missing data, it is common to have an offline setting, where the whole sequence of each event time series is available for us to infer the

missing data and fit a model.

Although anomaly detection in event time series has barely been researched, we note that a good probabilistic model lays the foundation for a good method to detect anomalies in many cases, since we can use models trained on normal data to evaluate how "normal" or equivalently "abnormal" new observations are. However, the exact form of an inference algorithm making use of such a model is still unclear, and even a clear definition of anomalies in a continuous time domain is lacking. Overall, there are still open questions related to detecting anomalies in event time series in continuous time, which we try to resolve in the later chapters.

3.0 Outlier Detection in Time Series

In this chapter, we address the problem of detecting outliers in regular time series. The challenges here include: (1) The time series are nonstationary, meaning its distribution changes over time. (2) There are often contextual variables (e.g., whether the day of the observation is a holiday) that can explain some of the abnormal behaviors in the time series. We would rather not treat these points as outliers once we have an explanation from the contextual variables. (3) The opposite of (2) can also happen. Some points seem normal by themselves, but when conditioned on the contextual variables, they may become very atypical, and therefore should be detected as outliers. We try to address these challenges by combining several ideas and techniques in statistics and machine learning. Part of this work has been published at FLAIRS 2017 [66].

3.1 Method

In this section, we introduce our method. Let the time series be $y = \{y_t \in \mathbb{R} : t = 1, 2, \ldots\}$ with context variables $x = \{x_t \in \mathbb{R}^p : t = 1, 2, \ldots\}$. Our goal is to compute an outlier score $v_t \in \mathbb{R}$ at each t based on the data available at t. Our method consists of two layers. In the first layer, we remove nonstationarity and temporal dependencies from the data and derive the local deviation scores. In the second layer, we model the local deviation scores conditioned on the context variables by Bayesian linear regression. We adopt Bayesian inference because it supports more robust online learning by allowing us to add uncertainty to the model through priors. This is important, because typically context variable observations for learning the second layer model are scarce at the beginning (e.g. the number of observed holidays is small). Adding uncertainty can reduce false alarms caused by high variance in the estimated parameters.

3.1.1 Variance Stabilization

Count data usually have heteroscedasticity, i.e., the variance changes with the mean. Therefore, we apply the square-root transformation ($f(x) = \sqrt{x + 0.5}$) to stabilize the variance [4], which is commonly used for Poisson distribution.

3.1.2 Seasonal-Trend Decomposition with LOESS

We introduce the Seasonal-Trend decomposition with LOESS (STL) [15] that is used as a building block for our first-layer model. STL is a nonparametric decomposition algorithm using locally weighted regression (LOESS) [16, 17]. Given a set of points $\{(x_i, y_i) : i = 1, \ldots, n\}$, LOESS fits a smoothed curve $y = g(x)$. For any x, to compute $g(x)$, it fits a d-degree polynomial to $\{(x_i, y_i)\}$ weighted by $v_i(x) = W\left(\frac{|x_i - x|}{\lambda_q(x)}\right)$, where $W(u) = (1 - u^3)^3$, if $u \in [0, 1]$, and 0 otherwise. $\lambda_q(x)$ is the distance between x and its q-th nearest neighbor in $\{x_i\}$. If $q > n$, it is $\lambda_n(x)\frac{q}{n}$.

The main steps of STL are as follows. To separate out seasonal signal, STL fits a curve to each subseries that consists of the points in the same phase of the cycles in the time series. After removing the seasonal signal, it fits another curve to all the points consecutively to get the trend. The residuals after further removing the trend are called remainders.

It is worth noting that STL is a robust algorithm. It deals with outliers by down-weighting them and iterating the procedure. The bisquare weight function, $B(u) = (1 - u^2)^2$, if $u \in [0, 1]$, and 0 otherwise, is used for this purpose, where u is the normalized remainder for each point. Figure 7 shows an example of STL.

3.1.3 First-Layer Model

The first-layer model takes the input time series, y_t, and outputs a local deviation score, z_t, at each time t. Since the time series are usually nonstationary and may even have structural changes (nonstationarity other than seasonality and trends), we use a sliding window to restrict the time span considered, and assume that the time series in an appropriate-sized window does not have structural changes. Let $n_{(p)}$ be the period of y. The window size u

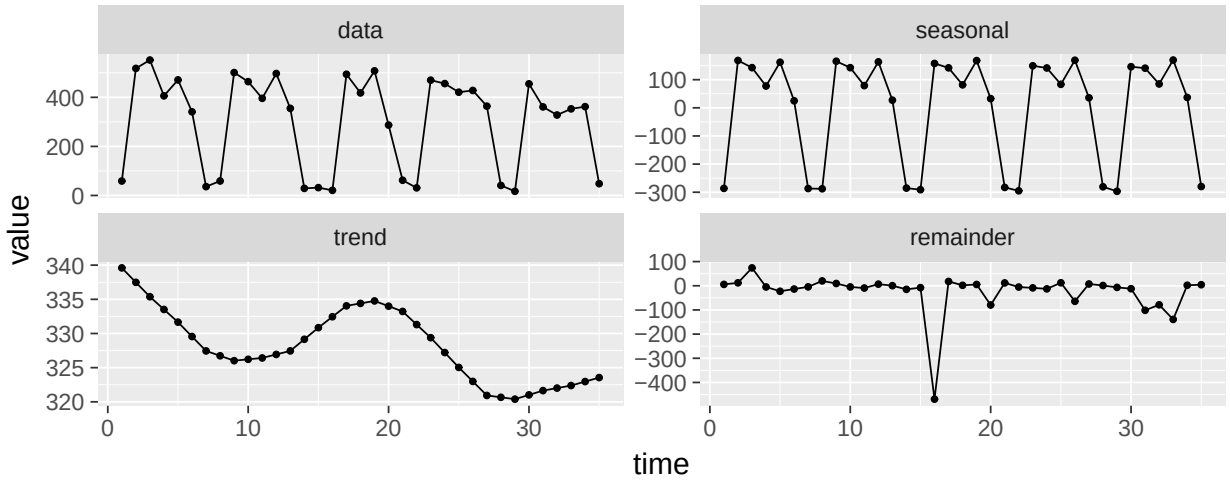


Figure 7: Seasonal-Trend decomposition of a time series. The top-right graph shows the original data, which has a strong (weekly) seasonality. The following graphs show the seasonal, trend, and remainder signals decomposed from the original time series. Notice the point at time 16 is an outlier.

cannot be too small compared to $n_{(p)}$ because STL needs enough cycles of data for smoothing. But also it cannot be too large due to nonstationarity. We found $5n_{(p)}$ is good to be used as a default value.

At time t, we denote the local time series of length u in the sliding window as

$$y_{u(t)} = \{y_{(t-u+1)}, y_{(t-u+2)}, \ldots, y_t\}.$$

STL is applied to decompose it into trend, seasonal, and remainder. On the remainder, $r(y_{u(t)})$, we calculate the deviation, z_t, of the last point, $r(y_t)$, from the population

$$z_t = \frac{r(y_t) - \hat{\mu}_t}{\hat{\sigma}_t}, \tag{3.1}$$

where $\hat{\mu}_t$ and $\hat{\sigma}_t$ are estimates of the population mean and standard deviation. Here we use the common choices: the sample mean and the sample standard deviation.

By keeping sliding the window as new data arrive, we get a sequence of local deviation scores, z, as the output of the first-layer model.

3.1.4 Second-Layer Model

The second-layer model takes the output of the first-layer model, z_t, and a set of contextual variables, x_t, as input, at each time t, and outputs a final outlier score, v_t. We adopt a Bayesian approach to model z_t given x_t. Specifically, we assume the following linear model

$$z_t|w, \beta, x_t \sim N(x_t^T w, \beta^{-1}).$$

That is, given w, β, and x_t, z_t follows a normal distribution. For the prior distribution of (w, β), we use the conjugate prior, which is a normal-Gamma distribution

$$w, \beta \sim N(w|m_0, \beta^{-1}S_0)Gam(\beta|a_0, b_0),$$

where we use the following parameterization for the probability density function (PDF) of the Gamma distribution

$$f(\beta|a, b) = \frac{b^a}{\Gamma(a)}\beta^{a-1}e^{-b\beta}.$$

Let $D_t = \{(z_1, x_1), (z_2, x_2), \ldots, (z_t, x_t)\}$ denote the data we observe so far at time t. When we observe a new sample (z_{t+1}, x_{t+1}), the posterior distribution for (w, β) is again normal-Gamma with recursively updated parameters

$$w, \beta|D_t, z_{t+1}, x_{t+1} \sim N(w|m_{t+1}, \beta^{-1}S_{t+1})Gam(\beta|a_{t+1}, b_{t+1}),$$

where

$$\begin{aligned} S_{t+1}^{-1} &= S_t^{-1} + x_{t+1}x_{t+1}^T, \\ m_{t+1} &= S_{t+1}(S_t^{-1}m_t + z_{t+1}x_{t+1}), \quad a_{t+1} = a_t + \frac{1}{2}, \\ b_{t+1} &= b_t + \frac{1}{2}(z_{t+1}^2 - m_{t+1}^T S_{t+1}^{-1} m_{t+1} + m_t^T S_t^{-1} m_t). \end{aligned} \tag{3.2}$$

The predictive distribution for z, given D_t and the corresponding context variable x, is a Student's t-distribution with location and scale

$$z|D_t, x \sim St(z|\mu, \sigma^2, \nu),$$

where the PDF of the distribution is

$$f(z|\mu, \sigma^2, \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu\sigma^2}}\left(1 + \frac{1}{\nu}\frac{(z-\mu)^2}{\sigma^2}\right)^{-\frac{\nu+1}{2}},$$

and

$$\nu = 2a_t,\ \mu = x^T m_t,\ \sigma^2 = \frac{b_t}{a_t}(1 + x^T S_t x). \tag{3.3}$$

We define the outlier score for z_{t+1}, given D_t and x_{t+1}, as

$$v_{t+1} = 1 - p_{t+1} = 1 - P\left(|T_\nu| > \frac{|z_{t+1} - \mu|}{\sigma}\right), \tag{3.4}$$

where T_ν follows the standard t-distribution with ν degree(s) of freedom, and p_{t+1} is the probability of z_{t+1} taking a more extreme value than the current value.

In summary, for $t = 0, 1, \ldots$, given (z_{t+1}, x_{t+1}), we compute the outlier score as in (3.3) and (3.4). Then, we update the distribution of the parameters as in (3.2).

3.2 Experiments

3.2.1 Datasets

We evaluate our method on time series data from three different domains.

Bike data consists of the time series (of length 733) that record the daily bike trip counts taken in San Francisco Bay Area through the bike share system from August 2013 to August 2015 [1]. Additional context variables available for the count data are holiday indicators and weather data. The weather data include precipitation, cloud cover, wind direction, mean temperature, mean dew point, mean humidity, mean sea level pressure, mean visibility, and mean wind speed. For temperature, we perform a preprocessing that transforms the value into the absolute value of the local deviation (similar to (3.1) with absolute value), because we expect both very high and very low temperatures to have an impact the number of bike trips. Outliers detected in such a time series may reflect various unaccounted events influencing the number of bike rentals, including unexpected closures due to malfunctions of the rental system.

CDS data consists of daily rule firing counts of a clinical decision support (CDS) system in a large teaching hospital [111]. The rules in the CDS are used to either alert on some

[1] `https://www.kaggle.com/benhamner/sf-bay-area-bike-share`

adverse conditions or recommend certain actions (such as vaccinations). The data include time series for 111 such rules, and each time series is of length 1187. Additional context variables collected are holiday indicators and the number of electronic health records (EHR) opened. Both are believed to influence the rule firings. Holidays may reduce the number of visits, and the number of EHR opened may give a rough estimate of the number of patients potentially screened by the rules during that day. Outliers may reflect the different events influencing the rules such as the beginning of the flu season, or CDS system malfunctions that may lead to rule silencing or aberrant rule firings.

Traffic data consists of time series of vehicular traffic volume measurements collected by sensors placed on major highways in Pittsburgh area [97]. The time series we use here are sampled at a fixed time across days for a year. We use data from two such sensors. The context variable available for these data is holiday indicator. Outliers in the time series may indicate traffic accidents, road repairs, severe weather patterns, or events such as concerts that lead to the surge in the traffic.

3.2.2 Experiment Setup

Since there are no outliers marked for our data, we test the performance of the detection methods on simulated outliers that correspond to randomly introduced changes in the original signal. More specifically, outliers are injected into the time series by randomly sampling a small percentage p of points and changing the value by a specified size δ as $y_i = y_i \cdot \delta$ for each point y_i. The values are rounded to the closest integers, so they are still counts. We use multiplicative change instead of additive, because the data show *heteroscedasticity* (the variance increases as the mean increases). We set $p = \{0.01, 0.05, 0.1\}$ and $\delta = \{2/1, 3/2, 6/5, 5/6, 2/3, 1/2\}$ respectively to see the influence of different settings on the performance. We consider the injected outliers as the ground-truth outliers when evaluating the performance.

3.2.3 Methods

We compare our method with a random baseline and baseline methods based on the widely-used probabilistic model for time series, (S)ARIMA [96, 9]:

- RND - detects outliers randomly.
- SARI - ARIMA$(1,1,0)\times(1,1,0)_7$, SARIMA with a weekly period, (seasonal) differencing, and (seasonal) order 1 autoregressive term.
- SIMA - ARIMA$(0,1,1)\times(0,1,1)_7$, SARIMA with a weekly period, (seasonal) differencing, and (seasonal) order 1 moving-average term.
- SARIMA - ARIMA$(1,1,1)\times(1,1,1)_7$, SARIMA combining the above two.

For all the ARIMA based methods we also use a sliding window. We estimate the parameters from the past points and make a prediction for the latest point. The outlier scores are derived similarly as in (4.16). We compare the following variants of our method:

- ND - our first-layer model, using the absolute value of the output as outlier scores.
- TL1 - our two-layer model using holiday information as a context variable.
- TL2 - our two-layer model using holiday and additional information (if available) as context variables.

We use R [89] for all the experiments. All methods compared use a sliding window of size 35 ($5n_{(p)}$). The hyperparameters for the two-layer method are $m_0 = 0$, $S_0 = I$, $a_0 = 1$, and $b_0 = 100$, where I is the identity matrix. They are not tuned, but we intentionally set the prior variance to make the model uncertain when the data are still scarce, so it does not raise many false alarms at the beginning. We add a bias term for the regression. For STL, we set the seasonal smoothing window size, $n_{(s)} = 7$, which is the smallest reasonable value according to [15], and $n_{(p)} = 7$, for the weekly periodicity, and use recommended values for the other parameters.

3.2.4 Evaluation

We use precision-alert-rate (PAR) curves to evaluate the methods [41]. Outlier detection methods are usually applied in monitoring and alerting systems, where the alert rate needs to be controlled by setting a threshold for the outlier score. The precision for a given alert rate is the most important factor in evaluating the performance, because whether it is high or low decides whether the system is useful or annoying, even harmful [41]. If the alert rate and the precision are not well-controlled, it may lead to so-called *alert fatigue* [57, 26], that

is users stop responding to the alerts due to their ineffectiveness. Since the probability of getting an outlier is assumed to be low by definition, alert rates cannot be set to be high in reality. We do not use precision-recall (PR) curves to evaluate the methods, because in reality, it is usually very hard to get all the outliers without causing alert fatigue. People instead control the alert rate while maintaining good precision.

3.2.5 Results

Figure 8 shows the PAR curves for Bike data with different outlier rates, p, and different outlier sizes (folds of changes), δ, leading to 18 data sets. The results are organized in a grid with different folds of changes in rows, and different outlier rates in columns. The maximum alert rate is kept at 0.1 (10% of data). We show the precision at different alert rates. The results show that it is easier to detect stronger outliers (corresponding to a larger fold of change), which is expected, since they are more likely to rise above the natural noise in the data. If the outlier signal is very weak, it may fall into the natural noise level, which is reflected by the PAR curves approaching RND. Also, as expected, the precision generally is higher, when more outliers are injected in the data.

Comparing the detection methods tested, we see the two versions of our two-layer method outperform other methods with the margin increasing for stronger and more frequent outliers. To make the comparison in different settings easier, we calculate the areas under the PAR curves (AUC-PAR). To make them comparable for different outlier rates, we normalize the alert rate relative to the outlier rates. That is, we calculate the precisions at alert rates corresponding to α times the outlier rate p, where $\alpha \in [0, 1]$, and normalize the AUC to be in $[0, 1]$. Table 1 shows the AUC-PAR for Bike data. Similar to the results in Figure 8, our two-layer methods are the best performing methods across a wide range of outlier sizes (folds) and rates.

We have performed the same experiments on CDS and Traffic data. We show the AUC-PAR results in Table 2 and 3 respectively. We note that for these two data sets we have multiple time series, so we report the averaged results. That is, given an alert rate, we average the precision over all time series. Once again the results show that our two-layer method

outperforms the baselines.

By comparing the results across different data sets, we notice that the quality of the detection may vary widely. This is due to the properties of the original time series. For example, while Bike data is relatively clean, Traffic and especially CDS data have much more noise and irregularities, that are detected as outliers. Hence the precision calculated based on injected outliers gets smaller.

Comparing ND with ARIMA based methods, we notice that ND performs either close to or better than the others in almost all the experiments. We think the main reason is that ND accounts for seasonality without differencing, so it does not "pollute" normal points like ARIMA based methods.

Comparing TL1 with ND and ARIMA based methods, we see an advantage in most cases. This confirms our assumption that whether the day is a holiday has a significant influence on the value observed on that day. TL1 makes use of that information to explain some of the "outliers" in the data. This can largely reduce the number of false alarms and therefore increase the precision.

Comparing TL2 with TL1, TL2 dominates TL1 in almost all cases. This proves the usefulness of additional information (EHR counts for CDS data and weather for Bike data), and demonstrates the flexibility of our method. Whenever there is new potentially useful information, we can add it as new context variable(s) to improve the performance. In reality, it is hard to tell beforehand which context variables will be helpful for detecting outliers. For our method, we can just add all the variables that might be helpful and have the model learn which are. This, we think, is a big advantage over a rule-based model, which needs expert knowledge and/or trial-and-error to find out which variables are useful and to define correct rules to filter out false alarms.

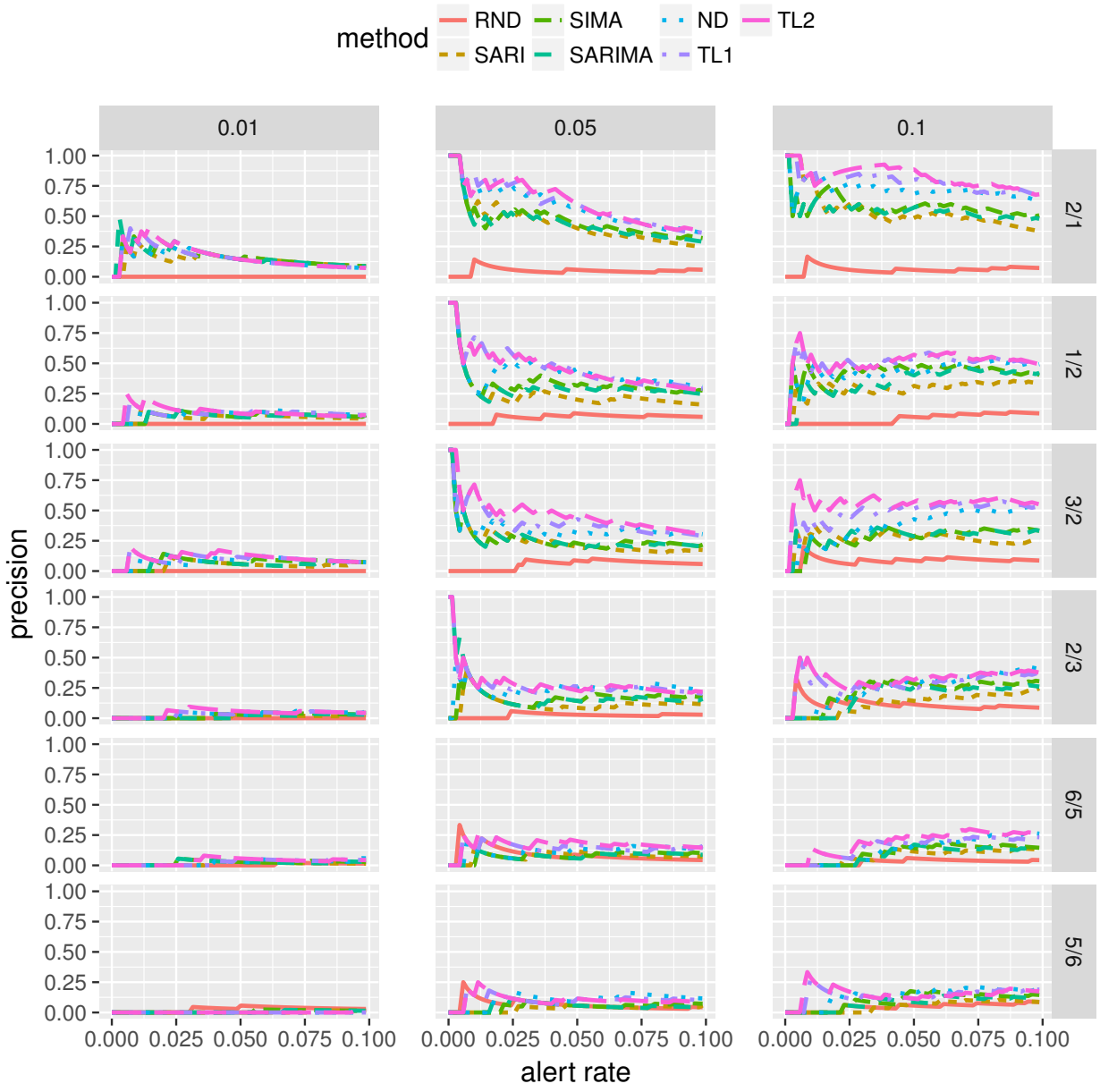


Figure 8: PAR curves for Bike data. Each column has a different rate for injection of outliers, indicated by the labels at the top. Each row has a different size (fold of change) for outliers, indicated by the labels on the right.

Table 1: AUC-PAR for Bike data.

rate	fold	RND	SARI	SIMA	SARIMA	ND	TL1	TL2
0.01	2/1	0.00	0.09	0.16	**0.24**	0.14	0.19	0.16
0.01	1/2	0.00	0.00	0.00	0.00	0.00	0.05	**0.09**
0.01	3/2	0.00	0.00	0.00	0.00	0.00	**0.05**	**0.05**
0.01	2/3	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
0.01	6/5	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
0.01	5/6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
0.05	2/1	0.05	0.56	0.56	0.55	0.71	0.75	**0.77**
0.05	1/2	0.04	0.31	0.38	0.35	0.48	**0.58**	0.57
0.05	3/2	0.03	0.33	0.30	0.31	0.38	0.47	**0.55**
0.05	2/3	0.02	0.13	0.16	0.23	0.24	0.31	**0.32**
0.05	6/5	0.12	0.07	0.06	0.06	0.10	0.15	**0.17**
0.05	5/6	0.09	0.02	0.04	0.05	0.08	0.10	**0.11**
0.1	2/1	0.06	0.54	0.58	0.55	0.72	0.78	**0.82**
0.1	1/2	0.04	0.29	0.42	0.36	0.44	0.51	**0.52**
0.1	3/2	0.09	0.27	0.27	0.29	0.38	0.49	**0.56**
0.1	2/3	0.11	0.12	0.22	0.19	0.25	0.29	**0.32**
0.1	6/5	0.03	0.08	0.10	0.09	0.13	0.16	**0.20**
0.1	5/6	0.05	0.05	0.10	0.08	0.14	0.14	**0.15**

Table 2: AUC-PAR for CDS data.

rate	fold	RND	SARI	SIMA	SARIMA	ND	TL1	TL2
0.01	2/1	0.00	0.06	0.05	0.05	0.03	0.07	**0.12**
0.01	1/2	0.01	0.02	0.02	0.02	0.03	0.03	**0.05**
0.01	3/2	0.00	0.02	0.02	0.02	0.02	0.02	**0.03**
0.01	2/3	0.00	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
0.01	6/5	0.01	0.01	0.01	0.01	**0.02**	**0.02**	0.01
0.01	5/6	0.01	0.01	0.01	0.01	**0.02**	**0.02**	0.01
0.05	2/1	0.05	0.22	0.22	0.22	0.25	0.30	**0.35**
0.05	1/2	0.05	0.10	0.11	0.11	0.16	0.17	**0.22**
0.05	3/2	0.05	0.11	0.11	0.11	0.14	0.14	**0.19**
0.05	2/3	0.05	0.07	0.08	0.08	0.11	0.10	**0.13**
0.05	6/5	0.05	0.07	0.07	0.07	**0.08**	0.07	**0.08**
0.05	5/6	0.05	0.06	0.06	0.06	**0.08**	0.07	0.07
0.1	2/1	0.10	0.32	0.37	0.35	0.43	0.47	**0.53**
0.1	1/2	0.10	0.18	0.22	0.21	0.30	0.31	**0.36**
0.1	3/2	0.10	0.20	0.22	0.21	0.28	0.28	**0.34**
0.1	2/3	0.10	0.13	0.15	0.14	0.22	0.21	**0.24**
0.1	6/5	0.10	0.14	0.14	0.14	**0.17**	0.16	**0.17**
0.1	5/6	0.09	0.11	0.12	0.11	**0.15**	0.14	**0.15**

Table 3: AUC-PAR for Traffic data.

rate	fold	RND	SARI	SIMA	SARIMA	ND	TL1
0.01	2/1	0.00	0.50	0.50	0.58	0.50	**0.64**
0.01	1/2	0.00	0.42	0.39	0.39	0.14	**0.58**
0.01	3/2	0.00	0.00	0.11	0.11	0.11	**0.47**
0.01	2/3	0.00	0.03	**0.14**	0.11	0.11	0.00
0.01	6/5	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
0.01	5/6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
0.05	2/1	0.18	0.61	0.74	0.69	0.67	**0.85**
0.05	1/2	0.04	0.29	0.36	0.35	0.39	**0.55**
0.05	3/2	0.00	0.29	0.43	0.40	0.41	**0.53**
0.05	2/3	0.21	0.14	0.26	0.22	0.19	**0.28**
0.05	6/5	0.00	0.06	0.03	0.02	0.05	**0.08**
0.05	5/6	**0.07**	0.04	0.03	0.06	0.01	0.03
0.1	2/1	0.12	0.61	0.74	0.68	0.74	**0.86**
0.1	1/2	0.05	0.28	0.47	0.43	0.54	**0.61**
0.1	3/2	0.11	0.38	0.52	0.45	0.51	**0.63**
0.1	2/3	0.07	0.13	**0.30**	0.26	0.27	**0.30**
0.1	6/5	0.12	0.19	0.18	0.19	0.18	**0.21**
0.1	5/6	**0.10**	0.06	**0.10**	**0.10**	0.07	0.07

4.0 Change-Point Detection in Time Series

In this chapter, we address the problem of detecting change-points in regular time series. Change-points refer to the time points when the distribution of the data is changed. We only focus on changes in the mean here. Besides the nonstationarity, a new challenge specific to change-point detection is that we need to filter out outliers that are abnormal but temporary and therefore are not change-points. If care is not taken, those temporary outliers can trigger many false alarms as false change-points. We develop two methods. One is based on likelihood-ratio statistics accompanies by several techniques to make the detection algorithm as robust to temporary outliers as possible. The other is based on Bayesian generative models, where we define generative models for normal and different abnormal behaviors of the time series to detect change-points while filtering out temporary outliers. Part of this work has been published at AIME 2017 [65], at BIBM 2017 [67], and in Artificial Intelligence in Medicine [68].

4.1 Likelihood-Ratio-Based Change-Point Detection

First, we propose a change-point detection method using likelihood-ratios. The overall detection framework is based on a sliding window, that is, at each time point, it looks back a constant amount of time, referred to as a window. All analysis is done only on the data within the window. We use the sliding window to restrict our attention to recent data, because the distribution of the time series can drift over time. In such a case, old data add bias to the inference on recent data. The sliding window not only deals with nonstationary behaviors, but also reduces the computational cost of the algorithm, so that it is suitable for online detection. For the data within the window, we perform several steps to get the final output, the score that reflects how significant the change in time series behavior is.

The first step is variance stabilization for count data, which is the same as in Section 3.1.1. Then, to deal with seasonality in the time series, we use STL, which is introduced in

Section 3.1.2. However, here we try to detect change-points, so we only remove the seasonal component and keep both trend and remainder components for the remaining steps. Next, we calculate the likelihood ratio statistics based on the sum of these components as the scores.

4.1.1 Likelihood Ratio Statistics

Given a set of data points $x = \{x_1, x_2, \ldots, x_n\}$ within a window at time t with the seasonal signal removed, we wish to derive a score indicating how likely a change in the mean has occurred in the time span $[t - n + 1, t]$. Ultimately, we wish to know if a change has occurred or not (1 or 0), and a score is a continuous quantity representing our belief that a change has occurred. By applying a threshold to the scores, we can convert them to binary labels indicating changes.

To calculate the scores, we formulate the following hypothesis test for each possible change-point $c, 1 < c \leq n$.

$$\begin{aligned} H_0 : &\quad x_i \sim F(\mu_0), && 1 \leq i \leq n. \\ H_1 : &\quad x_i \sim F(\mu_1), x_j \sim F(\mu_n), && 1 \leq i < c \leq j \leq n. \end{aligned}$$

F is a distribution family with a parameter for the mean. If we fix c, then the likelihood ratio statistic would be

$$r_c = \log L_{H_c} - \log L_{H_0}, \tag{4.1}$$

where L_{H_0} is the maximum likelihood of the sample x under the null hypothesis, and L_{H_c} is under the alternative hypothesis with known c. Since we do not know c, and instead try to detect whether there is a change at *any* point, the score for the sample x is

$$r_* = \max_{1<c\leq n} r_c, \tag{4.2}$$

and the corresponding maximizer is the suspected change-point.

The statistics depend on the distribution family F. Because the data can be quite noisy and contain outliers, we use Student's t-distribution to model the data. Specifically, the probability density function (PDF) is

$$p(x|\nu, \mu, \sigma^2) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu\sigma^2}} \left(1 + \frac{(x-\mu)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}, \tag{4.3}$$

where ν is the degrees of freedom, μ is the location, and σ^2 is the scale.

4.1.2 EM for MLE

We consider ν in Eq. 4.3 as given and only estimate μ and σ^2. For t-distributions, the maximum likelihood estimators (MLEs) do not have a closed-form solution, so we follow Liu and Rubin [63] and develop an EM algorithm for estimating the parameters under either the null or the alternative hypothesis. The EM algorithm is based on an equivalent form of the distribution as an infinite mixture of Gaussians, which includes an additional hidden variable τ:

$$\tau \sim \text{Gamma}(\nu/2, \nu/2), \quad x \sim N(\mu, \sigma^2/\tau), \tag{4.4}$$

where the parameters of the Gamma distribution are shape and rate. The marginal distribution of x in Eq. 4.4 is the t-distribution in Eq. 4.3.

Based on the above, for the null hypothesis, the EM algorithm is as follows. The E-step is

$$w_i = \text{E}[\tau_i | x_i, \nu, \mu_0, \sigma^2] = \frac{\nu + 1}{\frac{(x_i - \mu_0)^2}{\sigma^2} + \nu}. \tag{4.5}$$

The M-step is

$$\mu_0 = \frac{\sum_i w_i x_i}{\sum_i w_i}, \quad \sigma^2 = \frac{\sum_i w_i (x_i - \mu)^2}{n}. \tag{4.6}$$

We alternate between the E-step and M-step till convergence, use the final values of μ_0 and σ^2 as the MLEs.

For the alternative hypothesis, the E-step is almost the same as Eq. 4.5, except that μ_0 is replaced by μ_1 or μ_n depending on whether $i < c$ or not. The M-step is

$$\begin{aligned} \mu_1 &= \frac{\sum_{i<c} w_i x_i}{\sum_{i<c} w_i}, \quad \mu_n = \frac{\sum_{i \geq c} w_i x_i}{\sum_{i \geq c} w_i}, \\ \sigma^2 &= \frac{\sum_{i<c} w_i (x_i - \mu_1)^2 + \sum_{i \geq c} w_i (x_i - \mu_n)^2}{n}. \end{aligned} \tag{4.7}$$

4.1.3 Further Improvements

The data are counts, and even with transformation, low counts are problematic, because the variance is too low. To improve the performance, we add a small noise to the data.

Specifically, for every point x in the time series after transformation, we add a noise as

$$x' = x + \epsilon, \quad \epsilon = u - 0.5, \quad u \sim Beta(a, a). \tag{4.8}$$

We use a (symmetric) beta-distribution, so the size of the noise is within control, $\epsilon \in [-0.5, 0.5]$, and the mean of ϵ is 0.

The second improvement is based on the following observation. When calculating the likelihood ratio statistics, if say $c = 2$ or n, only one point is used for estimating μ_1 or μ_n, so the sample size is small. But our data contain outliers, which can bias the inference especially when the sample size is small. We can make sure the sample size is always greater than l by restricting $l < c \leq n - l + 1$, but an obvious drawback is that the expected delay of the detection would increase. However, noticing that new data always come from the right of the sliding window, and that usually the change can be detected quickly (in the right half of the window), we restrict $l < c \leq n$ instead, so the sample size for estimating μ_1 is at least l, while the delay of the detection would not be affected at all, if without the restriction it would be detected within $n - l$ observations after the change.

4.2 Generative-Model-Based Change-Point Detection

Our second approach for change-point detection is based on a Bayesian generative model, specifically the dynamic linear model (DLM). In the following, we present specifics of our approach. We start with a brief review of the dynamic linear model (DLM) which is used to model the time series. Then, we introduce a DLM extension that allows us to model seasonal (weekly) variations. Finally, we show how to build DLMs reflecting different normal and abnormal behaviors, and how to use them to detect anomalies. Again, we transform counts to real values using the variance stabilization transformation.

4.2.1 Dynamic Linear Model

A dynamic linear model (DLM) [37] is a time-series model where a sequence of observations $\{y_t \in \mathbb{R} : t = 1, 2, \ldots\}$ is modeled indirectly using a sequence of hidden state vectors $\{x_t \in \mathbb{R}^d : t = 1, 2, \ldots\}$ of dimension d. The dynamics of the model is captured by:

$$\begin{aligned} y_t &= Fx_t + v_t, \quad v_t \sim N(0, V). \\ x_t &= Gx_{t-1} + w_t, \quad w_t \sim N(0, W), \end{aligned} \tag{4.9}$$

where G is a transition matrix that models the change in the hidden state over time, and F is an emission matrix that reflects the expression of observations y_t given the current hidden state x_t. Both transition and observation behaviors are stochastic and corrupted by a zero mean Gaussian noise. The noise components are denoted as w_t and v_t in Equation 4.9. W and V are the covariance matrices of these noise components. At the beginning (at time $t = 0$), we assume the hidden state $x_0 \sim N(m_0, C_0)$, where m_0 and C_0 are the mean and covariance of x_0 respectively.

4.2.2 Dynamic Linear Model with Seasonal Variation

The DLM is very flexible in that it can define many different behaviors of the time series including seasonality. Here, we use it to represent weekly cycles, but it can be applied to different seasonality (e.g., monthly) with straightforward changes.

The best way to understand the seasonal DLM is to break the model into multiple components representing the dynamics of the hidden state (x_t). The components are: a baseline value (u_t), a slope value (l_t) reflecting the trend, and a seasonal component (s_t). The seasonal component reflects a seasonal cycle that varies over a period (e.g., 7 days). Let p denote this period. Using p we can define a function of time $[t]_p = (t + p - 1) \bmod p + 1$ that distributes the times into different phases of a cycle (e.g. individual days of the week). Then the dynamics of the time series of observations $\{y_t\}$ can be modeled using the baseline, slope

and seasonal subcomponents of the dynamics as:

$$\begin{aligned} y_t &= u_t + s_t^{([t]_p)} + v_t, \\ u_t &= u_{t-1} + l_t + w_t^{(u)}, \\ l_t &= l_{t-1} + w_t^{(l)}, \\ s_t^{(i)} &= s_{t-1}^{(i)} + w_t^{(i)}, \forall i = 1, 2, \ldots, p. \end{aligned} \tag{4.10}$$

In this Equation, v_t and w_t are independent noises following Gaussian distributions. Note that $s^{(i)}$ is the seasonal level of phase i in the cycle. To avoid redundancy in the parameters we assume:

$$\sum_{i=1}^{p} s_t^{(i)} = 0. \tag{4.11}$$

To represent this model as a DLM, we construct x_t as a composition of the three components above:

$$x_t = (u_t, l_t, s^{([t]_p)}, s^{([t-1]_p)}, \ldots, s^{([t-p+2]_p)})^T. \tag{4.12}$$

Given x_t, the transition and emission matrices in Equation (4.9) can be built from the transitions of its components in Equation (4.10). This leads to the following definitions of F and G matrices:

$$\begin{aligned} F &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & \ldots & 0 \end{bmatrix}_{1 \times d}, \\ G &= \begin{bmatrix} 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & -1 & -1 & \ldots & -1 & -1 \\ 0 & 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}_{d \times d}, \end{aligned} \tag{4.13}$$

where d is the length of x as in (4.12).

4.2.3 Multi-Process Dynamic Linear Model

The seasonal DLM can be used to model time series with trend and seasonality. However, our main objective is to use the model to detect anomalies in the time series. We address this problem by defining multiple DLMs representing normal and different abnormal behaviors and switching between them.

One way to define a collection of switching DLMs is to use a multi-process dynamic linear model (MPDLM) [37]. Briefly, let $\phi^{(i)} = \{F^{(i)}, G^{(i)}, V^{(i)}, W^{(i)}\}$ be the parameters of a DLM i reflecting one of the many time series behaviors. In the MPDLM, the individual DLMs may switch on and off over time depending on what model is driving the behavior of the time series at the moment. Let $M_t^{(i)}$ be a random variable indicating whether model i is driving the time series at time t and generating y_t, and M_t be the vector containing $M_t^{(i)}$ for all i.

Now let Y_t denote the time series up to time t, that is, $Y_t = \{y_u : u = 1, 2, \ldots, t\}$. Given the MPDLM and its parameters, we can calculate the prior probability of a model i driving the time series right before observing y_t: $p(M_t^{(i)} = 1|Y_{t-1})$. After observing y_t, we can calculate the posterior probability of a model i driving the time series: $p(M_t^{(i)} = 1|Y_t)$. These probabilities can help us to infer if there was a switch in the model at time t and hence a change in the time series behavior. We will elaborate the details of the above calculations in Section 4.2.4.

In this work, we use a combination of three DLMs: MS (Model Stable), MAO (Model Additive Outlier), and MLS (Model Level Shift). MS is a model for normal time series behavior, while MAO and MLS model abnormal behaviors. MAO represents a temporary outlier behavior in which the most recent observation is very different from previous observations (defined by the normal model), and this difference is limited to just one time point. MLS represents a baseline change behavior (change-point) in which the new observations are very different from the previous observations (again defined by the normal model), and the changes persist for a longer time.

The main difference between these models is in the variance and covariance V and W (Eq. 4.9). Take MS as the reference model. MAO has a much larger value in the variance V but not W, because a temporary outlier is treated as a temporary noise in the observation

and has no influence on the hidden state of the time series. On the other hand, MLS has a much larger value in some components of W. This makes it easier to absorb the change in the baseline, and as a result, modify the hidden state and its future evolution.

Notice that if we observe an aberrant y_t for the first time, there is no way we can tell whether MAO or MLS has generated it, because the aberrancy could be explained by either the noise in the observation or in the state. Therefore, we wait for the next observation y_{t+1} and calculate $p(M_t^{(i)} = 1|Y_{t+1})$. Figure 9 shows an example of applying the method to a real time series. The top graph shows the observed time series. The remaining three graphs show the posterior probabilities of the three models, as indicated by the labels on the side. Notice that there is one time unit delay in the probability outputs as described above. Briefly most of the time, the time-series behavior is stable and explained by the normal model (MS). This is captured by high posterior probabilities for MS (second graph). However, there are observations that deviate from the normal model. In that case, the posterior probabilities of the other two models (MAO or MLS) go up. The posterior probabilities can be used to infer which model is most likely to explain the observed deviation from the normal model.

4.2.4 Inference

In this section, we describe how to compute $p(M_t|Y_t)$ and $p(M_{t-1}|Y_t)$. This calculation is relatively straightforward as long as we have the joint distribution of the models at time t and $t-1$

$$p(M_{t-1}, M_t|Y_t) \propto p(M_{t-1}|Y_{t-1})p(M_t|M_{t-1}, Y_{t-1})p(y_t|M_t, M_{t-1}, Y_{t-1}), \tag{4.14}$$

where $p(M_{t-1}|Y_{t-1})$ is known, since the computation is carried out recursively along time.

To make the computation of $p(M_t|M_{t-1}, Y_{t-1})$ tractable, we need to make some independence assumptions. It is reasonable to assume $p(M_t|M_{t-1}, Y_{t-1}) = p(M_t|M_{t-1})$, i.e., the transition between the DLMs is first-order Markovian. Since it is more intuitive to specify $p(M_t)$ than $p(M_t|M_{t-1})$ from prior knowledge, we further assume $p(M_t|M_{t-1}) = p(M_t)$, although the inference algorithm presented here still works without this simplification.

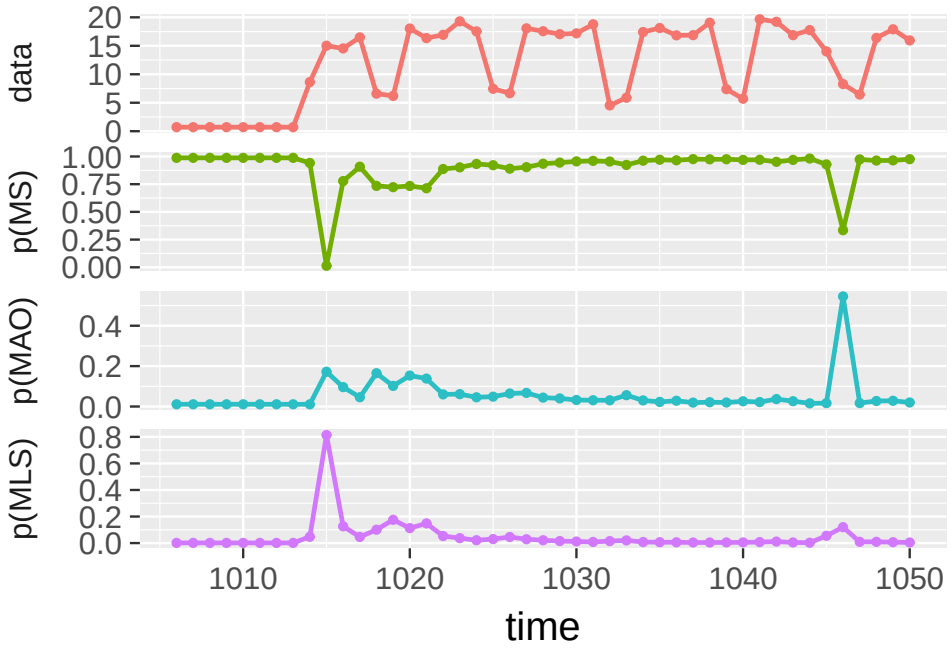


Figure 9: Applying the MPDLM method to a rule-firing count time series. The top graph shows the observed rule firing counts after the transformation. The remaining graphs plot the posterior probabilities of the three models (MS, MAO, MLS). There is one time unit delay for the probability outputs.

The last term in Equation 4.14 can be further broken down as

$$p(y_t|M_t, M_{t-1}, Y_{t-1}) = \iint p(x_{t-1}|M_{t-1}, Y_{t-1})p(x_t|x_{t-1}, M_t)p(y_t|x_t, M_t)dx_{t-1}dx_t. \quad (4.15)$$

The challenge here is to compute $p(x_{t-1}|M_{t-1}, Y_{t-1})$ or in general $p(x_t|M_t, Y_t)$. Conditioning on all $M_{t-i}, i > 0$ at all time points before t, the computation can be carried out through *Kalman filter* [49] in polynomial time. But for the marginal distribution, the computation is intractable, because the number of possible configurations for the chain of M_t's grows exponentially over t, and therefore $p(x_t|M_t, Y_t)$ is a mixture distribution whose number of mixtures is exponential in t. We approximate the mixture distribution by "collapsing" it at every time step to a single Gaussian distribution to make the inference tractable [37].

4.2.5 Parameter Setting

The prior distribution of the models $p(M_t)$ is a multinomial distribution. In the case that we have information about the frequency of different types of behaviors in the data, we can set the probabilities for the models accordingly. Otherwise, we can set a noninformative prior, that is each model has an equal probability.

In the parameters of each model, F and G are already known and fixed in Eq. 4.13, so we only need to set V and W. However, it is tricky to estimate them from the data, because (1) we do not have labels for the data, and since the data could be a mix of normal and abnormal data, the estimates could be biased; (2) even if we have labels, we may not have enough data to estimate the parameters for the abnormal models, MAO and MLS. Therefore, instead we derive some heuristics to set the parameters.

We use three tuning parameters to control the ratios of the variance parameters: $\kappa > 1, \delta > 1, \gamma \in [0, 1]$. Let $\hat{V}$ be an estimate of the variance for normal data. For MS, we set $V = \hat{V}$ and $W = 0$. For MAO, we set $V = \kappa\hat{V}$ and $W = 0$. For MLS, we set $V = \hat{V}$, $W^{(u)} = \gamma\delta\hat{V}$, $W^{(l)} = 0$, and $W^{(i)} = (1-\gamma)\delta\hat{V}, \forall i$. Notice that W is a diagonal matrix, and we used the notation in (4.10) to indicate the components on the diagonal. Intuitively, κ represents the ratio of the size of a temporary outlier to a normal point. Without prior information, setting $\delta = \kappa - 1$ is recommended, because MAO and MLS would have the same variance for y. γ further breaks down the variance in MLS into variance in the baseline and the seasonal levels. We may keep changing these parameters over time if needed, but we found keeping them as constants would suffice in our case.

We can either set $\hat{V}$ using our knowledge of the data, or estimate it from data. For example, one way to estimate the variance is

$$\hat{V} = \left(\frac{1}{p} \sum_{i=1}^{p} \text{CMAD}(\{y_t : [t]_p = i\}) \right)^2 \tag{4.16}$$

where p is the number of phases in a cycle, and CMAD is the median absolute deviation adjusted by a constant factor for consistency. CMAD for all y_t on the same phase is averaged over all phases. CMAD is a robust estimator, making it more desirable than sample variance, given that we know there will be abnormal data.

4.3 Experiments

4.3.1 Experiment Design

We test our method and compare it to alternative methods on rule firing counts from a large teaching hospital collected over a period of approximately 5 years [111]. We run and evaluate the methods by considering both (1) known and (2) simulated changes in their time series.

In the first part of the experiments, we use 14 CDSS rules with a total of 22 labeled change-points. These reflect known changes in the rule logic or confirmed changes in the firing rates due to various issues. In the second part, we simulate changes on the existing rule firing counts to help us analyze the sensitivity of the methods to the magnitude of the changes. We use the firing counts of 4 CDSS rules with no known change-points, and simulate change-points on these data by randomly sampling 10 segments of length 240 per rule and simulating a change in the middle of these segments. We simulate the change at time c in time series $\boldsymbol{x}$ by changing the values as $x'_i = \lambda x_i, i \geq c$. In different experiments, we set λ to 2/1, 3/2, 6/5, 1/2, 2/3, and 5/6 respectively, to cover both increasing and decreasing changes in different sizes. The final values x_i are rounded, so they are still nonnegative integers consistent with counts. We use multiplicative instead of additive changes, because the data are counts and have heteroscedasticity.

We use AMOC curves [28] to evaluate the performance of the methods. In general, a change-point can be detected within an acceptable delay. Meanwhile, normal points can be falsely detected as "change-points", resulting in false positives. In an AMOC curve, the delay of a detection is plotted against the false positive rate (FPR) by varying the threshold on the scores. If a change is not detected at all, a penalty is used as the delay. In the experiments, the maximum delay is 13, which is related to the sliding window size explained later, and the penalty is 14. The first 140 points in each time series are used as a warm-up, and no scores are produced.

We compare our methods with the following widely-used statistical methods for detecting changes in distributions:

- RND: a baseline that gives uniformly sampled scores;
- SCP: single change-point detection method for normal distribution [14, 51];
- MW: a method based on Mann-Whitney nonparametric statistics [87];
- Pois: a method based on Poisson likelihood ratios [14].

We compare two versions of the likelihood-ratio-based method and the generative-model-based method:

- NDT1: our method based on likelihood-ratios without restricting $l < c$;
- NDT2: our method based on likelihood-ratios with restricting $l < c$, where $l = 7$;
- DLM: our method based on DLM.

A window of 14 is used for change detection, while a window of 140 is for STL. The square-root transformation is also used for SCP.

For NDT, we use the robust STL implemented in R [89] and set the period to 7 (a week) and $s.window = 7$ (even smaller values are not recommended [15]). Default values are used for other parameters. $\nu = 3$ for Eq. 4.3 and 4.4. $a = 1$ for Eq. 4.8.

For DLM, the prior distribution of x_0 is set to be $N(0, 10^6 I)$, where I is the identity matrix. We set $\kappa = 100, \delta = \kappa - 1, \gamma = 0.99$ and $\hat{V} = 1$ for MPDLM. We also tried to estimate $\hat{V}$ as in (4.16), but the performance was slightly worse than using a constant in these experiments. The reason, we think, is that the *ratios* between the variances in different models matters more than the *values* of the variances. We prefer setting $\hat{V}$ as a constant for simplicity and robustness.

4.3.2 Results on Data with Known Change-Points

The means of the areas under the AMOC curves (AUC-AMOC) are in Table 4 (row 1). These are calculated by treating the data around each change-point as a single example. For each example, the AUC summarizes the AMOC curve by integrating the delay w.r.t. the FPR. These results show that NDT1 dominates the baselines, while NDT2 is better than NDT1, showing that the proposed likelihood-ratio-based method is better than traditional methods, and the further improvements we proposed in Section 4.1.3 are effective. Meanwhile, DLM is

Table 4: The mean AUC-AMOC averaged over all change-points. *Wilcoxon signed rank tests show that NDT2 and DLM are significantly ($p < 0.05$) better than all the baselines.

data	RND	SCP	MW	Pois	NDT1	NDT2	DLM
real	1.88	0.98	1.16	0.62	0.37	**0.32***	**0.19***
sim (2/1)	2.37	1.26	1.21	1.19	0.63	**0.39***	**0.28***
sim (3/2)	1.97	1.86	1.36	1.88	1.05	**0.82***	**0.68***
sim (6/5)	2.01	2.24	1.74	2.26	2.01	**1.72**	1.88
sim (1/2)	2.36	1.22	1.74	1.19	0.71	**0.57***	**0.50***
sim (2/3)	2.16	1.67	1.86	1.66	1.28	**1.11***	**0.94***
sim (5/6)	2.19	2.06	2.33	2.05	2.05	**1.89**	2.17

also outperforming all the baselines and is even better than NDT2, showing the benefit of using a generative model to take account of both temporary outliers and change-points.

4.3.3 Results on Data with Simulated Change-Points

Table 4 (row 2–7) shows the mean AUC-AMOC for different folds of the simulated changes (λ). NDT2 performs better than the baselines in all cases, while NDT1, although being worse than NDT2, is still better than or similar to the baselines. When $\lambda = 6/5, 5/6$, the differences between NDT2 and the baselines are not significant. DLM is better than NDT2 in all cases except when the change is small ($\lambda = 6/5, 5/6$), corresponding to the cases when NDT2 is not significantly better. This shows that when the changes are really small, it becomes a challenge for all the methods to reliably detect them.

To further investigate the difficulties in detecting smaller changes, we examine closely the AMOC curves. Figure 10 shows all the AMOC curves on the simulated data. They are grouped by experiment settings, i.e., the fold of the simulated changes (λ). Each subgraph corresponds to a different fold, shown in the label on the top. A general trend in these graphs is that, as the changes become smaller, all the curves get closer to the random baseline (RND).

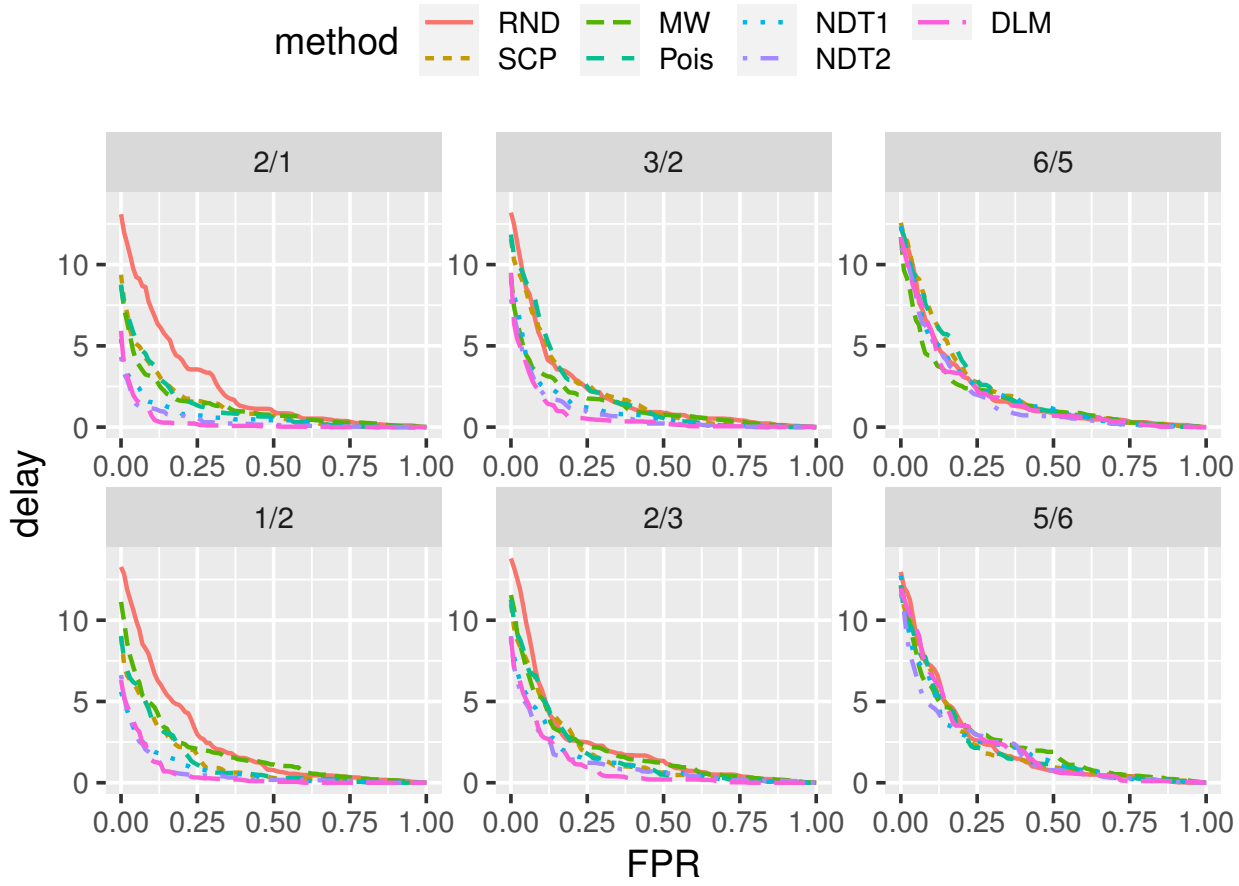


Figure 10: AMOC curves on simulated data averaged over all change-points. The label on top of each subgraph indicates the fold of the changes (λ).

This reflects that the smaller the change, the harder to detect it (in time). However, except when the change is at the smallest setting (the last column of the graphs), our methods dominate the baselines almost everywhere by a noticeable margin.

5.0 Event Sequence Model

In this chapter, we address the problem of modeling event sequences in continuous time. We propose a new nonparametric point process based on Gaussian processes (GP), which has the flexibility similar to GP-modulated point processes (see Section 2.1.2.1) and the applicability similar to Hawkes processes (see Section 2.1.2.2). Part of this work has been published at NeurIPS 2019 [64].

5.1 Introduction

Two main types of point process models have been developed independently over years. One type is what we call "regressive point processes", where the dependencies of the intensity function on the past events are *directly* modeled. Hawkes processes [42] are the most studied and used class of regressive point processes (e.g., [119],[118],[3],[61],[106],[114],[25]). A benefit of the regressive point processes is that they are easy to apply and interpret. They can be learned on a set of sequences and then applied on another *unseen* set of sequences. Since the influence of each past event on the intensity is explicitly modeled, it is easy to see how different types of events influence each other over time.

Another type is what we call "latent-state point processes", where the dependencies of the intensity on the past events are *indirectly* modeled through a latent state. Based on the past latent state, we can infer the future latent state and thereby predict future events. The most studied class of latent-state point processes are Gaussian-process-modulated point processes (e.g., [1],[56],[91],[35],[75],[76],[21],[53]). They use some transformation of a Gaussian process (GP) as the prior for the intensity function, which provides a probabilistic distribution over possible intensity functions and acts as the latent state. Then the posterior of the intensity function can be inferred from the data. The main benefit of GP-modulated point processes is that they provide a principled way to flexibly model the intensity functions. However, a significant drawback is that they are harder to apply, compared with Hawkes processes, due

to the need of inferring a separate latent state for *each* sequence. To make inference on any new sequence, long enough history of the sequence must be available. It is impossible to learn a model from a set of sequences and apply it to other unseen sequences (i.e. cold start).

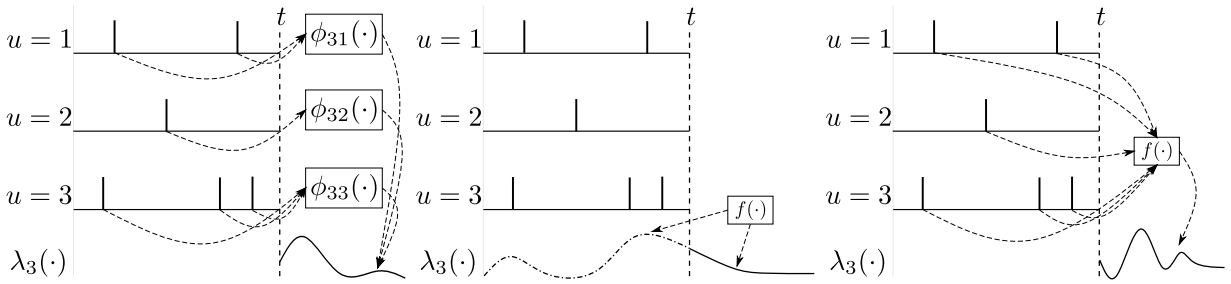
In this work, we propose a new nonparametric model, GP regressive point process (GPRPP), combining the advantages of the above two models: the flexibility of GP-modulated point processes and the applicability of Hawkes processes. Similar to Hawkes processes, our model directly captures the dependencies of the intensity function on the past events. However, unlike Hawkes processes, the dependencies are modeled nonparametrically through a GP. Meanwhile, different from GP-modulated point processes, the input of our GP is not defined by the "absolute" time relative to each sequence, but by the collection of "relative" times from past events of different types. This defines a latent state independent of specific sequences, and therefore can be learned from and applied to different sequences. Figure 11 illustrates the differences between GPRPP and the previous models.

To better model the dependencies of the intensity function on the past events, we propose a conditional GP model for GPRPP. It relies on a set of points introduced in the input space of the GP to capture the dependencies independent of specific sequences. These points, although bearing a similarity to the inducing points in sparse GPs [88, 99], function quite differently, because instead of marginalizing them out, we condition on them for inference.

5.2 Preliminary

The training data consist of multiple sequences $\mathcal{D} = \{y_c\}_{c=1}^{|\mathcal{D}|}$. Each y_c is a sequence of (time, label) pairs $y_c = \{(t_i, u_i)\}_{i=1}^{|y_c|}$, representing the time and the type of each event, where $t_i \in \mathbb{R}_{\geq 0}$ and $u_i = 1, \ldots, U$. A (temporal) point process has a conditional intensity function (CIF) $\lambda(t) = \lim_{dt \to 0^+} \frac{\mathbb{E}\left[N([t,t+dt))|\mathcal{H}_{t^-}\right]}{dt}$ as the instantaneous rate of events at time t given the history $\mathcal{H}_{t^-}$ up to t, where $N(\cdot)$ counts the number of events in an interval. For example, for a Hawkes process, the CIF of event type u_i is defined as

$$\lambda_{u_i}(t) = \mu_{u_i} + \sum_{t_j < t} \phi_{u_i u_j}(t - t_j) \tag{5.1}$$



(a) Hawkes process (b) GP-modulated point process (c) GP regressive point process

Figure 11: Illustrations of different point process models. The first three rows are a multivariate event sequence consisting of events (stems) of three types (u) on the timeline. The vertical line t marks the current time. The last row is the estimated conditional intensity function (CIF) $\lambda_3(\cdot)$ for event type $u = 3$. (a) In a Hawkes process, the CIF depends on the past events through the triggering kernels $\phi_{u_i u_j}$ (Eq. 5.1). (b) In a GP-modulated point process, the entire (transformed) CIF on the specific sequence is a function with a GP prior. (c) In a GPRPP, the (transformed) CIF depends on the past events through a function with a GP prior.

where $\phi_{u_i u_j}$ is a function of time that characterizes the influence of past events of type u_j on type u_i. It is called a *triggering kernel* in previous works. Meanwhile, μ_{u_i} defines the baseline intensity.

Given the CIF, the probability density of the data $\mathcal{D}$ is

$$p(\mathcal{D}) = \prod_{c=1}^{|\mathcal{D}|} p(y_c) = \prod_{c=1}^{|\mathcal{D}|} \prod_{i=1}^{|y_c|} \lambda_{u_i}(t_i) \exp\left(- \int_{t_0^c}^{T^c} \sum_{u=1}^{U} \lambda_u(t) dt \right) \tag{5.2}$$

where t_0^c and T^c are the start and end time of y_c. Without loss of generality, we assume $t_0^c = 0$ from now on. We note that the density of the data factorizes over the individual sequences y_c (Eq. 5.2), while the density of each sequence factorizes over the event types (Eq. 5.3):

$$p(y_c) = \prod_{u=1}^{U} \left[\prod_{i=1}^{|y_c|} \lambda_u(t_i)^{\delta(u_i, u)} \exp\left(- \int_{t_0^c}^{T^c} \lambda_u(t) dt \right) \right] \triangleq \prod_{u=1}^{U} p_u(y_c) \tag{5.3}$$

where $\delta(x, y) = 1$ when $x = y$, and 0 otherwise. In this way, a point process can be viewed as a set of sub-models, one for each type of events. The total likelihood is the product of

the likelihood of all the sub-models. For a Hawkes process, each sub-model is similar to a regression model, where the predictors are the elapsed times since the past events of all types, transformed through the triggering kernels.

5.3 GP Regressive Point Processes

Inspired by the regressive view of Hawkes processes, we propose a new model based on Gaussian processes (GPs). Since the density factorizes over sequences (Eq. 5.2), we describe the method for one sequence y_c in the following. To avoid cluttering, we use y to denote the sequence. Since the density of each sequence factorizes over event types (Eq. 5.3), we describe the method for $p_{\tilde{u}}(y)$ of one type $\tilde{u}$ (*target type*) and call the events of type $\tilde{u}$ the *target events*. The same method can be repetitively applied to all $p_u(y)$, $u = 1, \ldots, U$. However, we stress that even for one target type $\tilde{u}$, $p_{\tilde{u}}(y)$ can depend on *all* types of events in the history through the CIF (Eq. 5.1 and 5.4).

In our model, for each target type $\tilde{u}$, the CIF $\lambda_{\tilde{u}}(t)$ is a transformation of a function f drawn from a GP with mean μ and covariance function K, $f \sim \mathcal{GP}(\mu, K)$. We note that f, μ and K can be different for $\lambda_u(t)$ of a different type u. The input of f consists of the elapsed times since the last Q events of all the types. That is, $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^D_{\geq 0}$, $D = U \times Q$, and U is the size of the label set. To convert it to a valid CIF, we use the square transformation

$$\lambda_{\tilde{u}}(t) = f(x(t))^2 = f(t - s_1^1(t), \ldots, t - s_U^Q(t))^2 \tag{5.4}$$

where $s_u^q(t)$ is the time of the q-th (from last) event of type u before time t, which could be undefined when no such event exists. The input of the GP is $x(t) = (t - s_u^q(t))_{u=1,q=1}^{U,Q} \in \mathcal{X}$. That is, x depends on the current time t and the last Q events of *all* types. The d-th dimension of $x(t)$ is $x_d(t) = t - s_u^q(t)$, corresponding to the time elapsed since the q-th (from last) event of type u. In fact, Q *does not have to be the same for each type* u, i.e., we can have a different Q_u for each type u, but for notational simplicity, we use Q as if it were the same for all types. We note that the CIF of the model *directly* depends on the past events and call the model a *GP regressive point process* (GPRPP).

The square transformation ensures nonnegativity of λ and enables closed-form evaluation of the integrals in the likelihood as shown in the next section. It was originally proposed in [75] for GP-modulated point processes for event sequences without types and then exploited in later works (e.g, [76, 21]). Compared with these works, where the GP is a function of the *single* "absolute" time, in our model, the GP is a function of *multiple* "relative" times since past events, which keeps changing as new events happen. This makes the inference much harder, and new efficient algorithms to evaluate the integrals are developed in this work.

A key challenge in the model definition is to deal with undefined inputs. That is, inevitably at some time t (e.g., at the very beginning of a sequence), the q-th (from last) event of type u may not exist. Inspired by Hawkes processes, we come up with a novel kernel for the GP. We start by augmenting each input $x_d(t)$ with an additional indicator $\mathbb{I}[x_d(t)]$ to indicate whether the q-th (from last) event of type u exists, i.e., whether $s_u^q(t)$ (correspondingly $x_d(t)$) is defined. When $s_u^q(t)$ is undefined (there are less than q events of type u in the past), we can set a dummy value for $x_d(t)$ (e.g., $x_d(t) = \infty$), and $\mathbb{I}[x_d(t)] = 0$. Otherwise, $x_d(t) = t - s_u^q(t)$ as before, and $\mathbb{I}[x_d(t)] = 1$. The dimensionality of the input essentially becomes $2D$, but for notational simplicity, the indicators are implicit in $x(t)$. We define the kernel as

$$K(x(t), x'(t')) = \sum_{d=1}^{D} \underbrace{\mathbb{I}[x_d(t)]\,\mathbb{I}[x'_d(t')]}_{K_1} \underbrace{\gamma_d \exp\left(-\frac{(x_d(t) - x'_d(t'))^2}{2\alpha_d}\right)}_{K_2} \tag{5.5}$$

where $\gamma_d, \alpha_d > 0$. This is essentially a sum of D kernels, each of which is a product of two kernels K_1 and K_2. K_2 is the squared-exponential kernel on the value of $x_d(t)$. K_1 is the inner product on the indicator $\mathbb{I}[x_d(t)]$. We use the squared-exponential kernel, because it is widely used and has closed-form evaluations of ψ and Ψ as shown in the next section, but it can be replaced by any kernel with the latter property.

Remark. *The two inputs of the kernel have different notations for x and t, indicating they can come from different sequences at different* absolute *times. The kernel is actually isolated from the absolute time t in the individual sequences, since it only depends on the value of $x(\cdot)$ at t. This is very different from previous GP-based models (e.g., [75, 76, 21]), where the inputs of the kernel always come from the same sequence, and the kernel depends on the absolute time t in the sequence.*

We make the following two assumptions to justify the definition of our model.

Assumption 5.3.1. *For each type u of events, they have time-limited influences on the target events. That is, there is a time limit, $\Delta T_u < \infty$, for each type u of event, such that for any event of type u occurring at s_u, $\lambda_{\tilde{u}}(t)$ may depend on s_u only if $0 < t - s_u \leq \Delta T_u$.*

Assumption 5.3.2. *For each type u of events, there exists $M_u : \mathbb{R} \to \mathbb{Z}$ such that for any bounded time interval $\mathcal{I} = [t_{beg}, t_{end})$, $|\mathcal{I}| = t_{end} - t_{beg} < \infty$, the number of events $N_u(\mathcal{I}) \leq M_u(|\mathcal{I}|) < \infty$.*

Theorem 5.3.1. *Given that assumption 5.3.1 and 5.3.2 hold, there exists $Q < \infty$ such that $\lambda_{\tilde{u}}(t)$ depends on at most the last Q events of any type at any time t.*

Proof. Given any event sequence $y = \{(t_i, u_i)\}_i^{|y|}$, the CIF $\lambda_{\tilde{u}}(t)$ at any time t may only depend on the subset of events $\{(t_i, u_i) \in y : 0 < t - t_i \leq \Delta T_{u_i}\}$, according to Assumption 5.3.1. Focusing on a specific type u, the subset of events that $\lambda_{\tilde{u}}(t)$ may depend on is $\{(t_i, u_i) \in y : 0 < t - t_i \leq \Delta T_u, u_i = u\}$. Notice that all of these events occur within the time interval $[t - \Delta T_u, t)$, which is a bounded interval, since $\Delta T_u < \infty$. Therefore, from Assumption 5.3.2, we have $N_u([t - \Delta T_u, t)) \leq M_u(\Delta T_u) < \infty$ for some M_u, which holds for any time t. That is, $\lambda_{\tilde{u}}(t)$ depends on at most the last $M_u(\Delta T_u)$ events of type u at any time t. To complete the proof, let $Q = \max_{u=1,\dots,U} M_u(\Delta T_u)$. □

5.4 Conditional GPRPP

In this section, we propose a conditional GP model for GPRPP and call it conditional GP regressive point process (CGPRPP). The input of the GP is defined in the previous section and denoted as $x = x(t)$. When t is not important or clear from the context, we just denote the input as x.

Previously, different forms of sparse GPs based on inducing variables have been proposed to improve the efficiency of GPs (e.g., [88, 99]). Typically, a set of inducing points are introduced in the input space, and the inducing variables corresponding to the points are *marginalized* out for learning and inference. Our idea is similar, but the difference is that we

condition on the inducing variables, which correspond to the values of the CIF given different situations of the history. Therefore, we call these points *conditional points*.

Let $Z \in \mathcal{X}^M$ be a sequence of M conditional points in the input space. We will explain how to pick these points later. Given any input $x \in \mathcal{X}$, the output of the GP is $f_x = f(x) \in \mathbb{R}$. Let $f_Z = f(Z) \in \mathbb{R}^M$. We define μ_x and μ_Z as the prior mean μ of the appropriate dimensions for x and Z respectively. Let $K_{xx'} = K(x, x')$ as defined in Eq. 5.5 for any inputs x and x'. If x and x' are vectors, $K_{xx'}$ is the Gram matrix of the corresponding size. Then $p(f_Z) = \mathcal{N}(\mu_Z, K_{ZZ})$, and $p(f_x|f_Z) = \mathcal{N}(\mu_{x|Z}, \sigma^2_{x|Z})$, where

$$\mu_{x|Z} = \mu_x + K_{xZ}K_{ZZ}^{-1}(f_Z - \mu_Z), \qquad \sigma^2_{x|Z} = K_{xx} - K_{xZ}K_{ZZ}^{-1}K_{Zx} \tag{5.6}$$

From Eq. 5.3 and 5.4, the conditional density of the sequence y given f_x is

$$\ln p_{\tilde{u}}(y|f_x) = \sum_{n=1}^{N} \delta(u_n, \tilde{u}) \ln f(x(t_n))^2 - \int_0^T f(x(t))^2 dt \tag{5.7}$$

where $N = |y|$ is the total number of *all* types of events in y.

Remark. *It is worth noting that the conditional points in Z are independent of any specific sequence y, since they are points in $\mathcal{X}$.*

Assuming we observe $f_Z = m_Z$, we can maximize the conditional density $p_{\tilde{u}}(y|m_Z)$ to learn the hyper-parameters of the model:

$$\ln p_{\tilde{u}}(y|m_Z) = \ln \int p_{\tilde{u}}(y|f_x)p(f_x|m_Z)df_x$$

where $p(f_x|m_Z)$ is defined by Eq. 5.6 with $f_Z = m_Z$. Because there is a correspondence between $f(x(t))$ and the CIF $\lambda(t)$, m_Z essentially corresponds to different values of the CIF given different situations of the history, determined by different $z \in Z$. Even given the exact same history, the CIF may still be stochastic. Therefore, we allow noise in f_Z, which is a generalization of the noiseless case, so $f_Z = m_Z + \epsilon_Z$, where $p(\epsilon_Z) = \mathcal{N}(0, S_\epsilon)$. Then we can marginalize out ϵ_Z by integrating w.r.t. $p(\epsilon_z)$. In the end, we maximize

$$\ln p_{\tilde{u}}(y|m_Z) = \ln \iint p_{\tilde{u}}(y|f_x)p(f_x|m_Z, \epsilon_Z)p(\epsilon_Z)df_x d\epsilon_Z = \ln \int p_{\tilde{u}}(y|f_x)p(f_x|m_Z)df_x \tag{5.8}$$

where $p(f_x|m_Z, \epsilon_Z)$ is defined by Eq. 5.6, and

$$p(f_x|m_Z) = \int p(f_x|m_Z, \epsilon_Z)p(\epsilon_Z)d\epsilon_Z = \mathcal{N}(\tilde{\mu}_x, \tilde{\sigma}_x^2)$$

has a closed-form solution

$$\tilde{\mu}_x = \mu_x + K_{xZ}K_{ZZ}^{-1}(m_Z - \mu_Z), \quad \tilde{\sigma}_x^2 = K_{xx} - K_{xZ}K_{ZZ}^{-1}K_{Zx} + K_{xZ}K_{ZZ}^{-1}S_\epsilon K_{ZZ}^{-1}K_{Zx}. \quad (5.9)$$

Remark. *Because we* condition *on the pseudo-observations m_Z, they can move freely when we optimize Eq. 5.8, and their values are determined by fitting to the training data. Intuitively, they act as key points of the CIF, which are supposed to capture the key information regarding the entire CIF.*

Eq. 5.8 is hard to maximize directly. Instead, we derive a lower bound using Jensen's inequality and maximize the lower bound

$$\ln \int p_{\tilde{u}}(y|f_x)p(f_x|m_Z)df_x = \ln \mathbb{E}\left[p_{\tilde{u}}(y|f_x)\right] \geq \mathbb{E}\left[\ln p_{\tilde{u}}(y|f_x)\right] \quad (5.10)$$

where the expectation is w.r.t. $p(f_x|m_Z)$, and from Eq. 5.7

$$\mathbb{E}\left[\ln p_{\tilde{u}}(y|f_x)\right] = \sum_{n=1}^{N} \delta(u_n, \tilde{u})\mathbb{E}\left[\ln f(x(t_n))^2\right] - \sum_{n=1}^{N+1} \int_{t_{n-1}}^{t_n} \left(\mathbb{E}\left[f(x(t))\right]^2 + \mathrm{Var}\left[f(x(t))\right]\right) dt$$

where we define $t_0 = 0$ and $t_{N+1} = T$ to be the start and end time of the sequence y.

From [75], we have

$$\mathbb{E}\left[\ln f(x(t_n))^2\right] = -\tilde{G}\left(-\frac{\tilde{\mu}_x^2}{2\tilde{\sigma}_x^2}\right) + \ln\left(\frac{\tilde{\sigma}_x^2}{2}\right) - C \quad (5.11)$$

where $C \approx 0.57721566$ is the Euler-Mascheroni constant, $\tilde{G}$ is defined via the confluent hypergeometric function, and $\tilde{\mu}_x^2 = \mathbb{E}\left[f_x\right]^2, \tilde{\sigma}_x^2 = \mathrm{Var}\left[f_x\right]$ can be computed as in Eq. 5.9.

Meanwhile,

$$\begin{aligned}\int_{t_{n-1}}^{t_n} \mathbb{E}\left[f_x\right]^2 dt =& (t_n - t_{n-1})\mu_x^2 + 2\mu_x\psi_n^T K_{ZZ}^{-1}(m_Z - \mu_Z) \\ &+ (m_Z - \mu_Z)^T K_{ZZ}^{-1}\Psi_n K_{ZZ}^{-1}(m_Z - \mu_Z), \quad (5.12)\end{aligned}$$

$$\int_{t_{n-1}}^{t_n} \mathrm{Var}\left[f_x\right] dt = \sum_{d=1}^{D} \int_{t_{n-1}}^{t_n} \gamma_d \mathbb{I}\left[x_d(t)\right] dt - \mathrm{Tr}\left(K_{ZZ}^{-1}\Psi_n\right) + \mathrm{Tr}\left(K_{ZZ}^{-1}S_\epsilon K_{ZZ}^{-1}\Psi_n\right). \quad (5.13)$$

The definitions of ψ and Ψ are as follows. Define $v_{d,n,m} = t_n - s_d(t_m)$, where $s_d(t)$ is the q_d-th (from last) point of type u_d before t (q_d and u_d are determined by the dimension d). Then for any $z, z' \in Z$

$$\psi_n(z) = \sum_{d=1}^{D} \mathbb{I}\left[z_d\right] \mathbb{I}\left[s_d(t_n)\right] \gamma_d \frac{\sqrt{\pi\alpha_d}}{\sqrt{2}} \left[\operatorname{erf}\left(\frac{v_{d,n,n} - z_d}{\sqrt{2\alpha_d}}\right) - \operatorname{erf}\left(\frac{v_{d,n-1,n} - z_d}{\sqrt{2\alpha_d}}\right)\right], \tag{5.14}$$

$$\begin{aligned}\Psi_n(z, z') = \sum_{i,j}^{D} & \mathbb{I}\left[z_i\right] \mathbb{I}\left[z'_j\right] \mathbb{I}\left[s_i(t_n)\right] \mathbb{I}\left[s_j(t_n)\right] \gamma_i\gamma_j \frac{\sqrt{\pi\alpha_i\alpha_j}}{\sqrt{2(\alpha_i+\alpha_j)}} \\ & \exp\left(-\frac{(z_i + s_i(t_n) - z'_j - s_j(t_n))^2}{2(\alpha_i+\alpha_j)}\right) \\ & \left[\operatorname{erf}\left(\frac{\alpha_i(v_{j,n,n} - z'_j) + \alpha_j(v_{i,n,n} - z_i)}{\sqrt{2\alpha_i\alpha_j(\alpha_i+\alpha_j)}}\right)\right. \\ & \left. -\operatorname{erf}\left(\frac{\alpha_i(v_{j,n-1,n} - z'_j) + \alpha_j(v_{i,n-1,n} - z_i)}{\sqrt{2\alpha_i\alpha_j(\alpha_i+\alpha_j)}}\right)\right].\end{aligned} \tag{5.15}$$

We note that both $\sum_n \psi_n$ and $\sum_n \Psi_n$ can be combined to improve efficiency. A straightforward implementation would cost $O(ND^2)$, where N is the total number of points. The key thing to notice is that for fixed dimensions d, i, j, the types of points that matter are only the ones related to the dimensions, while we can integrate over the other types of points in closed form. Specifically,

$$\sum_n \psi_n(z) = \sum_{d=1}^{D} \mathbb{I}\left[z_d\right] \gamma_d \frac{\sqrt{\pi\alpha_d}}{\sqrt{2}} g_d, \tag{5.16}$$

$$\sum_n \Psi_n(z, z') = \sum_{i,j}^{D} \mathbb{I}\left[z_i\right] \mathbb{I}\left[z'_j\right] \gamma_i\gamma_j \frac{\sqrt{\pi\alpha_i\alpha_j}}{\sqrt{2(\alpha_i+\alpha_j)}} G_{ij}, \tag{5.17}$$

where

$$g_d = \sum_{k=b_d}^{N_{u_d}+1} \left[\operatorname{erf}\left(\frac{v_{d,(k),(k)} - z_d}{\sqrt{2\alpha_d}}\right) - \operatorname{erf}\left(\frac{v_{d,(k-1),(k)} - z_d}{\sqrt{2\alpha_d}}\right)\right],$$

$$\begin{aligned}G_{ij} = \sum_{k=b_{ij}}^{N_{u_i,u_j}+1} & \exp\left(-\frac{(z_i + s_i(t_{(k)}) - z'_j - s_j(t_{(k)}))^2}{2(\alpha_i+\alpha_j)}\right) \\ & \left[\operatorname{erf}\left(\frac{\alpha_i(v_{j,(k),(k)} - z'_j) + \alpha_j(v_{i,(k),(k)} - z_i)}{\sqrt{2\alpha_i\alpha_j(\alpha_i+\alpha_j)}}\right)\right. \\ & \left. -\operatorname{erf}\left(\frac{\alpha_i(v_{j,(k-1),(k)} - z'_j) + \alpha_j(v_{i,(k-1),(k)} - z_i)}{\sqrt{2\alpha_i\alpha_j(\alpha_i+\alpha_j)}}\right)\right].\end{aligned}$$

For g_d, N_{u_d} is the number of points of type u_d, $t_{(k)}$ is the time of the k-th such point (i.e., (k) maps the index k in the sub-sequence of type u_d to the index of the same point in the full sequence), b_d is the index of the first such point with at least q_d points of type u_d before it, and $t_{(N_{u_d}+1)} = T$. For G_{ij}, $N_{u_i,u_j} = N_{u_i} + N_{u_j}$ is the number of points in the combined sequence of points of both type u_i and type u_j, $t_{(k)}$ is the time of the k-th such point, b_{ij} is the index of the first such point with at least q_i points of type u_i and q_j points of type u_j before it, and $t_{(N_{u_i,u_j}+1)} = T$.

In this way, the calculation of $\sum_n \psi_n(z)$ and $\sum_n \Psi_n(z, z')$ can be done in $O(NDQ)$ if we share the same Q across all types u. If we only set $Q > 1$ for one type, e.g., for $u = 1$, and set $Q = 1$ for the other types, and if $N_1 Q_1 = O(N)$, then the bound becomes $O(ND)$. Either way, it is an improvement compared with $O(ND^2)$ for a straightforward implementation. Empirically, we can improve the performance even further by pre-calculating once and storing the values of v and $s_i - s_j$ at the beginning.

Theorem 5.4.1. *The time complexity for calculating ψ and Ψ using our algorithm is $O(M^2NDQ)$. By setting each conditional point to be active on only one dimension and $Q_u = 1$ for $u \neq \tilde{u}$, the complexity can be reduced to $O(M^2N)$.*

Proof. We only prove the bound for $\sum_n \Psi_n(z, z')$, since $\sum_n \psi_n(z)$ is more efficient to compute. To compute $\sum_n \Psi_n(z, z')$, we need to sum over all pairs of dimensions $i, j = 1, \ldots, D$. However, for each pair of (i, j), we only need to sum over at most $N_{u_i} + N_{u_j}$ items, where N_u is the number of points of type u, and u_i, u_j are the types for dimension i, j. The reason is that the points of the other types in the middle can be integrated over in closed-form. Therefore, the total number of items to sum over is at most

$$\sum_{i=1}^{D} \sum_{j=1}^{D} (N_{u_i} + N_{u_j}) = 2D \sum_{u=1}^{U} N_u Q_u$$

where Q_u is the regression hyper-parameter Q for type u.

In summary, in the most general case, the complexity is $O(D \sum_u N_u Q_u)$. If we use the same Q for all types, then it becomes $O(NDQ)$. If we use $Q > 1$ for only one type, say $u = 1$, and $N_1 Q_1 = O(N)$, then it becomes $O(ND)$. In practice, we can use the symmetric property of the sum and almost halve the amount of computation.

In the most general case when each conditional point can have D active dimensions, we need to do the above computation for each pair of (z, z'). The total complexity is $O(M^2NDQ)$, where M is the total number of conditional points. However, if we set each conditional point active on only one dimension, then the complexity becomes $O(M^2NQ)$. Additionally, if we set $Q > 1$ for only one type, say $u = 1$, and $N_1Q_1 = O(N)$, then it becomes $O(M^2N)$. □

5.4.1 Learning

We also add an independent noise kernel $\sigma^2 I$ to the existing kernel (Eq. 5.5), which results in a new term in the integral of the variance (Eq. 5.13). For learning the model, we maximize the lower bound (Eq. 5.10) w.r.t. the set of hyper-parameters $\Theta = \{\mu, \alpha, \gamma, \sigma, m_Z, S_\epsilon\}$.

We assume that m_Z provides sufficient information for the inference on the test data $\mathcal{D}_*$.

Assumption 5.4.1. *Conditioned on m_Z, the test data $\mathcal{D}_*$ is independent from the training data $\mathcal{D}$, $p(\mathcal{D}_*|\mathcal{D}, m_Z) = p(\mathcal{D}_*|m_Z)$.*

5.4.2 Inference

For inference of the test likelihood, to compare with non-Bayesian models, we use a point estimate of the CIF, instead of model averaging. We use the optimal hyper-parameters Θ^* learned from the training data $\mathcal{D}$ to estimate the mean CIF

$$\lambda^*_{\tilde{u}}(t) = \mathbb{E}\left[\lambda_{\tilde{u}}(t)|\Theta^*\right] = \mathbb{E}\left[f(x(t))|\Theta^*\right]^2 + \mathrm{Var}\left[f(x(t))|\Theta^*\right] \tag{5.18}$$

on the test data $\mathcal{D}_*$, where the conditional mean and variance are defined in Eq. 5.9. Then we use the mean CIF as our prediction to compute the likelihood $p(\mathcal{D}_*|\lambda^*)$ on the test data.

5.4.3 Time Prediction

For predicting the time of the next target event, given the history up to a time point t, we compute the *expected time* for the next target event given the CIF $\lambda_{\tilde{u}}$

$$\mathbb{E}\left[s_{\tilde{u}}|\lambda_{\tilde{u}}\right] = \int_t^\infty s_{\tilde{u}}\lambda_{\tilde{u}}(s_{\tilde{u}}) \exp\left(-\int_t^{s_{\tilde{u}}} \lambda_{\tilde{u}}(v)dv\right) ds_{\tilde{u}} \tag{5.19}$$

for $s_{\tilde{u}} \in (t, \infty)$, where $\lambda_{\tilde{u}}$ depends on the history $\mathcal{H}_{t^-}$ and f_x. That is

$$\mathbb{E}\left[s_{\tilde{u}}|\lambda_{\tilde{u}}\right] = \mathbb{E}\left[s_{\tilde{u}}|\mathcal{H}_{t^-}, f_x\right] = \int_t^\infty s_{\tilde{u}} f(x(s_{\tilde{u}}))^2 \exp\left(-\int_t^{s_{\tilde{u}}} f(x(v))^2 dv\right) ds_{\tilde{u}} \tag{5.20}$$

From here, we can take expectation w.r.t. f_x using the conditional-point approximation. In the end, the prediction is

$$\begin{aligned}\mathbb{E}\left[s_{\tilde{u}}|\mathcal{H}_{t^-}\right] &= \iint \mathbb{E}\left[s_{\tilde{u}}|\mathcal{H}_{t^-}, f_x\right] p(f_x|m_Z^*, \epsilon_Z) p(\epsilon_Z|S_\epsilon^*) df_x d\epsilon_Z \\ &= \int \mathbb{E}\left[s_{\tilde{u}}|\mathcal{H}_{t^-}, f_x\right] p(f_x|m_Z^*, S_\epsilon^*) df_x\end{aligned} \tag{5.21}$$

where m_Z^* and S_ϵ^* are part of the hyper-parameters Θ^* learned from the training data. The expectation w.r.t. f_x is evaluated using Monte-Carlo sampling, and $\mathbb{E}\left[s|\mathcal{H}_{t^-}, f_x\right]$ is evaluated by sampling the point process through Ogata's modified thinning algorithm [82].

An alternative approach, which is more efficient, is to use the mean CIF

$$\lambda_{\tilde{u}}^*(s_{\tilde{u}}) = \mathbb{E}\left[\lambda_{\tilde{u}}(s_{\tilde{u}})|\Theta^*\right] = \mathbb{E}\left[f(x(s_{\tilde{u}}))|\Theta^*\right]^2 + \mathrm{Var}\left[f(x(s_{\tilde{u}}))|\Theta^*\right], \quad s_{\tilde{u}} \in (t, \infty) \tag{5.22}$$

to predict the events without sampling f_x. That is, we estimate $\lambda_{\tilde{u}}^*$ using the learned hyper-parameters Θ^* and the history $\mathcal{H}_{t^-}$, and plug $\lambda_{\tilde{u}}^*$ into Eq. 5.19 to estimate the time to the next event. We used this approach in the experiments and found it to be effective.

5.4.4 Conditional Point Placement

Due to the high dimensionality of the input space of f, it is preferable that the conditional points are placed beforehand and fixed in the learning procedure. Based on the additive form of our kernel, we place the conditional points independently on each dimension. Each conditional point will be active on only one dimension. In our experiments, for simplicity, we put the conditional points regularly on each dimension within a region. If prior knowledge is available, it can be used to determine the region; otherwise, we can use the following heuristics. The left bound of the region is usually 0. The right bound can be set to the maximum (or some quantile) of the time span between two ($Q = 1$) or more ($Q > 1$) consecutive points of the same type, since beyond that, the conditional points will have limited effects.

5.5 Experiments

We compare our method with two state-of-the-art nonparametric Hawkes process variants. **HP-GS** [114] is a nonparametric Hawkes process using a set of (Gaussian) basis functions to approximate the triggering kernels, with sparse and group lasso regularization. For each experiment, we tune its tuning parameters α_S and α_G in a wide range $\{10^{-2}, 10^{-1}, \ldots, 10^4\}$ as in the original work using cross-validation based on the likelihood. In all the experiments, the bandwidth of the Gaussian kernels is set to be optimal, that is the inverse of the cut-off frequency, based on the positions of the kernels. The cut-off frequency $\omega_0 = \pi M/T$, where M is the number of kernels and T is the right bound on the kernels. **HP-LS** [25] is another nonparametric Hawkes process. This method allows very flexible triggering kernels to be estimated by discretizing the kernels and solving a least-square problem. Its parameters are set in accordance with the other methods for each experiment. For our method, to improve efficiency, when we set $Q > 1$, we only set it for the target type and keep $Q = 1$ for the others. We tie the parameters for different dimensions $q = 1, \ldots, Q$ of the same type u.

5.5.1 Synthetic Datasets

First we generate two synthetic datasets representing two distinctive types of event sequences using the thinning algorithm [82]. The first dataset is generated through a renewal process. The baseline intensity is μ. When there is a new event, the intensity temporarily becomes $A(1 - \sin(2\pi t/\tau))$, for a limited time $t \in (0, \tau)$ after the event. Each new event will reset the intensity. We set $\mu = 0.1, A = 0.1, \tau = 20$.

The second dataset is generated through a Hawkes process. The baseline intensity is μ. The triggering kernel is $A \exp(-(t-b)^2/\sigma^2)$, i.e., a Gaussian kernel. Different from the renewal process, each new event will add a new Gaussian kernel on top of the existing intensity. We set $\mu = 0.1, A = 0.1, b = 10, \sigma^2 = 4$.

For each dataset we generate 200 sequences of length of 100 time units each. Each dataset is split into 100 training sequences and 100 testing sequences. For the first dataset, we set $Q = 1$ and conditional points at $0, 5, \ldots, 15$ for CGPRPP. For HP-GS, the kernels are also

placed at $0, 5, \ldots, 15$. For HP-LS, we set $h = 1, k = 20$. For the second dataset, we use the same settings for all the methods, except that we vary $Q = 1, 5, 10, 20, 40$ for CGPRPP to see the effect of adding more regression terms.

We visualize the influence from a past event of a specific type to the target events as the changes in the intensity of the target type over time since that event. For Hawkes processes, it is similar to plotting the triggering kernels, except that the triggering kernels are added on top of the baseline intensity, so we can compare the intensity after an event with the baseline intensity. For CGPRPP, it is equivalent to simulating an event at time 0 and plotting the changes in the intensity as time elapses.

The true influence functions are in the first column in Figure 12, followed by the inferred influence functions for each method. For the first dataset, HP-GS cannot learn the influence function, because its limitation in the dependencies of the CIF on the past events. Although the triggering kernels are nonparametric, the baseline intensity and the triggering kernels are additive in the CIF. This limitation is quite common in nonparametric Hawkes processes (e.g., [117, 22, 118]). HP-LS is more flexible and learns a better influence function, but the discretization tends to make the function noisy. CGPRPP almost completely recovers the true influence function. We note that this influence represents an inhibition followed by an excitation, which is common in practice such as neural spike trains [25]. However, most Hawkes process variants can only model either excitations or inhibitions, but not a mix of both at the same time. In contrast, CGPRPP models the whole CIF as a nonparametric function of the past events and therefore can model these more complex dependencies.

For the second dataset, HP-GS is a perfect match for the data, so unsurprisingly it recovers the influence function very well. HP-LS also learns the influence reasonably well, although still suffering from discretization. Interestingly, CGPRPP with $Q = 40$ (similar for $Q = 10, 20$) learns an influence function very close to HP-GS.

The test log-likelihood on the synthetic datasets are shown in Table 5. CGPRPP performs the best on the first dataset, while HP-GS and CGPRPP perform similarly on the second dataset, with CGPRPP being marginally better. The likelihood results are concordant with how well the models recovered the influence function. For the second dataset, we show the performance of CGPRPP selected with the *training* likelihood ($Q = 40$). A comparison of

GPRPP based on variational sparse GP [75] and conditional GP, and the effect of varying Q on the performance of CGPRPP are in the following sections.

5.5.2 Conditional GP vs. Variational Sparse GP

We compare the performance of GPRPP based on variational sparse GP with inducing points [75] and CGPRPP based on conditional GP with conditional points. Figure 13 shows the test log-likelihood of GPRPP and CGPRPP with $Q = 1, 5, 10, 20, 40$ on the second synthetic dataset. Conditional-GP-based model outperforms variational-sparse-GP-based model in all cases, showing that conditional GP can capture the dependencies between events better.

5.5.3 Effect of Varying Q

Figure 13 shows the test log-likelihood of CGPRPP with $Q = 1, 5, 10, 20, 40$ on the second synthetic dataset. We notice that Q does affect the performance of CGPRPP, especially when it is small and the model is a mismatch for the data. However, as Q increases, the performance tends to stabilize. For data generated through Hawkes processes, it is beneficial to have Q large enough so the model is capable of approximating the compound influences from all the past events. However, in general, for data generated through processes other than Hawkes processes, an optimal Q may need to be neither too small nor too large, and therefore selecting Q may be necessary. In the experiments, we simply use the training likelihood to select Q, which turns out to be effective in most cases. A potential improvement is to use cross-validation, which we do not explore in this work.

5.5.4 IPTV Dataset

The IPTV dataset consists of TV viewing records of users over 11 months [77, 114]. Each sequence consists of times and types of the TV programs viewed by a user. Events in this dataset are generally very bursty, i.e., one event tends to trigger a group of events of the same type happening in a relatively short amount of time, while the distance between these

burst groups are relatively large. This is a distinctive characteristic of data generated by Hawkes processes, so we expect the Hawkes process baselines to perform well. To the best of our knowledge, HP-GS has the best performance on this dataset, but our goal is to confirm whether CGPRPP can also fit the data well and achieve similar or better performance.

The data are extracted from THAP[1] [113], which contain 302 users in total. For efficiency, we randomly sample 200 users and use 100 for training and the others for testing. The original dataset contains 302 users and 16 different types of events (genres of TV programs). Table 7 shows the counts of these different types of events. For efficiency, we randomly sampled 200 users and used 100 users for training and the others for testing. We removed the last two types of programs, "education" and "ads", due to extremely low counts.

All the models are trained on one month and tested on the following three months. We used data in March for training and the following months for testing (on separate users). We picked March a priori, because it has fewer irregularities such as holidays than the first two months.

For HP-GS, we put the kernels at every 20 minutes from 0 up to 24 hours, since the length of most TV programs is about 20 to 40 minutes [114]. For HP-LS, we train multiple models with $h = 1.25, 5, 20$ minutes and $k = (24 * 60 + 20)/h$ correspondingly. For CGPRPP, the conditional points are also placed at every 20 minutes up to 24 hours. We set $Q = 5, 10, 20$ and select Q based on the *training* likelihood.

Table 6 shows the test log-likelihood of the models on different months. This is the total log-likelihood of all types of events. For HP-LS, we show the best *test* log-likelihood across different h and k. As expected, HP-GS performs the best, confirming the bursty characteristic of the data. However, HP-LS does not perform well. A problem of HP-LS is that the discretization tends to make the influence function noisy and fail to generalize well. CGPRPP has a competitive performance close to HP-GS, showing its capability to model bursty events.

[1] `https://github.com/HongtengXu/Hawkes-Process-Toolkit`

5.5.5 MIMIC Datasets

To show the flexibility of CGPRPP in modeling other complex event patterns than the bursty patterns as in many previously used datasets similar to the IPTV dataset, we derive multiple new event sequence datasets from MIMIC III [48] consisting of lab tests ordered to patients in a hospital. Lab orders tend to have more complex dependencies such as a complex mix of multiple inhibitions and excitations over time (e.g., see Figure 14).

In MIMIC III, there are types of labs that tend to occur together. We collect these labs into groups, which we call lab classes. These classes are built using the following procedure. First, we collect the occurrences of all the labs. Then, we calculate the Intersection over Union (IoU) for each pair of labs based on their occurrence timestamps. That is, if two labs always co-occur, then their IoU will be 1. In contrast, if they never co-occur, then it will be 0. Finally, we put two labs into the same class, if their IoU is above 0.95.

In the experiments, we focus on patients that have been admitted to the hospital. Within these patients, we have 710 types of labs. After grouping them, we get 598 classes. We extract 20 different datasets targeting the most frequent 20 classes. Each dataset consists of 10 different lab classes, one of which is the target we try to predict, while the others are the predictors. The target classes are shown in Table 8. The labels of the labs in the same class are separated by semicolons. For each class, the labs all share the same property (without forcing it) in terms of "fluid" and "category", confirming that our grouping algorithm is reasonable.

To build the predictors for each target lab class, we find 10 different lab classes using heuristics. First, we find the admissions that have at least one occurrence of the target. Then, we calculate the event-wise probability of occurrence and admission-wise probability of occurrence for each class. The former is defined as the number of occurrences for the class divided by the total number of occurrences for all classes. The latter is defined as the number of admissions having at least one occurrence of the class divided by the total number of admissions. The difference between the two is that the former puts the frequency of the class over time into consideration, while the latter only considers the "popularity" of the class among the admissions.

After calculating the two probabilities, we keep only the classes that have an admission-wise probability greater than 0.5. Then we rank these classes by the ratio of the event-wise probabilities of occurrence between the subpopulation of admissions containing at least one target and the whole population, and pick the top 10. The intuition is that the latter probability can be seen as a prior probability of the event occurring, while the former as a posterior probability conditioned on that the target is present in the sequence (admission). Denote the event that, given a lab occurs, it is of the specific class u as E_u, and the event that a lab of the target class $\tilde{u}$ also occurs in the same sequence as $O_{\tilde{u}}$. Then essentially, we iteratively find each predictor u as

$$\arg\max_u \frac{p(E_u|O_{\tilde{u}})}{p(E_u)} = \arg\max_u \frac{p(E_u|O_{\tilde{u}})p(O_{\tilde{u}})}{p(E_u)} = \arg\max_u p(O_{\tilde{u}}|E_u).$$

Using the above heuristics, the target itself will always be selected as the top 1 predictor. Table 9 shows an example of the selected predictors for lab class 355. The first row is the target class itself, followed by the other predictors.

We sample 200 admissions (sequences) randomly from each dataset, where 100 admissions are used for training and the others for testing. For HP-GS, we put the kernels at $0, 8, \ldots, 48$ hours. We also test a different version of the method, HP-GS-A, using the adaptive basis-function-selection algorithm in [114] to place the kernels. For HP-LS, we train multiple models with $h = 0.5, 2, 8$ hours and $k = (48 + 8)/h$ correspondingly. For CGPRPP, the conditional points are also placed at $0, 8, \ldots, 48$ hours. We set $Q = 1, 10$ and select Q based on the *training* likelihood. As a reference, we also test against a model based on deep neural networks, the neural self-modulating multivariate point process (NSMMPP) [80]. The number of hidden units is selected from $64, 128, \ldots, 1024$ as in the original work through a validation set (80/20 split from the full training set).

Table 10 shows the full results of test log-likelihood on the MIMIC datasets. Each column is a different dataset with a different target lab class. They are ordered from the most frequent (355) to the least frequent (18) based on their occurrences. We compare CGPRPP with $Q = 1$ and $Q = 10$, while CGPRPP* is the model selected with the best *training* likelihood. For HP-LS, we show the best *test* log-likelihood across different h and k on each dataset. CGPRPP* achieves the best or close to the best performance on all datasets except class

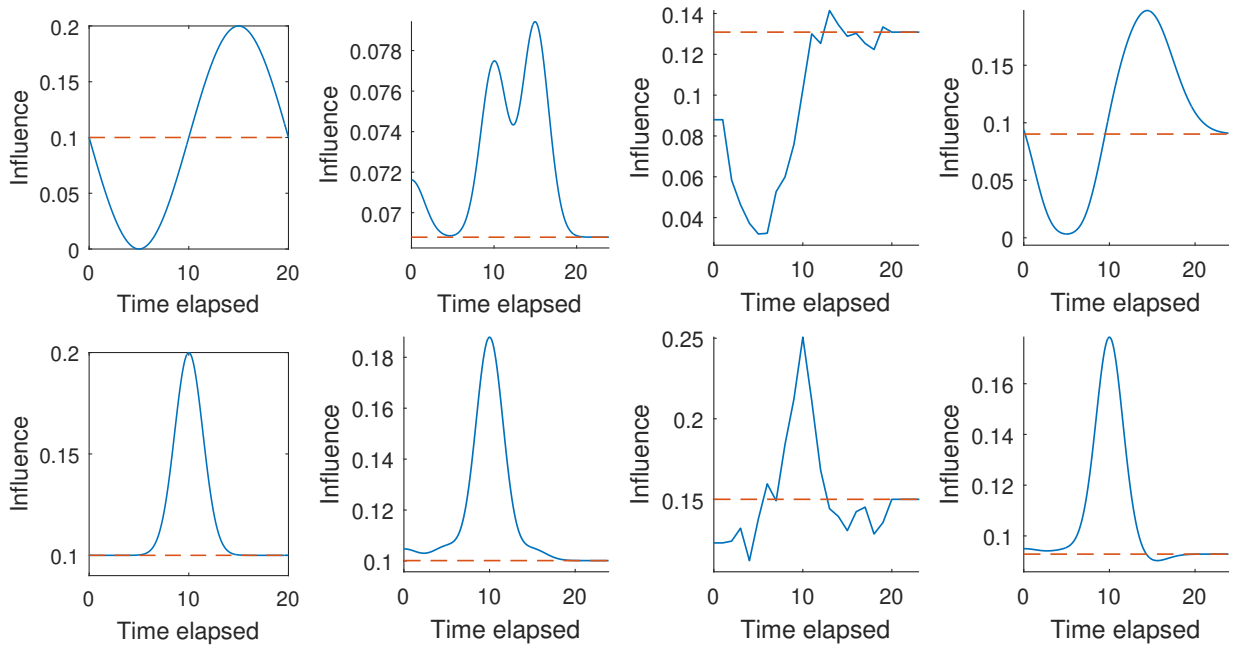
550 and 18. On class 355, 60, 151, 113, and 140, CGPRPP* outperforms the second best by a large margin. In some cases (e.g., class 550) CGPRPP with a different Q actually has a much better result, although not being selected.

As an example, we plot the influence functions for class 355 from past events of the same type in Figure 14. HP-GS learns a smooth influence function with excitations around 24 and 48 hours. This corresponds to the fact that right after a lab being ordered, it might need to be repeated after one or two days. In contrast, HP-LS ($h = 0.5$ with the best test likelihood) learns a much noisier pattern due to discretization, which is harder to interpret. Compared with HP-GS, CGPRPP learns not only similar excitations around 24 and 48 hours, but also a strong inhibition after each excitation, showing a more flexible fit to the data.

5.5.6 Time Prediction Evaluation

We also evaluate the performance of our method for predicting the time of each target event on the MIMIC datasets. On each dataset, we repeat the experiment for each method 5 times and show the average results. The setting of each method is the same as for likelihood evaluation, except for HP-LS, where we only test for $h = 2$. We sample 100 times to estimate the expected time to each next event for all the methods.

We evaluate the accuracy of the time predications using root mean square error (RMSE), where the difference between the predicted time and the true time of each event is calculated. The results are in Table 11. The unit is hour. CGPRPP* has the best or close to best results in most cases, except for lab class 8. In that case, CGPRPP ($Q = 10$) is selected over CGPRPP ($Q = 1$) based on the training likelihood, although the latter has a much better time prediction accuracy on the test data.



(a) Ground truth (b) HP-GS (c) HP-LS (d) CGPRPP

Figure 12: Influences from past events on the first (top) and second (bottom) synthetic datasets. Solid lines are the CIFs after an event. Dashed lines are the baseline intensities. The ground truth is in the first column, followed by the result of each method.

Table 5: Test log-likelihood on synthetic datasets.

Data	HP-GS	HP-LS	CGPRPP
1	-2671	-2770	**-2455**
2	-4074	-4161	**-4071**

Table 6: Test log-likelihood on IPTV dataset.

Month	HP-GS	HP-LS	CGPRPP
1	**-1.477e+05**	-1.779e+05	-1.479e+05
2	-1.509e+05	-1.825e+05	**-1.502e+05**
3	**-1.608e+05**	-1.928e+05	-1.608e+05

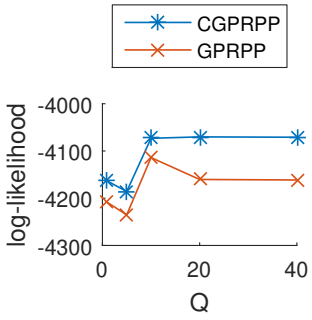


Figure 13: Test log-likelihood of GPRPP based on variational sparse GP (GPRPP) and conditional GP (CGPRPP) with $Q = 1, 5, 10, 20, 40$.

Table 7: IPTV event types and counts.

Type	Count
drama	284092
news	190584
entertainment	122773
others	116449
sports	74502
kids	39712
movie	33437
daily life	33225
economy	23985
law	13636
music	12456
documentary	11162
military	10007
science	6790
education	798
ads	390

Table 8: Target lab classes used for experiments.

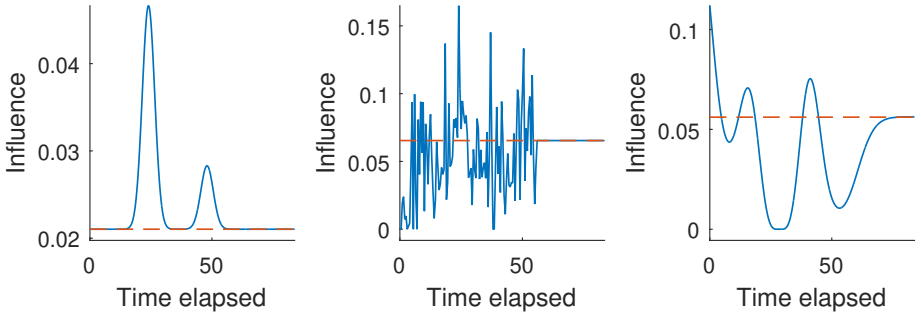
ID	Lab labels	Fluid	Category	Count
355	Hemoglobin; MCH; MCHC; MCV; Platelet Count; RDW; Red Blood Cells; White Blood Cells	Blood	Hematology	4619733
60	Anion Gap; Bicarbonate; Chloride; Sodium	Blood	Chemistry	2500535
3	Base Excess; Calculated Total CO2; pCO2; pO2	Blood	Blood Gas	1942338
95	Creatinine; Urea Nitrogen	Blood	Chemistry	1236906
368	INR(PT); PT	Blood	Hematology	756797
354	Hematocrit	Blood	Hematology	693788
151	Potassium	Blood	Chemistry	669880
550	Bilirubin; Blood; Glucose; Ketone; Leukocytes; Nitrite; Urine Appearance; Urine Color; Urobilinogen	Urine	Hematology	598026
113	Glucose	Blood	Chemistry	595635
140	Magnesium	Blood	Chemistry	559517
294	Basophils; Eosinophils; Lymphocytes; Monocytes; Neutrophils	Blood	Hematology	547408
17	pH	Blood	Blood Gas	524600
150	Phosphate	Blood	Chemistry	489990
80	Calcium, Total	Blood	Chemistry	484701
394	PTT	Blood	Hematology	403567
1	Specimen Type	Blood	Blood Gas	398697
53	Alanine Aminotransferase (ALT); Asparate Aminotransferase (AST)	Blood	Chemistry	296876
7	Free Calcium	Blood	Blood Gas	246208
8	Glucose	Blood	Blood Gas	193253
18	Potassium, Whole Blood	Blood	Blood Gas	187020

Table 9: Predictors selected for lab class 355.

Class ID	Lab labels	Fluid	Category
355	Hemoglobin; MCH; MCHC; MCV; Platelet Count; RDW; Red Blood Cells; White Blood Cells	Blood	Hematology
294	Basophils; Eosinophils; Lymphocytes; Monocytes; Neutrophils	Blood	Hematology
394	PTT	Blood	Hematology
368	INR(PT); PT	Blood	Hematology
140	Magnesium	Blood	Chemistry
113	Glucose	Blood	Chemistry
53	Alanine Aminotransferase (ALT); Asparate Aminotransferase (AST)	Blood	Chemistry
150	Phosphate	Blood	Chemistry
95	Creatinine; Urea Nitrogen	Blood	Chemistry
54	Albumin	Blood	Chemistry

Table 10: Test log-likelihood on MIMIC datasets.

					CGPRPP		
Data	HP-GS	HP-GS-A	HP-LS	NSMMPP	$Q = 1$	$Q = 10$	*selected
355	-3668	-3947	-6510	-3664	**-3249**	-3374	**-3249**
60	-4673	-5051	-7299	-4660	-4246	**-4203**	-4246
3	**-3721**	-3733	-5722	-3737	-3759	-3847	-3759
95	-4064	-4390	-5712	-3982	**-3817**	-3933	-3933
368	-3366	-3711	-5625	**-3309**	-3378	-3538	-3378
354	-4344	-4792	-7185	-4409	-4225	**-3984**	-4225
151	-3338	-3574	-5323	-3763	**-3093**	-3313	**-3093**
550	-1053	-1064	-1744	**-1039**	-1175	-1063	-1175
113	-4656	-5049	-7143	-4539	-4276	**-4142**	-4276
140	-3206	-3475	-4625	-3244	-2942	**-2933**	-2942
294	-1011	-1054	-1308	**-941.2**	-993.6	-1131	-993.6
17	-3783	-3807	-5339	**-3758**	-3808	-4120	-3808
150	-3238	-3537	-4894	-3377	**-3100**	-3144	**-3100**
80	**-3388**	-3772	-5365	-3903	-3402	-3426	-3402
394	-3098	-3251	-4945	-3268	**-3010**	-3127	**-3010**
1	**-3220**	-3291	-3772	-3228	-3234	-3737	-3234
53	-1913	-2138	-2963	-1916	-1900	**-1803**	-1900
7	**-2502**	-2533	-3514	-2626	-2512	-2729	-2512
8	**-1633**	-1667	-3142	-1786	-1694	-1652	-1694
18	-1596	-1678	-3085	**-1532**	-1648	-1817	-1648



(a) HP-GS (b) HP-LS (c) CGPRPP

Figure 14: Influences from past events of the same type as the target class 355 on the MIMIC dataset.

Table 11: RMSE (hour) of time predictions on MIMIC datasets.

					CGPRPP		
Data	HP-GS	HP-GS-A	HP-LS	NSMMPP	$Q = 1$	$Q = 10$	*selected
355	16.43	14.78	16.98	22.06	**13.79**	16.59	**13.79**
60	11.81	10.4	11.19	13.2	**9.956**	10.6	**9.956**
3	13.46	**13.45**	24.75	60.44	19.13	18.85	19.13
95	17.43	17.17	17.36	**15.99**	16.17	17.43	17.43
368	25.32	19.24	31.7	19.71	**17.03**	22.18	**17.03**
354	49.1	50.13	50.46	**46.87**	49.61	48.52	48.52
151	23.36	23.3	23.53	24.12	**22.37**	25.91	**22.37**
550	101.4	96.11	139.3	177	96.97	**91.01**	96.97
113	13.32	11.65	13.4	**9.457**	11.22	10.52	11.22
140	17.45	11.86	11.8	**9.633**	9.892	11.49	9.892
294	98.16	87.78	104.3	185.6	**74.12**	76.23	**74.12**
17	28.19	**28.18**	31.96	29.16	28.63	95.36	28.63
150	22.65	15.76	**15.45**	16.27	15.87	15.86	15.86
80	44.9	44.53	46.39	**43.67**	44.15	44.18	44.15
394	29.76	26	46.24	30.05	25.45	**21.04**	25.45
1	25.85	**25.84**	32.43	26.53	25.97	29.36	25.97
53	51.7	36.41	59.48	38.55	27.84	**27.49**	**27.49**
7	24.02	**22.92**	34	56.59	25.71	24.54	25.71
8	25	**24.64**	63.39	26.08	28.75	65.34	65.34
18	58.99	57.89	86.18	1611	62.2	**56.49**	62.2

6.0 Outlier Detection in Event Sequences

In this chapter, we study the problem of detecting outliers in event sequences in continuous time. We define two new types of outliers corresponding to *unexpected absences* and *unexpected occurrences* of events using generative models based on point processes. Scoring methods for detecting these outliers are developed based on Bayesian decision theory and hypothesis testing with theoretical guarantees.

6.1 Introduction

The focus of this work is on anomaly/outlier detection methods in event sequences. The problem we want to solve is, given the current time and the history of the event sequence, to determine whether the recent occurrence or absence of events is anomalous. There are two types of outliers that may arise here. First, given the history of past events and the recent absence of the events, the event may be overdue. We refer to these as omission outliers. Second, given the history of past events, the event that has just arrived is unexpected in terms of its timing, that is, it has arrived either too early or was not expected at all. We refer to these as commission outliers.

Both types of outliers are often related to problems of practical importance. Take for example, a person suffering from a disease and taking specific medications on a regular schedule to treat the disease. Given the schedule, the history of past events and current time, we may infer that the person has not taken the medication yet and the medication is overdue (omission). The detection of the overdue medication can be then used to generate a reminder alert. Another important problem could be related to communication failures. Assume the messages arrive with some frequency (that may vary to a certain degree), but there is currently a long period of silence. This, when compared to expected/normal behavior, may indicate a communication failure (disconnection), and its detection can be very important in practice. To illustrate the second problem (commission) and its importance, consider a

patient who takes a medication too early compared to the normal schedule. The detection of this event or its prevention (if we have information to detect the intention prior to the administration of the medication) is extremely important and may prevent adverse situations like high concentration of the drug and its possible toxic effects. Similar situations may happen when one receives a medication that is unrelated to his/her condition. The occurrence of this event may indicate a medical error, and once again its timely detection that can prevent or alleviate the consequences of its occurrence is extremely important.

In order to solve the above outlier detection problems, we study models that are able to accurately represent the event sequences, and outlier detection methods based on the models that are able to detect both omission and commission outliers. To build a flexible model of event sequences, we explore general point process models that permit inclusion of context and event history to model the occurrence of the next event. Briefly, the occurrences of events may, in many problems of practical importance, depend on the context. For example, a medication is administered only to a patient who suffers from a specific disease or a condition, so the disease defines a context inducing the occurrences of the events. Similarly, the patient who does not have the disease should not take the medication, so medication administration events under normal circumstances should not occur. Given a point-process model for normal event sequences, we propose and develop probabilistic outlier detection methods based on the distribution defined by the point process. We develop decision rules and scoring methods for detecting these outliers based on Bayesian decision theory and hypothesis testing with theoretical guarantees.

To demonstrate the performance of our outlier detection methods and their abilities to detect outliers in event sequences, we conduct experiments on both synthetic and real-world data. We show that our methods can successfully detect omission and commission outliers, even when the occurrences of events depend on the context, and the context may change dynamically in time.

6.2 Method

6.2.1 Problem Formulation

First, we formally define the problem of contextual outlier detection in continuous-time event sequences. An event sequence can be formulated as $S_{\tilde{u}} = \{(t_i : t_i \in \mathcal{T}\}_{i=1}^{N_{\tilde{u}}}$, i.e., a sequence of timestamps t_i of the events, where t_i is the time of the i-th event, $N_{\tilde{u}}$ is the total number of the events in the sequence, and $\mathcal{T} \subseteq \mathbb{R}$ is the domain of time. We call $S_{\tilde{u}}$ the *target sequence* and the events the *target events*, because they are the targets in which we aim to detect outliers. Meanwhile, we may observe contextual information along with $S_{\tilde{u}}$. We assume the contextual information can be either represented as or converted to discrete events. For example, continuous variables can be converted to discrete events via discretization. We denote these events as $S_{\mathcal{C}} = \{(t_i, u_i) : t_i \in \mathcal{T}, u_i \in \mathcal{C}\}_{i=1}^{N_{\mathcal{C}}}$, where t_i and u_i are the time and type (mark) of the i-th event, $N_{\mathcal{C}}$ is the total number of the events, and $\mathcal{C} \subseteq \mathbb{Z}$ is the finite set of distinct marks for different types of events. We call $S_{\mathcal{C}}$ the *context sequence* and the events the *context events*.

We stress that $S_{\tilde{u}}$ and $S_{\mathcal{C}}$ share the same time domain $\mathcal{T}$ and, therefore, we can combine them into a single sequence $S_{\mathcal{M}} = \{(t_i, u_i) : (t_i, u_i) \in S_{\mathcal{C}} \text{ or } t_i \in S_{\tilde{u}}, u_i = \tilde{u}\}$, where a new type $\tilde{u} \notin \mathcal{C}$ is assigned to all the events in $S_{\tilde{u}}$. For detecting outliers, we only rely on information in the *past* from both $S_{\tilde{u}}$ and $S_{\mathcal{C}}$. We denote the combined history of $S_{\tilde{u}}$ and $S_{\mathcal{C}}$ up to time t as $\mathcal{H}_t^{\mathcal{M}} = \{(t_i, u_i) : (t_i, u_i) \in S_{\mathcal{M}}, t_i < t\}$. Meanwhile, $\mathcal{H}_t$-$t = \{t_i : t_i \in S_{\tilde{u}}, t_i < t\}$ is the history of the target sequence $S_{\tilde{u}}$ only without any contextual information.

Now, we are ready to define two types of outlier detection problems we want to solve. The first one is to detect commission outliers (unexpected events). Given an observed target event at time t_n, and the combined history $\mathcal{H}_{t_n}^{\mathcal{M}}$ of the target sequence $S_{\tilde{u}}$ and the context sequence $S_{\mathcal{C}}$ up to time t_n, the goal is to assign a label $y_c(t_n) \in \{0, 1\}$ to t_n indicating whether it is a commission outlier. Notice that $y_c(t)$ is only defined if t is the time of a target event. In this work, instead of hard labels, we consider outputting a commission outlier score $s_c(t_n)$ for t_n to indicate how likely it is a commission outlier.

The second problem is to detect omission outliers. We define a blank interval $B \subseteq \mathcal{T}$ as

an interval in which there is no event of type $\tilde{u}$, i.e., no event from the target event sequence $S_{\tilde{u}}$. Given a blank interval $B = (t_b, t_e)$ and the combined history $\mathcal{H}_{t_b}^{\mathcal{M}}$ of the target sequence $S_{\tilde{u}}$ and the context sequence $S_{\mathcal{C}}$ up to time t_b, the goal is to assign a label $y_o(B) \in \{0, 1\}$ to B indicating whether there are *any* omission outliers in B. Notice that $y_o(B)$ is only defined when B is a blank interval for $S_{\tilde{u}}$ (no *target* event within). In this work, instead of hard labels, we consider outputting an omission outlier score $s_o(B)$ for B to indicate how likely it contains any omission outliers.

6.2.2 Probabilistic Models

We develop algorithms for detecting outliers in continuous-time event sequences based on probabilistic models, specifically (temporal) point processes. Point processes are probabilistic models for discrete points in continuous domains. For a continuous-time event sequence, the points are the events, and the domain is the time $\mathcal{T}$. In this case, the models are also called *temporal* point processes. A temporal point process can be defined as a counting process $N(\cdot)$ on $\mathcal{T}$, where $N(\tau)$ is the number of points in the interval $\tau \subseteq \mathcal{T}$. We make the common assumption that *at most one event can happen at a given time.*

For a temporal point process, the conditional intensity function (CIF), $\lambda(t)$, characterizes the probability of observing an event in an infinitesimal time interval $[t, t + dt) \subseteq \mathcal{T}$ given the history up to time t. That is

$$\lambda(t)dt = p(N([t, t + dt)) = 1|\mathcal{H}_t). \tag{6.1}$$

For our problem, we only model the *target* events using a point process, while the history $\mathcal{H}_t = \mathcal{H}_t^{\mathcal{M}}$ contains both the target events and the context events. Because $\lambda(t)$ is conditioned on $\mathcal{H}_t$ by definition, we omit $\mathcal{H}_t$ for the rest of the paper and always condition on it implicitly.

For a sequence of target events $S_{\tilde{u}} = \{t_i : t_i \in \mathcal{T}\}_{i=1}^{N_{\tilde{u}}}$ generated from the point process with CIF $\lambda(t)$ (conditioned on the combined history $\mathcal{H}_t^{\mathcal{M}}$), the probability density is

$$p(S_{\tilde{u}}) = \prod_{i=1}^{N_{\tilde{u}}} \lambda(t_i) \exp\left(-\int_{\mathcal{T}} \lambda(s)ds\right). \tag{6.2}$$

An intuitive interpretation of the equation is that for the observed events at time $t_i, i = 1, \ldots, N_{\tilde{u}}$, $\lambda(t_i)dt$ is the probability of observing the events at those specific time points. Meanwhile, $\exp\left(-\int_{\mathcal{T}} \lambda(s)ds\right)$ corresponds to the fact that there are no events at any other time points in $\mathcal{T}$.

When detecting outliers, we assume that we already have a point-process model for the target events in *normal* cases. The model may be specified by an expert or, more generally, learned from existing data. If the model is learned from data, we assume that either the training data is outlier-free or that the outliers in the training data are insignificant for learning a model to detect outliers in the test data.

In general, the model should be able to represent the dependencies between the target events and the context events. For a point-process model, it means that instead of a CIF that only depends on the target events, $\lambda(t) = f(\mathcal{H}_{t^-}t)$, it has a CIF that also depends on the context events, $\lambda(t) = f(\mathcal{H}_t^{\mathcal{M}})$, where $f(\cdot)$ denotes the mapping represented by the model. In this work, we use a flexible model adapted from the continuous-time LSTM [80], which we briefly describe in the next section.

6.2.3 Continuous-Time LSTM with Context

The input to the continuous-time LSTM consists of the marked events in the combined sequence, $(t_i, u_i) \in S_{\mathcal{M}}$. That is, we not only use the target events but also the context events as input, although we only model the CIF of the target events, $\lambda(t)$. The output consists of the hidden states $\boldsymbol{h}(t_i)$ corresponding to the input. It is a nonlinear mapping from the content in the memory cell $\boldsymbol{c}(t_i)$ of the LSTM at time t_i. As in a traditional LSTM, each continuous-time LSTM unit also has an input gate $\boldsymbol{i}$, an output gate $\boldsymbol{o}$, and a forget gate $\boldsymbol{f}$. The relations between the memory cells, the hidden states, the input, and these gates are summarized as follows.

Let $\boldsymbol{u}_i$ be a vector representation of the mark u_i, which is a learnable embedding. For $t \in (t_{i-1}, t_i]$, $\boldsymbol{c}(t)$ is a continuous function changing over time from $\boldsymbol{c}_i$ to $\bar{\boldsymbol{c}}_i$, and for $\boldsymbol{c}_i$ and $\bar{\boldsymbol{c}}_i$

there are separate input gates and forget gates:

$$\boldsymbol{h}(t) = \boldsymbol{o}_i \odot \tanh(\boldsymbol{c}(t)) \tag{6.3}$$

$$\boldsymbol{c}(t) = \bar{\boldsymbol{c}}_i + (\boldsymbol{c}_i - \bar{\boldsymbol{c}}_i) \exp\left(-\boldsymbol{\delta}_i (t - t_{i-1})\right) \tag{6.4}$$

$$[\boldsymbol{i}_{i+1}; \boldsymbol{o}_{i+1}; \boldsymbol{f}_{i+1}] = \sigma(\boldsymbol{W}\boldsymbol{u}_i + \boldsymbol{U}\boldsymbol{h}(t_i) + \boldsymbol{d}) \tag{6.5}$$

$$[\bar{\boldsymbol{i}}_{i+i}; \bar{\boldsymbol{f}}_{i+1}] = \sigma(\bar{\boldsymbol{W}}\boldsymbol{u}_i + \bar{\boldsymbol{U}}\boldsymbol{h}(t_i) + \bar{\boldsymbol{d}}) \tag{6.6}$$

$$\boldsymbol{z}_{i+1} = \tanh(\boldsymbol{W}_z\boldsymbol{u}_i + \boldsymbol{U}_z\boldsymbol{h}(t_i) + \boldsymbol{d}_z) \tag{6.7}$$

$$\boldsymbol{c}_{i+1} = \boldsymbol{f}_{i+1} \odot \boldsymbol{c}(t_i) + \boldsymbol{i}_{i+1} \odot \boldsymbol{z}_{i+1} \tag{6.8}$$

$$\bar{\boldsymbol{c}}_{i+1} = \bar{\boldsymbol{f}}_{i+1} \odot \bar{\boldsymbol{c}}_i + \bar{\boldsymbol{i}}_{i+1} \odot \boldsymbol{z}_{i+1} \tag{6.9}$$

$$\boldsymbol{\delta}_{i+1} = g(\boldsymbol{W}_\delta\boldsymbol{u}_i + \boldsymbol{U}_\delta\boldsymbol{h}(t_i) + \boldsymbol{d}_\delta, 1) \tag{6.10}$$

where $[\boldsymbol{a}; \boldsymbol{b}]$ denotes the concatenation of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\odot$ is the elementwise product, $\sigma(\cdot)$ is the logistic function, and $g(x, s) = s\log(1 + \exp(x/s))$ is the scaled softplus function with parameter s. All the $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{d}$ with/without different subscripts and bars are learnable parameters of the continuous-time LSTM.

Finally, to convert the output of the continuous-time LSTM to the CIF of the target events, $\lambda(t)$, we have $\lambda(t) = g(\boldsymbol{w}_\lambda^T \boldsymbol{h}(t), s)$ where $\boldsymbol{w}_\lambda$ and s are learnable parameters. The model is learned by maximizing the likelihood (Eq. 6.2) for all sequences in the training data. Monte-Carlo integration is used to evaluate $\int \lambda(s)ds$.

6.2.4 Detecting Commission Outliers

To derive an outlier scoring method for commission outliers (unexpected events), we first describe a generative process for defining normal points and outliers. Then based on the generative process, we derive a Bayes decision rule, from which we derive the scoring method for commission outliers.

Suppose we are given a target event t_n (and the history up to time t_n, $\mathcal{H}_{t_n}^{\mathcal{M}}$). Define a random variable Z_n, such that $Z_n = 1$ if t_n is a commission outlier, and $Z_n = 0$ otherwise. We are interested in calculating $p(Z_n = 1|t_n)$.

Assume the process that generates outliers is independent from the process that generates normal points. Then, the generative process for the normal points and outliers can be viewed

together as a marked point process. That is, for each event t_n there is a hidden mark Z_n associated indicating whether it is an outlier. The overall CIF $\lambda_g(t) = \lambda_1(t) + \lambda_0(t)$, where $\lambda_0(t)$ is the CIF for the normal point process, and $\lambda_1(t)$ for the point process that generates outliers.

Based on the definition of the CIF of a marked point process

$$p(t_n) = \lambda_g(t_n) \exp\left(-\int_{t_c}^{t_n} \lambda_g(t)dt\right), \tag{6.11}$$

$$p(Z_n = 1, t_n) = \lambda_1(t_n) \exp\left(-\int_{t_c}^{t_n} \lambda_g(t)dt\right). \tag{6.12}$$

From the above, we can derive the conditional distribution

$$p(Z_n = 1|t_n) = \frac{\lambda_1(t_n)}{\lambda_g(t_n)} = 1 - \frac{\lambda_0(t_n)}{\lambda_g(t_n)}. \tag{6.13}$$

Therefore, the Bayes decision rule is

$$Z_n^* = \arg\max_z p(Z_n = z|t_n) = \mathbb{I}\left[\lambda_1(t_n) > \lambda_0(t_n)\right] \tag{6.14}$$

where $\mathbb{I}[x] = 1$ if x is true, and 0 otherwise. However, this rule cannot be directly applied, because λ_1 is unknown. Assuming λ_1 is a constant, the decision rule becomes $Z_n^* = \mathbb{I}\left[\lambda_0(t_n) < \theta_c\right]$, where θ_c is a threshold. This justifies ranking by

$$s_c(t_n) = -\lambda_0(t_n) \tag{6.15}$$

across all $n = 1, \ldots, N_{\tilde{u}}$, so we use $-\lambda_0(t_n)$ as the commission outlier score: the higher the score, the more likely t_n is a commission outlier.

6.2.5 Detecting Omission Outliers

To derive an outlier scoring method for omission outliers, we first describe a generative process. Based on the process, we derive a Bayes decision rule, from which we derive the scoring method for omission outliers. Finally, we provide an alternative justification for the scoring method based on hypothesis testing.

Assume we have generated a sequence of normal points with the normal CIF $\lambda_0(t)$. To generate omission outliers, we assume that each point can be removed independently with probability p_1. After the removal, we have a new sequence of points with (unobservable) omission outliers. Then, given any blank interval B, we can derive the probability of at least one removal occurring in the interval.

To derive the method, we first define some notations. For any interval $\tau \subseteq \mathcal{T}$, let $N(\tau)$ be the number of points observed, and $N_0(\tau)$ be the number of points generated by the normal point process with CIF $\lambda_0(t)$, so $N(\cdot)$ is the result of combining $N_0(\cdot)$ with random removal, and we can observe $N(\cdot)$ but not $N_0(\cdot)$. Furthermore, we define an auxiliary random variable that counts the number of points removed in a blank interval B as K_B.

For any blank interval B, we observe $N(B) = 0$, but $K_B = k$ can take different values $k = 0, 1, \ldots$. The joint probability for each k is

$$\begin{aligned} p(K_B = k, N(B) = 0) =& p(K_B = k, N_0(B) = k) \\ =& p_1^k F_k(B) \end{aligned} \tag{6.16}$$

where $F_k(B)$ denotes the probability that k points are generated by the normal point process $N_0(\cdot)$ in B for $k = 0, 1, \ldots$. They depend on the normal CIF $\lambda_0(t)$. Then the posterior probability of $K_B = 0$ can be calculated as

$$p(K_B = 0 | N(B) = 0) = \frac{F_0(B)}{\sum_{k=0}^{\infty} p_1^k F_k(B)} \tag{6.17}$$

Define a random variable Z_B to indicate whether there are any omission outliers in the blank interval B: $Z_B = 0$ is equivalent to $K_B = 0$; $Z_B = 1$ is equivalent to $K_B > 0$.

$$\begin{aligned} p(Z_B = 1 | N(B) = 0) =& 1 - p(Z_B = 0 | N(B) = 0) \\ =& 1 - p(K_B = 0 | N(B) = 0). \end{aligned} \tag{6.18}$$

Then the Bayes decision rule is

$$\begin{aligned} Z_B^* &= \arg\max_z p(Z_B = z | N(B) = 0) \\ &= \mathbb{I}\left[p(K_B = 0 | N(B) = 0) < 0.5\right]. \end{aligned} \tag{6.19}$$

Without further assumptions, $p(K_B = 0|N(B) = 0)$ (Eq. 6.17) cannot be evaluated in closed form, but we can get a lower bound

$$p(K_B = 0|N(B) = 0) \geq F_0(B) = \exp\left(-\int_B \lambda_0(s)ds\right) \tag{6.20}$$

because

$$\sum_{k=0}^{\infty} p_1^k F_k(B) \leq \sum_{k=0}^{\infty} F_k(B) = 1.$$

Then the posterior probability of B containing any omission outliers can also be bounded

$$p(Z_B = 1|N(B) = 0) \leq 1 - \exp\left(-\int_B \lambda_0(s)ds\right). \tag{6.21}$$

Therefore, we propose to use

$$s_o(B) = \int_B \lambda_0(s)ds \tag{6.22}$$

as the omission outlier score. When we rank the blank intervals by $s_o(B)$, we essentially rank them by an upper bound of $p(Z_B = 1|N(B) = 0)$.

There is a notable special case where we can get a closed-form $p(K_B = 0|N(B) = 0)$. That is, if the normal point process $N_0(\cdot)$ is an inhomogeneous Poisson process, then

$$F_k(B) = \frac{\left(\int_B \lambda_0(s)ds\right)^k}{k!} \exp\left(-\int_B \lambda_0(s)ds\right) \tag{6.23}$$

for $k = 0, 1, \ldots$. The posterior becomes

$$\begin{aligned} p(K_B = 0|N(B) = 0) &= \frac{F_0(B)}{\sum_{k=0}^{\infty} p_1^k F_k(B)} \\ &= \exp\left(-p_1 \int_B \lambda_0(s)ds\right). \end{aligned} \tag{6.24}$$

Therefore, the posterior probability of B containing any omission outliers is

$$p(Z_B = 1|N(B) = 0) = 1 - \exp\left(-p_1 \int_B \lambda_0(s)ds\right). \tag{6.25}$$

This justifies scoring the interval B by $s_o(B) = \int_B \lambda_0(s)ds$, because if we rank the intervals by their scores, the result will be the same as ranking by their posterior probabilities of containing omission outliers, $p(Z_B = 1|N(B) = 0)$.

Without assuming that the process is an inhomogeneous Poisson process, an alternative justification for using $\int_B \lambda_0(s)ds$ as the omission outlier score can be given based on hypothesis testing for inter-event time, i.e., the time interval between two consecutive events t_{n-1} and t_n. Let T_n be the random variable for the inter-event time. Assume B is an observed *inter-event* interval. The null and alternative hypotheses are

$$H_0 : B \text{ is normal}; \quad H_1 : B \text{ contains omission outliers.}$$

Assuming the null hypothesis is true, i.e., B is an inter-event interval generated by the normal point process with CIF $\lambda_0(t)$, the probability that the inter-event time is at least as long as $|B|$ is

$$p(T_n > |B|) = \exp\left(-\int_B \lambda_0(s)ds\right) \tag{6.26}$$

which is the p-value. A lower p-value means that the observation is more extreme, given that the null hypothesis is true, which means it is more likely to contain omission outliers. This justifies scoring by $\int_B \lambda_0(s)ds$, where a higher score means that B is more likely to contain omission outliers.

6.2.6 Bounds on FDR and FPR

In this section, we prove some bounds on the performance of the proposed outlier scoring methods. We recall the definitions of false discovery rate (FDR) and false positive rate (FPR). Let y denote the true label (1=outlier, 0=normal) of an object (a target event or a blank interval) and $\hat{y}$ denote the predicted label. Then FDR and FPR are defined as

$$FDR = p(y = 0|\hat{y} = 1), \quad FPR = p(\hat{y} = 1|y = 0).$$

Given the above definitions, we can prove the following theorems.

Theorem 6.2.1. *If we use the commission outlier score $s_c(t_n) = -\lambda_0(t_n)$, where t_n is the time of a target event, with a threshold $\theta_c \leq 0$, such that the decision rule is $\hat{y}_c(t_n) = \mathbb{I}[s_c(t_n) > \theta_c]$, and let λ_1 denote the CIF of the independent process generating commission outliers, then the FDR is bounded above by $\frac{-\theta_c}{\lambda_1 - \theta_c}$.*

Proof. From Eq. 6.13 and implicitly conditioned on the event t_n and the history

$$p(y_c(t_n) = 0) = p(Z_n = 0) = \frac{\lambda_0(t_n)}{\lambda_0(t_n) + \lambda_1}$$

Given that $\hat{y}_c(t_n) = 1$, i.e., $-\lambda_0(t_n) > \theta_c$, we get

$$p(y_c(t_n) = 0 | \hat{y}_c(t_n) = 1) < \frac{-\theta_c}{-\theta_c + \lambda_1}$$

□

Theorem 6.2.2. *If we use the omission outlier score $s_o(B) = \int_B \lambda_0(s)ds$ for an* inter-event *interval B, with a threshold $\theta_o \geq 0$, such that the decision rule is $\hat{y}_o(B) = \mathbb{I}[s_o(B) > \theta_o]$, then the FPR is bounded above by $\exp(-\theta_o)$.*

Proof. Let T_n be the random variable for the inter-event time corresponding to the observed inter-event interval B, assuming it is generated from the normal point process. From Eq. 6.26

$$\begin{aligned}
&p(\hat{y}_o(B) = 1 | y_o(B) = 0) \\
=&p\left(\int_B \lambda_0(s)ds > \theta_o \middle| y_o(B) = 0\right) \\
=&p\left(\exp\left(-\int_B \lambda_0(s)ds\right) < \exp(-\theta_o) \middle| y_o(B) = 0\right) \\
=&p\left(p(T_n > |B|) < \exp(-\theta_o)\right) \\
=&\exp(-\theta_o)
\end{aligned}$$

The last equality is justified by the fact that $p(T_n > |B|) = 1 - p(T_n \leq |B|)$, and $p(T_n \leq |B|)$ is the cumulative distribution function of T_n, implying it follows a uniform distribution. □

Theorem 6.2.3. *If we use the omission outlier score $s_o(B) = \int_B \lambda_0(s)ds$ for a blank interval B, with a threshold $\theta_o \geq 0$, such that the decision rule is $\hat{y}_o(B) = \mathbb{I}[s_o(B) > \theta_o]$, and assume that the normal point process is an inhomogeneous Poisson process and the probability of omission is p_1, then the FDR is bounded above by $\exp(-p_1\theta_o)$.*

Proof. From Eq. 6.25 and implicitly conditioned on $N(B) = 0$ and the history

$$p(y_o(B) = 0) = p(K_B = 0) = \exp\left(-p_1 \int_B \lambda_0(s)ds\right)$$

Given that $\hat{y}_o(B) = 1$, i.e., $\int_B \lambda_0(s)ds > \theta_o$, we get

$$p(y_o(B) = 0|\hat{y}_o(B) = 1) < \exp\left(-p_1\theta_o\right)$$

□

6.3 Experiments

We perform experiments on both synthetic and real-world event sequences. First, we briefly describe the compared methods. Next, we conduct experiments on synthetic data. Finally, we experiment with real-world clinical data.

6.3.1 Compared Methods

We compare the following methods in the experiments. **RND**: A baseline that generates outlier scores by sampling from a uniform distribution on $[0, 1]$. **LEN**: A baseline that detects outliers based on the empirical distribution of the inter-event time lengths. **PPOD** (Point-Process Outlier Detection): Our method based on a point-process model but only using the history of the target events as the context. **CPPOD** (Contextual Point-Process Outlier Detection): Our method based on a point-process model using the history of both the target events and the context events as the context. **GT** (Ground Truth): Our method using the *ground-truth* point-process model to calculate the outlier scores (only available on synthetic data).

Next we briefly describe the method **LEN**. For training, the lengths of all the inter-event time of the target events, $L = \{l_i : l_i = t_{i+1} - t_i, (t_i, \tilde{u}), (t_{i+1}, \tilde{u}) \in S_{\tilde{u}}\}$ are collected. Then, an empirical distribution of the inter-event time can be formulated as $\hat{F}(l) = \frac{1}{|L|}\sum_{i=1}^{|L|} \mathbb{I}\left[l_i \leq l\right]$. Here, for simplicity, we describe the method as if we only had one sequence in the training

data, but it is easy to see how it works for multiple sequences, which is the case in our experiments.

For testing, **LEN** outputs a commission outlier score for a target event at time $t_n \in S_{\tilde{u}}$ as $s_c(t_n) = -\min\{\hat{F}(t_n - t_{n-1}), 1 - \hat{F}(t_n - t_{n-1})\}$ where $t_n - t_{n-1}$ is the inter-event time between the current and previous target events. Intuitively, if the inter-event time is too small ($\hat{F}(\cdot)$ is small) or too big ($1 - \hat{F}(\cdot)$ is small), it is likely that t_n has occurred at an abnormal time (too early or too late) and therefore is a commission outlier. The negation makes sure that a higher score indicates that it is more likely to be an outlier. For a blank interval B, **LEN** outputs an omission outlier score as the length of B, $s_o(B) = |B|$. Intuitively, the longer the blank interval, the more likely it contains omission outliers.

PPOD and **CPPOD** rely on the continuous time LSTM [80] to model the CIF. We choose the number of hidden units in the model from $\{64, 128, 256, 512, 1024\}$ by maximizing the likelihood on the internal validation set that consists of 20 percent of the training set. We stress that for training and validation we do not use any labeled outlier data.

6.3.2 Experiments on Synthetic Event Sequences

We generate synthetic event sequences using two different types of point processes. One is the inhomogeneous Poisson process. The other is the Gamma process. For each type of processes, there is a set of parameters that determine the distribution of the points. We allow the parameters to vary according to a context state x.

To keep things simple, we allow two different values for the state $x \in \{0, 1\}$. Associated with each value of the state is a set of values of the parameters for the point process.

For the inhomogeneous Poisson process, the CIF is a piecewise constant function with the value $\lambda = f(x)$, where x is the context state. In the experiments, we set $f(0) = 0.1$ and $f(1) = 1$.

For the Gamma process, the inter-event time follows a Gamma distribution $Gam\,(a_x, b_x)$ (a_x shape, b_x rate), where x is the context state. In the experiments, we set $(a_0, b_0) = (10, 10)$ and $(a_1, b_1) = (100, 10)$.

The changes of the context state x are driven by a continuous-time Markov chain with a

transition matrix $Q = \begin{bmatrix} -0.05 & 0.05 \\ 0.05 & -0.05 \end{bmatrix}$ such that

$$p\left(x(t+dt) = j | x(t) = i\right) = \mathbb{I}\left[i = j\right] + Q_{ij} dt$$

where dt is infinitesimal time. Each change of the state generates a context event.

For each point process type, we simulate 40 sequences. Each sequence is simulated in the same time range $\mathcal{T} = [0, 1000]$. We use 50 percent of the sequences for training and the others for testing.

6.3.2.1 Simulation of Commission and Omission Outliers To define outliers, we simulate commission and omission outliers on top of the existing data. In this way, we can obtain ground-truth labels for testing.

To define commission outliers, we simulate a new sequence of target events independently from the existing data, and then merge the new events with the existing events. We use a Poisson process with a parameter λ_c to generate the outliers. λ_c controls the rate of such outliers. In the experiments, for each dataset, we set $\lambda_c = \alpha \hat{\lambda}_{test}$, where $\alpha = 0.1$ and $\hat{\lambda}_{test}$ is the empirical rate of the target events calculated from the original test data. We also vary α to study the effect.

To define omission outliers, we randomly remove target events in the original sequences according to independent Bernoulli trials. That is, each event is removed with probability p_1 and kept with probability $1 - p_1$. We always keep the event if it marks the start time of the sequence. In the experiments, we set $p_1 = \alpha$ and $\alpha = 0.1$. We also vary α to study the effect.

6.3.2.2 Detection of Commission and Omission Outliers We detect the presence of commission and omission outliers differently. To test for commission outliers, each method outputs an outlier score at the time of each target event. That is, whenever there is a new target event, we ask the question: is this event a commission outlier or not?

Testing for omission outliers is trickier, because we need to decide the checkpoints more carefully, i.e., when to ask for outlier scores. The simplest thing to do is to only check at the

target event times. That is, whenever there is a new target event, we ask the question: is there any omission outlier starting from the previous target event till now?

However, this may become unsatisfactory in real-world applications, because there could be cases when the target events just stop occurring for a long period of time or even forever (potentially due to malfunctions of the underlying system). These are interesting and important cases we are supposed to detect, but the above testing method will not work. Therefore, we use a combined approach. We still have a checkpoint at each target event time, but on top of that, we also randomly generate checkpoints in long blank intervals.

Specifically, we have a parameter w set to $2/\hat{\lambda}_{train}$, where $\hat{\lambda}_{train}$ is the empirical rate of the target events estimated from the training data for each dataset, so within w, on average, we should see two events normally. Then, whenever the blank interval from the previous checkpoint till now is longer than w, we generate a new checkpoint within the interval by uniform sampling, and set the previous checkpoint to the generated checkpoint. We keep generating checkpoints until we reach the next target event or the end of the sequence.

6.3.2.3 Results Figure 15 and 16 show the receiver operating characteristic (ROC) curves of the outlier detection methods on the synthetic data generated from inhomogeneous Poisson processes and Gamma processes. The curves of **GT** and **CPPOD** are almost identical.

Both **GT** and **CPPOD** achieve the best performance for both commission and omission outliers, showing the effectiveness of our outlier scoring methods. The fact that **CPPOD** almost has the same performance as **GT** is an evidence that the model based on the continuous-time LSTM is flexible enough to represent these different processes that generate the data. **PPOD** being worse than **CPPOD** shows the importance of the context events in these cases. Although **LEN** performs much better than **RND**, it is worse than our proposed methods, because it is based on the empirical distribution of inter-event times instead of a flexible model that can capture the dependencies of the target events on themselves and on the context events.

We vary α for simulating commission and omission outliers, and see its effect. Table 12 shows the AUROC of the methods. As we can see, changing α does not affect the advantage and disadvantage of each method.

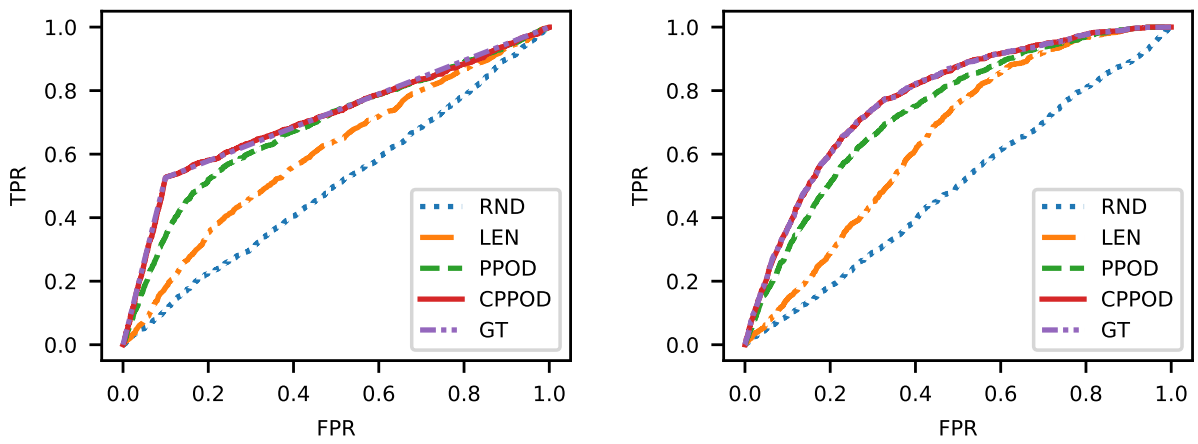


Figure 15: ROC curves on synthetic data (Poisson process). Left: commission. Right: omission.

6.3.2.4 Empirical Verification of the Bounds on FDR and FPR

To empirically verify the bounds on FDR and FPR as presented in Section 6.2.6, we randomly repeat the experiments using **GT** on the synthetic data 10 times, with the same training data but different test data. Each time, we calculate the FDR and FPR for different thresholds on the scores. For verifying FPR, we only test the inter-event time intervals for omission outliers. Their means and standard deviations over all repetitions are shown with the theoretical bounds in Figure 17 and 18. For FPR, the bounds overlap the empirical rates. Figure 19 shows the FDR (omission outlier) with means and standard deviations on data simulated from inhomogeneous Poisson processes with the theoretical bound. As we can see, the FDR has high variance when the threshold is high, because there are fewer samples above a higher threshold. Nonetheless, the empirical FDR conforms with the bound.

6.3.3 Experiments on Real-World Clinical Data

In this part, we use real-world clinical data derived from MIMIC III [48]. MIMIC III consists of de-identified electronic health records of ICU patients. We pick four types of events as our targets and form four separate datasets by collecting the target events and corresponding context events. The target events and their context events are listed in the

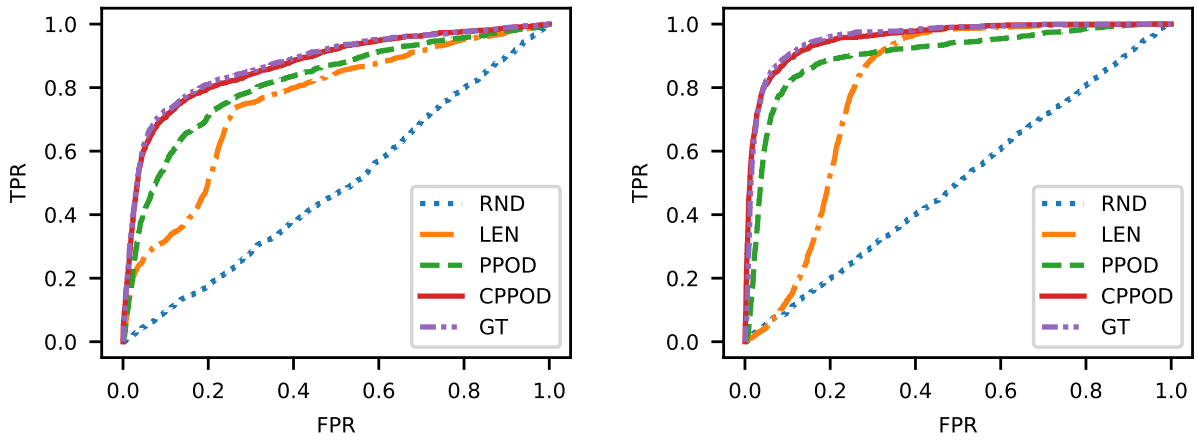


Figure 16: ROC curves on synthetic data (Gamma process). Left: commission. Right: omission.

Table 13. The medical category (medication, lab, or vital sign) of each type of events is in brackets following the type. For example, Potassium Chloride is a type of medications, and Potassium (Blood) is a type of lab tests. The latter is used as the context for the former, as the administration of the medication can be triggered by observing an abnormally low value in the lab test.

For every event type in the table, we record that type in the sequence data. However, for Potassium (Blood) and Total Calcium (Blood), we further split the context events into three subtypes depending on whether the value in the lab test is low, normal, or high. For INR(PT) (previous state), we create context events of two subtypes based on whether the value of the *previous* event is normal or abnormal (with a 1-second delay). For Arterial Blood Pressure systolic (ABPs) and Non-invasive Blood Pressure systolic (NBPs), we split the events into two subtypes depending on whether the value is normal or low. These subtypes help us define better the contexts influencing the target events, since depending on their values, the target events can be more/less likely to occur.

All target and context events for one patient admission form one event sequence. For the first three datasets (first three targets), we have randomly selected 2000 sequences. For the last one, we randomly selected 500 sequences, because each sequence contains much more

Table 12: AUROC on synthetic data. First column: dataset name abbreviation (C=commission, O=omission) [α].

Dataset	**RND**	**LEN**	**PPOD**	**CPPOD**
Poi (C) [0.1]	0.500 (± 0.010)	0.601 (± 0.008)	0.684 (± 0.010)	**0.711** (± 0.012)
Poi (C) [0.05]	0.493 (± 0.011)	0.627 (± 0.011)	0.684 (± 0.014)	**0.716** (± 0.019)
Poi (O) [0.1]	0.503 (± 0.008)	0.650 (± 0.006)	0.737 (± 0.006)	**0.778** (± 0.005)
Poi (O) [0.05]	0.491 (± 0.018)	0.650 (± 0.008)	0.736 (± 0.007)	**0.776** (± 0.009)
Gam (C) [0.1]	0.485 (± 0.007)	0.754 (± 0.006)	0.816 (± 0.008)	**0.871** (± 0.006)
Gam (C) [0.05]	0.479 (± 0.018)	0.776 (± 0.011)	0.840 (± 0.010)	**0.897** (± 0.006)
Gam (O) [0.1]	0.505 (± 0.012)	0.799 (± 0.005)	0.901 (± 0.007)	**0.956** (± 0.003)
Gam (O) [0.05]	0.503 (± 0.013)	0.803 (± 0.009)	0.919 (± 0.008)	**0.961** (± 0.007)

events than the previous three. For each dataset, we use 50 percent of the sequences for training and the others for testing.

We generate commission and omission outliers on top of the existing sequences with the same processes described for synthetic data. This allows us to obtain ground-truth labels for analyses. Similarly, we detect commission and omission outliers using the same approaches applied to synthetic data.

6.3.3.1 Results Table 14 shows the AUROC of the methods for the datasets derived from MIMIC data. The results have more variations across different datasets in this case, which can be seen by examining the performance of **LEN**. Omission outliers appear to be more challenging than commission outliers except for INR(PT) lab test. Comparing the methods, **CPPOD** and **PPOD** outperform **RND** and **LEN** on all the datasets for both commission and omission outliers.

In all cases, **CPPOD** is either the best or very close to the best. In the latter cases, the best method is always **PPOD**, and the difference is very small. These are the cases where the additional context events are not as influential as the history of the target events themselves

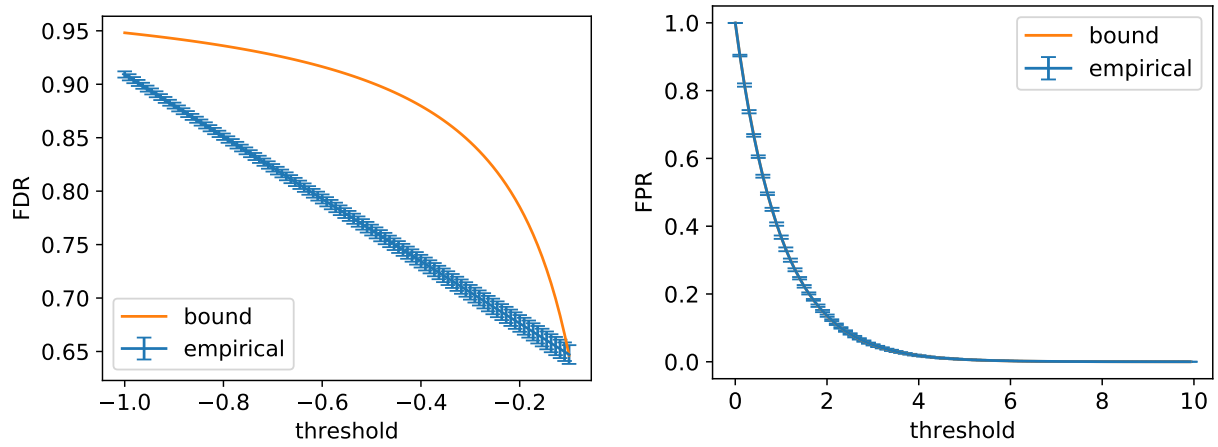


Figure 17: FDR (commission outlier) and FPR (omission outlier) on synthetic data (Poisson process).

for the occurrences of the target events, so **PPOD** is as good as but simpler than **CPPOD**. However, for Potassium Chloride and Calcium Gluconate, we can see a clear advantage of **CPPOD** over **PPOD** by using additional context events.

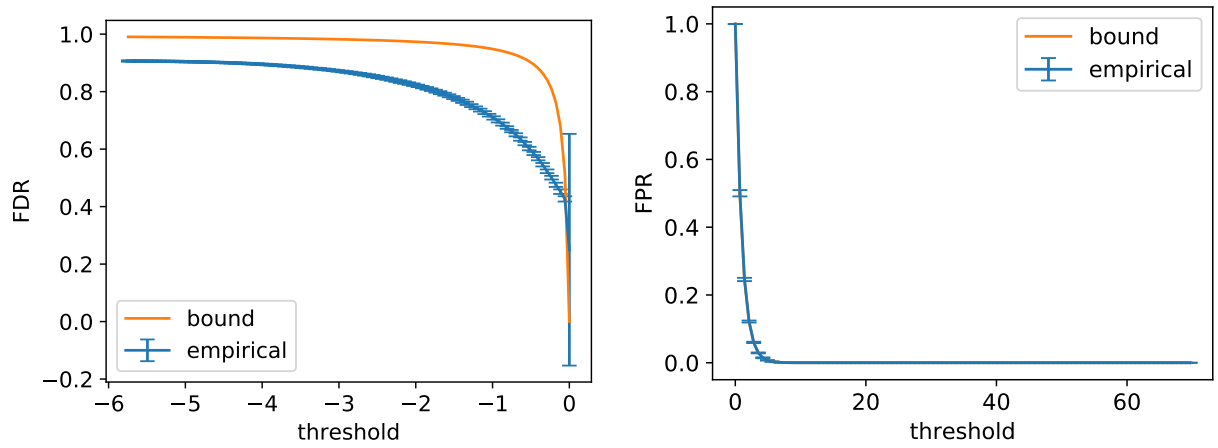


Figure 18: FDR (commission outlier) and FPR (omission outlier) on synthetic data (Gamma process).

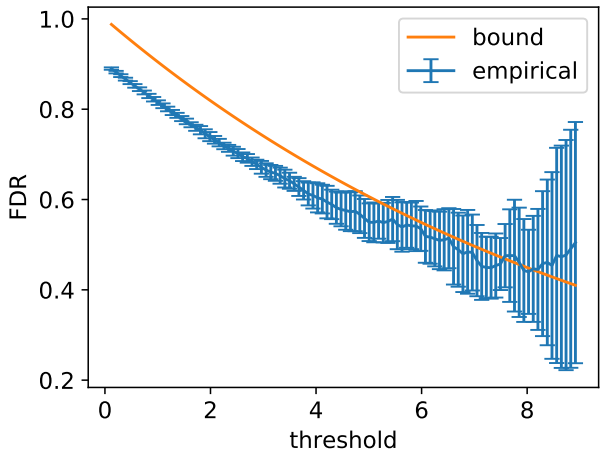


Figure 19: FDR (omission outlier) on synthetic data (Poisson process).

Table 13: Names of target and context events from MIMIC. INR=international normalized ratio; PT=prothrombin time.

Target	Context
INR(PT) [Lab]	INR(PT) [Lab] (previous state); Heparin [Medication]; Warfarin [Medication]
Calcium Gluconate [Medication]	Total Calcium (Blood) [Lab]
Potassium Chloride [Medication]	Potassium (Blood) [Lab]
Norepinephrine [Medication]	Arterial Blood Pressure systolic [Vital Sign]; Non-invasive Blood Pressure systolic [Vital Sign]

Table 14: AUROC on MIMIC data. First column: target name abbreviation (C=commission, O=omission) [α].

Dataset	**RND**	**LEN**	**PPOD**	**CPPOD**
INR (C) [0.1]	0.496 (± 0.010)	0.596 (± 0.009)	0.682 (± 0.010)	**0.687** (± 0.009)
INR (C) [0.05]	0.486 (± 0.014)	0.613 (± 0.018)	**0.702** (± 0.014)	0.701 (± 0.018)
INR (O) [0.1]	0.498 (± 0.011)	0.726 (± 0.008)	**0.747** (± 0.009)	0.746 (± 0.010)
INR (O) [0.05]	0.487 (± 0.013)	0.736 (± 0.011)	0.779 (± 0.012)	**0.782** (± 0.012)
Cal (C) [0.1]	0.504 (± 0.013)	0.739 (± 0.012)	0.830 (± 0.010)	**0.866** (± 0.006)
Cal (C) [0.05]	0.470 (± 0.020)	0.753 (± 0.017)	0.843 (± 0.012)	**0.885** (± 0.010)
Cal (O) [0.1]	0.493 (± 0.016)	0.526 (± 0.009)	0.759 (± 0.009)	**0.775** (± 0.008)
Cal (O) [0.05]	0.513 (± 0.021)	0.531 (± 0.014)	0.759 (± 0.014)	**0.761** (± 0.014)
Pot (C) [0.1]	0.498 (± 0.012)	0.733 (± 0.013)	0.839 (± 0.009)	**0.878** (± 0.009)
Pot (C) [0.05]	0.488 (± 0.020)	0.707 (± 0.016)	0.827 (± 0.012)	**0.878** (± 0.009)
Pot (O) [0.1]	0.495 (± 0.017)	0.533 (± 0.012)	0.735 (± 0.011)	**0.749** (± 0.011)
Pot (O) [0.05]	0.503 (± 0.015)	0.539 (± 0.014)	0.727 (± 0.015)	**0.744** (± 0.015)
Nor (C) [0.1]	0.494 (± 0.014)	0.864 (± 0.010)	0.890 (± 0.012)	**0.897** (± 0.013)
Nor (C) [0.05]	0.506 (± 0.013)	0.868 (± 0.014)	0.899 (± 0.013)	**0.899** (± 0.013)
Nor (O) [0.1]	0.510 (± 0.010)	0.468 (± 0.016)	**0.835** (± 0.010)	0.832 (± 0.009)
Nor (O) [0.05]	0.506 (± 0.023)	0.489 (± 0.018)	**0.830** (± 0.013)	0.826 (± 0.012)

7.0 Change-Point Detection in Event Sequences

In this chapter, we study the problem of detecting change-points in event sequences, focusing on using (temporal) point process models. We develop methods based on the discretized-time approach and the continuous-time approach respectively. By studying their properties, we draw important conclusions on these two approaches with practical implications. Finally, through experiments, we verify these properties and compare the performance of the methods.

7.1 Introduction

Point processes are widely-used generative models for event sequences, which consist of discrete events in continuous time. As in traditional time series models, there might be changes over time in the model generating the data. In reality, these changes could reflect abnormal or novel events occurring in the underlying system that need human attention, so it could be very valuable to detect these changes automatically and send alerts to relevant people accordingly.

To detect change-points in event sequences, theoretically there are two possible approaches:

- Discretize the time by binning the events into counts across time and apply change-point detection methods for count time series.
- Keep using continuous time and detect change-points based on point processes.

It is natural to assume that the first approach based on time discretization, while being seemingly simple, will lose information and therefore result in worse performance in detection. In contrast, the second approach should in general perform better, because it utilizes complete information in the data. We develop a specific method based on the second approach, and compare it with a method based on the first approach derived from change-point detection method for count time series.

7.2 Problem Statement

Assume that the data we observe are events occurring continuously over a period of time $\mathcal{T} \in \mathbb{R}$, so they can be represented as a sequence of points in $\mathcal{T}$ as $S = \{t_i : t_i \in \mathcal{T}\}_{i=1}^{N}$. Given the observed data, we wish to know whether there is any change in the underlying model and correspondingly the system generating the data.

Point processes are models that generate discrete points in continuous domains (such as time). For a (temporal) point process, its conditional intensity function (CIF), denoted as $\lambda(t)$, defines the rate of events (or points) occurring at each instant t given the history of the sequence $\mathcal{H}_{t^-} = \{t_i \in S : t_i < t\}$:

$$\lambda(t)dt = \mathbb{E}\left[N([t, t+dt))|\mathcal{H}_{t^-}\right] \tag{7.1}$$

where $N(\cdot)$ is the count of points within any sub-interval in $\mathcal{T}$, and dt is an infinitesimal amount of time.

For different types of point processes, their CIFs are different. For example, for a homogeneous Poisson process, the CIF at any instant is a constant that is not affected by the time or the previous events

$$\lambda(t) = \mu \tag{7.2}$$

where $\mu > 0$ is a constant. For a Hawkes process, however, the CIF is influenced by the previous events

$$\lambda(t) = \mu + \sum_{t_i<t} \phi(t - t_i) \tag{7.3}$$

where $\mu > 0$ is a constant, called the baseline intensity, t_i is the time of the i-th events before t, and $\phi(\cdot)$ is a nonnegative function, called the triggering kernel (not to be confused with kernels in Gaussian processes), characterizing the influences from the previous events over time on $\lambda(t)$. A simple example of the triggering kernel that is widely used is the exponential kernel

$$\phi(t) = \alpha\beta \exp\left(-\beta t\right), \quad t > 0. \tag{7.4}$$

Notice that the triggering kernels are defined only for $t > 0$, while for $t \leq 0$ they are assumed to be 0.

Given the CIF of a point process, the probability density of the sequence S generated from the process is

$$p(S) = \prod_{i=1}^{N} \lambda(t_i) \int_{\mathcal{T}} \lambda(t)dt. \tag{7.5}$$

In this work, we focus on the problem of detecting baseline shifts in the CIF $\lambda(t)$ for Poisson processes. That is, we wish to detect changes in μ. Given that the point process is a Poisson process, this is equivalent to detecting any changes in the CIF. We note that the goal of this work is to study the difference and connection between discretized-time change-point detection approach and continuous-time change-point detection approach, which is why we focus on Poisson processes.

7.3 Method

7.3.1 Detecting Change-Points in Discretized Time

We briefly explain how to detect change-points in *discretized* time, which also provides some intuitions for how to detect change-points in *continuous* time. First. let Δ be the bin size. Without loss of generality, assume that the time domain $\mathcal{T} = (0, T]$ and $T = |\mathcal{T}|$ is a multiplier of Δ. Then, we can discretize the time domain $\mathcal{T}$ into $M = \frac{|\mathcal{T}|}{\Delta}$ bins, $\{(0, \Delta], (\Delta, 2\Delta], \ldots, ((M-1)\Delta, M\Delta]\}$.

For the observed sequence of events $S = \{t_i : t_i \in \mathcal{T}\}_{i=1}^{N}$, we can count the number of points within each bin as

$$y_j = \sum_{i=1}^{N} \mathbb{I}\left[t_i \in ((j-1)\Delta, j\Delta)\right] \tag{7.6}$$

for $j = 1, 2, \ldots, M$, where $\mathbb{I}[x] = 1$ if x is true and otherwise 0. Then $Y = \{y_j\}_{j=1}^{M}$ form a time series of counts (integers). If the original event sequence S is generated from a Poisson process, then each y_j will follow a Poisson distribution $Pois\left(\int_{(j-1)\Delta}^{j\Delta} \lambda(t)dt\right)$, where the mean of the Poisson distribution is equal to the integral of the CIF within the corresponding time bin. When the CIF is a constant, $\lambda(t) = \mu$, every y_j follows the same Poisson distribution $Pois\left(\mu\Delta\right)$.

The change-point in Y can be defined as the index c such that the counts before c, $Y_{c^-} = \{y_j : j <= c\}$, and the counts after c, $Y_{c^+} = \{y_j : j > c\}$, follow the Poisson distributions $Pois(\mu_1)$ and $Pois(\mu_2)$ respectively, while $\mu_1 \neq \mu_2$. If we have a method to find c in the discretized time domain (indexes), then we can convert it back to the continuous time domain as

$$t_c = c\Delta \tag{7.7}$$

where t_c is the suspected change-point corresponding to c in the continuous time domain. Apparently, by going through the above process, we inevitably lose some accuracy in the location of the change-point, since t_c can only be a multiplier of Δ.

To find the change-point, we start from, given an index k, deciding whether k is a change-point, or equivalently, whether the counts before and after k follow the same distribution. For this purpose, we can design a likelihood ratio statistic

$$l_k = \frac{p(Y_{k^-}|\hat{\mu}_1)p(Y_{k^+}|\hat{\mu}_2)}{p(Y|\hat{\mu}_0)} \tag{7.8}$$

where $\hat{\mu}_1$, $\hat{\mu}_2$, and $\hat{\mu}_0$ are the maximum likelihood estimates of the means of the Poisson distributions fit to Y_{k^-}, Y_{k^+}, and Y respectively. Since $2\ln l_k$ asymptotically follows a Chi-square distribution with one degree of freedom $\chi^2(1)$, we can perform a hypothesis test to decide whether k is a change-point.

The intuition behind the likelihood ratio statistic is that when k is really a change-point, the distributions before and after k are expected to be different, such that trying to fit a single distribution with μ_0 will result in much lower likelihood compared with fitting two separate distributions, and therefore l_k should be large. Alternatively, if it is not a change-point, then the numerator and denominator in l_k should be similar, and therefore l_k should be small (close to 1).

Based on the intuition of the likelihood ratio statistic, the larger l_k is, the more likely it is a change-point. Therefore, we can estimate the location of the potential change-point as

$$\hat{c} = \underset{k=1,\ldots,M-1}{\arg\max}\ l_k \tag{7.9}$$

and decide whether $\hat{c}$ is a true change-point based on $l_{\hat{c}}$.

7.3.2 Detecting Change-Points in Continuous Time

For detecting change-points in the baseline intensity in *continuous* time, we draw our intuition from the previous section (with discretized time) using the likelihood ratio statistic. Specifically, instead of considering the change-point to be a discrete index, $c \in \{1, \ldots, M-1\}$, we consider it to be in the original continuous time domain, $\tau_c \in \mathcal{T}$. From Eq. 7.8 and 7.9, we derive their corresponding continuous-time versions

$$l(\tau) = \frac{p(S_{\tau^-}|\hat{\mu}_1)p(S_{\tau^+}|\hat{\mu}_2)}{p(S|\hat{\mu}_0)}, \tag{7.10}$$

and

$$\hat{\tau}_c = \arg\max_{\tau \in \mathcal{T}} l(\tau), \tag{7.11}$$

where $S_{t^-} = \{t_i \in S : t_i <= t\}$ and $S_{t^+} = \{t_i \in S : t_i > t\}$.

It may seem that we have already solved the problem, but notice that it is not clear how to efficiently perform the optimization in Eq. 7.11, because $\hat{\mu}_1$, $\hat{\mu}_2$, and τ depend on each other. When we move τ, we may get different estimates for $\hat{\mu}_1$ and $\hat{\mu}_2$, so we cannot fix $\hat{\mu}_1$ and $\hat{\mu}_2$ when we optimize w.r.t. τ. Furthermore, $\hat{\mu}_1$ and $\hat{\mu}_2$ depend on τ *discontinuously* as the densities take the form in Eq. 7.5.

We develop a method based on a continuous approximation of the likelihood ratio. To achieve that, we first develop a model that allows a change-point in the baseline intensity, while the location of the change-point is treated as a parameter of the model. Let b be that parameter. We define the CIF as

$$\lambda(t) = \mu_1\sigma_1(t) + \mu_2\sigma_2(t), \tag{7.12}$$

where $\mu_1 > 0, \mu_2 > 0$, and

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{7.13}$$

$$\sigma_1(t) = \sigma(w(b-t)), \tag{7.14}$$

$$\sigma_2(t) = \sigma(w(t-b)). \tag{7.15}$$

If we set w to a large number, $\sigma_1(x) \approx \mathbb{I}[t < b]$ and $\sigma_2(t) \approx \mathbb{I}[t > b]$. Meanwhile, it always holds that $\sigma_1(t) + \sigma_2(t) = 1$. Therefore, the above CIF is essentially a mixture of two baselines μ_1 and μ_2, and there is a "soft" switch at time b.

The benefit of the mixture model, compared with dealing with the original likelihood ratio statistic, is that we can optimize jointly all the parameters, including the location of the change-point b, since the likelihood is a continuous function of the parameters. Specifically, we try to maximize the following function

$$\ell(\theta) = \ln(p(S_{b^-}|\mu_1)) + \ln(p(S_{b^+}|\mu_2)) \tag{7.16}$$

where $\theta = (\mu_1, \mu_2, b)$. This is equivalent to maximizing Eq. 7.10 after taking a logarithm, because the denominator in Eq. 7.10 does not depend on θ.

One more issue that we need to resolve is local optima. Because the original likelihood ratio statistic can have many discontinuous points, the continuous approximation can also have many local optima. To deal with this issue, we start from K randomly chosen initial points when we perform the optimization, get K local optimal results $\Theta = \{\hat{\theta}_k\}_{k=1}^K$, and pick the best result $\hat{\theta} = \arg\max_{\theta \in \Theta} l(\theta)$ as our final estimate.

7.4 Properties of the Methods in Poisson Processes

In this section, we study in detail the properties of the likelihood-ratio-based change-point detection methods in Poisson processes. We develop closed-form solutions to the maximum likelihood estimation (MLE) of the intensities for both discretized-time or continuous-time versions.

7.4.1 Discretized-Time Estimation

For discretized time, the log-likelihood for a sequence of counts $Y = \{y_j\}_{j=1}^M$ is

$$\ell(\mu) = \left(\sum_{j=1}^{M} y_j\right) \ln \mu - \mu M - \sum_{j=1}^{M} \ln(y_j!), \tag{7.17}$$

so the MLE for the parameter μ is

$$\hat{\mu} = \frac{1}{M} \sum_{j=1}^{M} y_j. \tag{7.18}$$

Plug it into Eq. 7.8, we have

$$l_k = N_1 \ln \frac{N_1}{M_1} + N_2 \ln \frac{N_2}{M_2} - N \ln \frac{N}{M} \tag{7.19}$$

where

$$N_1 = \sum_{y_j \in Y_{k^-}} y_j, \quad N_2 = \sum_{y_j \in Y_{k^+}} y_j, \quad N = N_1 + N_2 \tag{7.20}$$

$$M_1 = k, \quad M_2 = M - k. \tag{7.21}$$

That is, N_1 is the number of points with time $t \leq k\Delta$, M_1 is the corresponding number of bins, and N_2, M_2 are similarly the opposite.

To study the effect of the bin size Δ, we map the discretized time back into the original continuous time. Let $\tau_k = k\Delta$ be the location of the potential change-point k in continuous time, and $T = M\Delta$ be the total length of time. Then from the above equation we have

$$\begin{aligned} l(\tau_k) &= l_k \\ &= N_1 \ln N_1 - N_1 \ln \frac{\tau_k}{\Delta} + N_2 \ln N_2 - N_2 \ln \frac{T - \tau_k}{\Delta} - N \ln N + N \ln \frac{T}{\Delta} \\ &= N_1 \ln N_1 - N_1 \ln \tau_k + N_2 \ln N_2 - N_2 \ln(T - \tau_k) - N \ln N + N \ln T. \end{aligned} \tag{7.22}$$

All the terms directly involving the bin size Δ have been canceled out. Meanwhile, for a given sequence, when we try to estimate the location of the change-point from Eq. 7.9, the total number of points N and the total time length T are fixed, so the estimate of the change-point only depends on the number of points N_1 and N_2 on each side of the change-point, but *it is independent of the bin size* Δ. This is a somehow surprising result, since we would have assumed the bin size has some influence on the likelihood ratio statistic.

Theorem 7.4.1. *For an event sequence of time length T generated by a Poisson process with a change-point and a given time $t \in (0, T)$, assuming both T and τ are multipliers of the bin size Δ used for time discretization, then the likelihood ratio statistic $l(\tau)$ as in Eq. 7.10 and the corresponding discretized-time version $l_{\tau/\Delta}$ as in Eq. 7.8 are independent of Δ.*

This result has practical implications for choosing the bin size when we try to detect change-points in Poisson processes using time discretization. Specifically, it implies that a smaller bin size is always better as long as we have the capability to collect data with such a fine time resolution in the first place and enough computational resources for storing and processing the data.

7.4.2 Continuous-Time Estimation

For continuous time, the log-likelihood for a sequence of points $S = \{t_i : t \in (0, T]\}_{i=1}^{N}$ generated from a Poisson process with intensity λ in the interval $(0, T]$ is

$$\ell(\mu) = N \ln \lambda - \lambda T, \tag{7.23}$$

so the MLE for the parameter λ is

$$\hat{\lambda} = \frac{N}{T}. \tag{7.24}$$

Plug it into Eq. 7.10, we have

$$l(\tau) = N_1 \ln \frac{N_1}{T_1} + N_2 \ln \frac{N_2}{T_2} - N \ln \frac{N}{T}, \tag{7.25}$$

where

$$N_1 = \sum_{i=1}^{N} \mathbb{I}\left[t_i \leq \tau\right], \quad N_2 = \sum_{i=1}^{N} \mathbb{I}\left[t_i > \tau\right], \tag{7.26}$$

$$T_1 = \tau, \quad T_2 = T - \tau. \tag{7.27}$$

The equation is similar to the corresponding discretized-time version, except that the discrete time indexes (integers) are replaced by the continuous time indexes (real numbers).

Because N_1 and N_2 depends on τ and the points in S, to analyze the properties of $l(\tau)$, we focus on the segment between two consecutive points t_i and t_{i+1} in S, where N_1 and N_2 are fixed. For $\tau \in [t_i, t_{i+1})$,

$$\frac{\partial l}{\partial \tau} = -\frac{N_1}{\tau} + \frac{N_2}{T - \tau}, \tag{7.28}$$

$$\frac{\partial^2 l}{\partial \tau^2} = \frac{N_1}{\tau^2} + \frac{N_2}{(T - \tau)^2} \geq 0. \tag{7.29}$$

Therefore, within $[t_i, t_{i+1})$, $l(\tau)$ is convex, and overall in $(0, T]$, $l(\tau)$ is piecewise convex with jumps at the points in S.

Theorem 7.4.2. *For an event sequence S generated from a Poisson process, the likelihood ratio statistic defined in Eq. 7.10 is a piecewise convex function with jumps at the points $t_i \in S$.*

To find the change-point estimate using Eq. 7.11, we find the maximum of $l(\tau)$ wrt τ. From the above theorem, we have the following corollary.

Corollary 7.4.2.1. *For an event sequence S generated from a Poisson process, a solution to the change-point estimation (Eq. 7.11) must be either at an event time t_i or its immediate left $t_i - \epsilon$, where ϵ is infinitesimal.*

From the corollary, we can design a very efficient algorithm for change-point estimation (Algorithm 1). The time complexity of the algorithm is $O(N)$.

Algorithm 1 Change-point estimation

Input: Event sequence $S = \{t_i\}_{i=1}^N$; a small constant ϵ representing infinitesimal.

Output: Change-point estimate $\hat{\tau}_c$.

1: $\hat{\tau}_c = t_1$ ▷ Initialize $\hat{\tau}_c$

2: **for** $\tau_k \in \{t_i\}_{i=1}^N \cup \{t_i - \epsilon\}_{i=1}^N$ **do** ▷ Check every point and its immediate left

3: $\quad \hat{\tau}_c \leftarrow \arg\max_{\tau \in \{\hat{\tau}_c, \tau_k\}} l(\tau)$

4: **end for**

5: **return** $\hat{\tau}_c$

7.5 Experiments

7.5.1 Experiment Setup

We perform experiments to compare different methods we developed and discussed in the previous sections. The methods we compare include

- **DT**: Discretized-time change-point detection with **different bin sizes**.
- **C-Mix**: continuous-time change-point detection by optimizing the likelihood ratio using a mixture model.

- **C-Sol**: continuous-time change-point detection by solving the optimization problem in closed-form as in Algorithm 1.

We repeat the experiments randomly for multiple times. Each time, to generate the data, we simulate an event sequence of length $T = 10^5$ from a randomly generated Poisson process with a change-point sampled from the uniform distribution $Unif(\frac{1}{3}T, \frac{2}{3}T)$. The intensities before and after the change-point are sampled from $Unif(0, 0.3)$ and $Unif(0.7, 1)$ with a random order. This makes sure we have enough difference in the intensity before and after the change-point. We also tested on uniformly drawing intensity in $Unif(0, 1)$ both before and after the change-point. Although all the conclusions from the likelihood results still hold, the change-point distance results have high variances to draw meaningful conclusions.

After generating the data, each method is independently applied to the same sequence. For discretized-time change-point detection, we test against 10 different bin sizes $10^3/2^1$, $10^3/2^2$, ..., $10^3/2^{10}$, where the minimum bin size is less than 1. For **C-Mix**, because of local optima, we repeat the optimization 200 times with initial values of the parameters randomly sampled from uniform distributions in the corresponding domains of the parameters ($(0, T)$ for the change-point and $(0, 1)$ for the intensities).

7.5.2 Results

First, we compare the performance of the methods by comparing the likelihood ratios achieved in the end for 5 random experiments. Table 15 shows the results. We can see a clear trend in discretized-time change-point detection: as we decrease the bin size, the performance tends to get better (the likelihood ratio gets higher). However, **C-Sol** always has (slightly) better results than even the finest bin size we tested.

C-Mix achieves results better than using large bins on some sequences but worse results on the others. This is most likely due to the large amount of local optima making the optimization procedure struggle to find the global optimum, since **C-Sol** does not have this issue. Comparing **C-Mix** and **C-Sol**, the latter achieves better results than the former on all the sequences. On top of that, we note that **C-Sol** is a more efficient algorithm, since it does not rely on general optimization procedures but instead uses a closed-form solution.

Next, we compare both the likelihood ratio and the absolute distance between the estimate change-point and true change-point, $|\hat{\tau}_c - \tau_c|$, for each method in each sequence by repeating the experiment randomly for 100 times and computing the median, mean, and standard deviation. The results are in Table 16 and 17. We did not compare with **C-Mix**, since it showed inferior performance than **C-Sol** in the previous experiments.

For likelihood ratios, we can draw the same conclusions as in the previous small sample we tested on. As the bin size gets smaller, we see a clear trend that the discretized-time approach gets better performance. However, **C-Sol** achieves even better performance compared with discretized time, and the difference is significant (Wilcoxon signed-rank test $\alpha = 0.0001$).

For change-point distances, the discretized-time approach also tends to perform better when the bin size gets smaller. When the bin size is small enough, the standard deviation becomes small, and the performance becomes stable. **C-Sol** achieves similar performance compared to the best result achieved across all the bin sizes. Using the smallest bins ($10^3/2^{10}$ and $10^3/2^9$) achieves slightly better results than **C-Sol**, although the difference between **C-Sol** and $10^3/2^{10}$ or $10^3/2^9$ is not statistically significant (Wilcoxon signed-rank test $\alpha = 0.05$).

7.6 Discussion

In this work, we have studied two approaches to detect change-points in event sequences: discretized-time approach and continuous-time approach. We have derived specific methods for Poisson processes for both approaches and studied their properties. The methods are based on likelihood ratios. For continuous-time approach, we propose to use a mixture model to represent the intensity, such that the change-point can be estimated as part of the model parameters jointly with the other parameters.

For discretized-time approach, we found a somehow surprising result: the bin size does not directly affect the likelihood ratio we use for detecting change-points, which implies that in theory a smaller bin will always result in a higher likelihood ratio.

For continuous-time approach, we found that the global maximum of the likelihood ratio must be located near an event. Using this result, we developed a more efficient algorithm

than the mixture-model-based method to estimate the change-point in continuous time.

There are connections between the two approaches as shown in the similarity between the equations of the likelihood ratios. When comparing the performance in the experiments, the continuous-time approach achieves (slightly) better performance in terms of the likelihood ratio and similar performance in terms of the error in the estimated change-point, compared with the best performance achieved across all bin sizes for discretized-time approach. For larger bins, the discretized-time approach generally gets worse performance with large amount of variance. Given the efficiency of the improved continuous-time algorithm ($O(N)$ where N is the number of events in the sequence), we argue that the continuous-time approach should be the better choice in general, since there is no need to choose a good bin size to get good performance. The advantage of the discretized-time approach is in the ability of changing the bin size to control the trade-off between accuracy (smaller bins) and efficiency (larger bins). Also, in cases when there are restrictions on computational resources such that we have to use time discretization, it is generally better to use a bin size as small as the resources allow to get stable and good performance.

Table 15: Likelihood ratios on five simulated event sequences with change-points.

		Sequence				
Method	Bin size	1	2	3	4	5
DT	$10^3/2^1$	25633.057	9154.329	13702.079	12402.339	27994.639
	$10^3/2^2$	25647.890	9154.329	13818.393	12402.339	28212.430
	$10^3/2^3$	25754.951	9154.329	13818.393	12402.339	28299.918
	$10^3/2^4$	25778.447	9171.827	13822.594	12402.339	28299.918
	$10^3/2^5$	25778.447	9176.786	13837.159	12402.339	28299.918
	$10^3/2^6$	25784.063	9177.537	13842.360	12402.339	28312.799
	$10^3/2^7$	25784.063	9177.537	13842.888	12402.339	28319.975
	$10^3/2^8$	25784.814	9179.397	13842.888	12402.599	28323.564
	$10^3/2^9$	25785.876	9179.397	13843.792	12403.866	28323.564
	$10^3/2^{10}$	25786.783	9179.397	13843.792	12404.499	28324.461
C-Mix		25775.464	9141.904	13753.221	12098.199	28066.927
C-Sol		25787.152	9179.608	13844.305	12404.658	28325.051

Table 16: The medians, means, and standard deviations of the likelihood ratios on simulated event sequences with change-points.

Method	Bin size	median	mean	std
DT	$10^3/2^1$	13584.850	14659.352	5963.812
	$10^3/2^2$	13643.007	14706.218	5989.542
	$10^3/2^3$	13653.604	14728.415	6005.381
	$10^3/2^4$	13656.237	14737.715	6008.402
	$10^3/2^5$	13670.651	14742.104	6010.383
	$10^3/2^6$	13673.251	14744.494	6012.020
	$10^3/2^7$	13673.515	14745.738	6012.482
	$10^3/2^8$	13673.515	14746.396	6012.701
	$10^3/2^9$	13673.967	14746.862	6012.849
	$10^3/2^{10}$	13673.967	14747.170	6012.935
C-Sol		13674.493	14747.499	6012.964

Table 17: The medians, means, and standard deviations of the distances between the estimated change-points and the true change-points on simulated event sequences.

Method	Bin size	median	mean	std
DT	$10^3/2^1$	122.321	146.862	101.855
	$10^3/2^2$	64.774	70.966	51.229
	$10^3/2^3$	29.041	32.759	24.799
	$10^3/2^4$	13.460	16.223	12.115
	$10^3/2^5$	7.570	8.293	5.877
	$10^3/2^6$	5.167	5.979	5.187
	$10^3/2^7$	2.297	3.803	4.004
	$10^3/2^8$	2.060	3.387	3.954
	$10^3/2^9$	1.231	3.229	4.257
	$10^3/2^{10}$	1.168	2.865	4.054
C-Sol		1.309	2.932	3.934

8.0 Conclusion and Future Work

In this dissertation, we studied methods for event time series prediction and anomaly detection. In general, we studied two highly-related problems: predictive modeling of event time series (or event sequences) and anomaly detection in event time series. They are highly related, because a model of the time series can be used to perform anomaly detection, and the quality of the model directly affects the performance of the anomaly detection.

Meanwhile, in terms of methodology, we have developed methods according to two general approaches:

1. **Discretized time**: convert the event time series into regular time series of counts and perform any inference task on the regular time series instead.
2. **Continuous time**: model the event time series directly and make any inference in continuous time.

The advantage of the first approach (discretized time) is that we can utilize existing research in statistics related to time series models. By adapting and applying these models to account for important properties of the converted time series, such as nonstationarity, we developed anomaly detection algorithms with better performance compared with traditional methods. Chapter 3 and 4 follow the first approach.

The advantage of the second approach (continuous time) is that we do not lose any information compared with the first approach. By using the accurate timing information of each event, we developed better models that can predict the occurrence of each event, defined new types of anomalies related to the timing of each event, and developed algorithms that can detect these new types of anomalies or that can detect an existing type of anomalies with better accuracy. Chapter 5, 6, and 7 follow the second approach.

In Chapter 3 and 4, we developed anomaly detection methods for regular time series of counts. To model the time series accurately, we utilized models based on nonparametric decomposition (ND) and dynamic linear model (DLM). A key property that we tried to take into account for modeling these time series is nonstationarity, especially trends and seasonality.

Both ND and DLM allow us to achieve this goal with some adaptations. We developed methods for detecting two types of anomalies, temporary outliers and change-points, based on these models. Experiments on synthetic and real data showed that our methods can detect both types of anomalies better than methods based on traditional statistical models.

In Chapter 5, we proposed a new nonparametric method for modeling dependencies between events in event sequences using Gaussian processes. Similar to Hawkes processes and different from previous GP-modulated point processes, the proposed model can be learned on a sample of sequences and then applied to other unseen sequences. However, we showed that the proposed model is more flexible than state-of-the-art nonparametric Hawkes process variants. It can learn the dependencies between events that are common in practice but difficult for the Hawkes process variants to represent, e.g., a mix of inhibitions and excitations after an event. Our method showed competitive or better performance on different datasets compared with the Hawkes process variants.

In Chapter 6, we studied the new problem of detecting commission and omission outliers in event sequences. These are new types of outliers that we defined in continuous time. We proposed outlier scoring methods based on Bayesian decision theory and hypothesis testing with theoretical guarantees. The proposed methods depend on a probabilistic model for normal data. While any point-process model can be plugged in, we used a model adapted from the continuous-time LSTM that can consider the contexts of the events. Experiments on both synthetic and real-world event sequences showed the flexibility of the adapted model and, more importantly, the effectiveness of the proposed outlier scoring methods.

In Chapter 7, we studied and compared two approaches to detect change-points in event sequences: discretized-time approach and continuous-time approach. We developed specific methods for Poisson processes for both approaches and studied their properties. Although for both approaches, the methods are based on likelihood ratios, the algorithms for change-point detection are quite different. In the experiments, we verified a result we found through studying the theoretical properties: for the discretized-time approach, a smaller bin size always gives the same or better result than a larger bin size. Furthermore, we showed that the new algorithm we developed for the continuous-time approach, which has a linear time complexity, can achieve the same or better performance than the discretized-time approach,

without the need to pick a bin size, which makes it a better choice in practice.

We note that the work presented in the dissertation is not free from limitations, and anomaly detection in continuous-time event sequences is a relatively new direction that may still need more exploration. Here we list these limitations and potential future work.

- For anomaly detection in time series using the discretized-time approach, the methods based on nonparametric decomposition use an existing algorithm, STL [15], to perform the decomposition of time series. However, the algorithm can only handle univariate time series. Extension of STL to multivariate time series could be a challenging task for future work. Meanwhile, the method based on dynamic linear models is relatively easier to extend for multivariate time series, but dealing with the increased size of the latent state could pose a challenge.
- When we model the time series, one important property of the time series considered is seasonality. In our models, based on either nonparametric decomposition or dynamic linear models, the seasonal period is assumed to be known. This is usually not hard to achieve, if we can see a clear pattern when plotting the data, or there is domain knowledge to help us decide it. However, in cases when it is hard to decide the period or even to decide whether there is seasonality, we may rely on approaches from Fourier analysis to decide the periodicity in the data from the frequency domain. Therefore, combining frequency-domain approaches with our anomaly detection methods for practical applications could be an interesting direction.
- For change-point detection in time series, we focused on sudden changes in the distribution, but there could be more subtle changes that happen, instead of instantaneously, *gradually* over a longer period of time. These gradual changes are harder to detect compared to sudden changes and may require us to observe a larger amount of data before making any decisions. Although some of the ideas, such as likelihood ratio statistics, may be applied in both cases, care must be taken when detecting gradual changes, such that they would not be confused with natural variations already present in the normal data.
- For outlier detection in event sequences using the continuous-time approach, the scoring methods are targeting one type of events with context provided by multiple types of events. In reality, we might also be interested in detecting outliers in multiple types

of events jointly. This can be achieved by combining the scores from all target types of events in a reasonable way, such as taking a summary statistic of all the scores, but developing a way that can be justified from a theoretical perspective would be important and challenging.

- We developed algorithms that can detect anomalies in continuous time, but our methods rely on a model learned from normal data. Although in practice, this is not necessarily hard to achieve, it would make the methods even more applicable if we can learn the model in presence of anomalies without any labeling. In the statistics community, theories related to robust estimation can help us understand the theoretical performance and limitations of any potential algorithms. In the machine learning community, there is research on robust learning and learning from noisy labels in general. Further research into this direction can help us understand the theoretical possibility and limitations of learning event time series models in presence of different types of anomalies and provide us with new models and algorithms that can adapt to these situations.
- We studied anomaly detection of an exiting type (change-points) and two new types (commission and omission outliers) in continuous time. These are the types of anomalies that we can see immediate practical usefulness, but they are by no means exhaustive. In general, the dependencies between the events can be quite complex across different types and different time. There are potentially new types of anomalies for event time series defined according to these structural dependencies instead of occurrences and absence of events. Defining new types of anomalies with practical usage and developing algorithms to detect them could generate many interesting problems and solutions.
- For the continuous-time approach, we focused on point process models in the temporal domain, but point processes can also be used in the spatial domain (or the combined spatial-temporal domain). They have applications in problems such as modeling earthquakes or crimes across different locations (over time). Extending our models and algorithms to the spatial domain could be an important addition to the existing research with valuable applications.

Bibliography

[1] Ryan Prescott Adams, Iain Murray, and David JC MacKay. Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 9–16. ACM, 2009.

[2] Charu C. Aggarwal. *Outlier Analysis*. Springer New York, 2013. ISBN 9781461463955.

[3] Emmanuel Bacry and Jean-Francois Muzy. Second order statistics characterization of Hawkes processes and non-parametric estimation. *arXiv:1401.0903 [physics, q-fin, stat]*, January 2014.

[4] M. S. Bartlett. The use of transformations. *Biometrics*, 3(1):39–52, 1947. ISSN 0006-341X.

[5] Iyad Batal, Dmitriy Fradkin, James Harrison, Fabian Moerchen, and Milos Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 280–288, 2012.

[6] Iyad Batal, Gregory F Cooper, Dmitriy Fradkin, James Harrison, Fabian Moerchen, and Milos Hauskrecht. An efficient pattern mining approach for event detection in multivariate temporal data. *Knowledge and Information Systems*, 46(1):115–150, 2016.

[7] Charles Blundell, Jeff Beck, and Katherine A Heller. Modelling reciprocating relationships with Hawkes processes. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2600–2608. Curran Associates, Inc., 2012.

[8] G. E. P. Box and George C. Tiao. A change in level of a non-stationary time series. *Biometrika*, 52(1/2):181–192, June 1965. ISSN 0006-3444.

[9] George EP Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[10] G. Casella and R.L. Berger. *Statistical Inference*. Duxbury Advanced Series in Statistics and Decision Sciences. Thomson Learning, 2002. ISBN 978-0-534-24312-8.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. ISSN 0360-0300.

[12] Ih Chang, George C. Tiao, and Chung Chen. Estimation of time series parameters in the presence of outliers. *Technometrics*, 30(2):193–204, 1988. ISSN 0040-1706.

[13] Chung Chen and Lon-Mu Liu. Joint estimation of model parameters and outlier effects in time series. *Journal of the American Statistical Association*, 88(421):284–297, 1993. ISSN 01621459.

[14] Jie Chen and Arjun K. Gupta. *Parametric Statistical Change Point Analysis*. Birkhäuser Boston, Boston, 2012. ISBN 978-0-8176-4800-8 978-0-8176-4801-5.

[15] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[16] William S. Cleveland and William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368): 829–836, December 1979. ISSN 01621459.

[17] William S. Cleveland and Susan J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, September 1988. ISSN 0162-1459.

[18] Daryl J. Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Springer, New York, 2003.

[19] Daryl J. Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Springer Science & Business Media, 2007.

[20] Hongyi Ding, Mohammad Emtiyaz Khan, Issei Sato, and Masashi Sugiyama. Bayesian nonparametric Poisson-process allocation for time-sequence modeling. *arXiv:1705.07006 [stat]*, May 2017.

[21] Hongyi Ding, Mohammad Khan, Issei Sato, and Masashi Sugiyama. Bayesian nonparametric Poisson-process allocation for time-sequence modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 1108–1116, 2018.

[22] Sophie Donnet, Vincent Rivoirard, and Judith Rousseau. Nonparametric Bayesian estimation of multivariate Hawkes processes. *arXiv:1802.05975 [math, stat]*, February 2018.

[23] Nan Du, Mehrdad Farajtabar, Amr Ahmed, Alexander J. Smola, and Le Song. Dirichlet-Hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, pages 219–228, Sydney, NSW, Australia, 2015. ACM Press. ISBN 978-1-4503-3664-2.

[24] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, 2016.

[25] Michael Eichler, Rainer Dahlhaus, and Johannes Dueck. Graphical modeling for multivariate Hawkes processes with nonparametric link functions. *Journal of Time Series Analysis*, 38(2):225–242, March 2017. ISSN 01439782.

[26] Peter J. Embi and Anthony C. Leonard. Evaluating alert fatigue over time to EHR-based clinical trial alerts: Findings from a randomized controlled study. *Journal of the American Medical Informatics Association*, 19(e1):e145–e148, 2012.

[27] Tom Fawcett and Foster Provost. Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316, 1997.

[28] Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 1, pages 53–62, 1999. ISBN 1-58113-143-7.

[29] A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3):350–363, January 1972. ISSN 0035-9246.

[30] Emily Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. Nonparametric Bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems*, pages 457–464, 2008.

[31] Piotr Fryzlewicz. Wild Binary Segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281, 2014.

[32] Pedro Garcia-Teodoro, J Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[34] Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems*, pages 1962–1970, 2011.

[35] Tom Gunter, Chris Lloyd, Michael A. Osborne, and Stephen J. Roberts. Efficient Bayesian nonparametric modelling of structured point processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 310–319, Arlington, Virginia, United States, 2014. AUAI Press. ISBN 978-0-9749039-1-0.

[36] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26 (9):2250–2267, 2013.

[37] P J Harrison and C F Stevens. Bayesian forecasting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 38(3):205–247, 1976. ISSN 00359246.

[38] Milos Hauskrecht, Michal Valko, Branislav Kveton, Shyam Visweswaram, and Gregory Cooper. Evidence-based anomaly detection. In *Annual American Medical Informatics Association Symposium*, pages 319–324, November 2007.

[39] Milos Hauskrecht, Michal Valko, Iyad Batal, Gilles Clermont, Shyam Visweswaram, and Gregory Cooper. Conditional outlier detection for clinical alerting. In *Proceedings of the American Medical Informatics Association*, November 2010.

[40] Milos Hauskrecht, Iyad Batal, Michal Valko, Shyam Visweswaran, Gregory F Cooper, and Gilles Clermont. Outlier detection for patient monitoring and alerting. *Journal of Biomedical Informatics*, 46(1):47–55, 2013.

[41] Milos Hauskrecht, Iyad Batal, Charmgil Hong, Quang Nguyen, Gregory F Cooper, Shyam Visweswaran, and Gilles Clermont. Outlier-based detection of unusual patient-management actions: an ICU study. *Journal of Biomedical Informatics*, 64:211–221, 2016.

[42] Alan G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971. ISSN 0006-3444.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[44] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodoligies. *Artificial Intelligence Review*, 22(1969):85–126, October 2004. ISSN 0269-2821, 1573-7462.

[45] Charmgil Hong and Milos Hauskrecht. MCODE: multivariate conditional outlier detection. *CoRR*, abs/1505.04097, 2015.

[46] Charmgil Hong and Milos Hauskrecht. Multivariate conditional anomaly detection and its clinical application. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 4239–4240, 2015.

[47] Charmgil Hong and Milos Hauskrecht. Multivariate conditional outlier detection and its clinical application. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 4216–4217, 2016.

[48] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3:160035, May 2016. ISSN 2052-4463.

[49] Re E Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960. ISSN 00219223.

[50] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, Proceedings, pages 389–400. Society for Industrial and Applied Mathematics, April 2009. ISBN 978-0-89871-682-5.

[51] Rebecca Killick and Ia Eckley. Changepoint: An R Package for changepoint analysis. *Lancaster University*, pages 1–15, 2013. ISSN 1548-7660.

[52] Rebecca Killick, Paul Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107 (500):1590–1598, 2012.

[53] Minyoung Kim. Markov modulated Gaussian Cox processes for semi-stationary intensity modeling of events data. In *International Conference on Machine Learning*, pages 2640–2648, July 2018.

[54] Genshiro Kitagawa. Non-gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82(400):1032–1041, 1987. ISSN 0162-1459.

[55] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1939–1947. ACM, 2015.

[56] Thomas A. Lasko. Efficient inference of Gaussian-process-modulated renewal processes with application to medical event data. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 469–476, Arlington, Virginia, United States, 2014. AUAI Press. ISBN 978-0-9749039-1-0.

[57] Eva K. Lee, Amanda F. Mejia, Tal Senior, and James Jose. Improving patient safety through medical alert management: An automated decision tool to reduce alert fatigue. *AMIA Annual Symposium Proceedings*, 2010:417–421, 2010. ISSN 1942-597X.

[58] Jeong Min Lee and Milos Hauskrecht. Recent-context-aware LSTM-based clinical time-series prediction. In *In Proceedings of AI in Medicine Europe (AIME)*, 2019.

[59] Jeong Min Lee and Milos Hauskrecht. Multi-scale temporal memory for clinical event time-series prediction. In *In Proceedings of the International Conference on AI in Medicine (AIME)*, 2020.

[60] Jeong Min Lee and Milos Hauskrecht. Clinical event time-series modeling with periodic events. In *The Thirty-Third International Flairs Conference*. AAAI, 2020.

[61] Young Lee, Kar Wai Lim, and Cheng Soon Ong. Hawkes processes with stochastic excitations. In *International Conference on Machine Learning*, pages 79–88, 2016.

[62] Wenzhao Lian, Ricardo Henao, Vinayak Rao, Joseph Lucas, and Lawrence Carin. A multitask point process predictive model. In *International Conference on Machine Learning*, pages 2030–2038, 2015.

[63] Chuanhai Liu and Donald B. Rubin. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5:19–39, 1995.

[64] Siqi Liu and Milos Hauskrecht. Nonparametric regressive point processes based on conditional Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1062–1072, 2019.

[65] Siqi Liu, Adam Wright, and Milos Hauskrecht. Change-point detection method for clinical decision support system rule monitoring. *Conference on Artificial Intelligence in Medicine*, 10259:126–135, June 2017.

[66] Siqi Liu, Adam Wright, and Milos Hauskrecht. Online conditional outlier detection in nonstationary time series. In *The Thirtieth International Flairs Conference*, pages 86–91, May 2017.

[67] Siqi Liu, Adam Wright, Dean F. Sittig, and Milos Hauskrecht. Change-point detection for monitoring clinical decision support systems with a multi-process dynamic linear model. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 569–572. IEEE, 2017.

[68] Siqi Liu, Adam Wright, and Milos Hauskrecht. Change-point detection method for clinical decision support system rule monitoring. *Artificial Intelligence in Medicine*, July 2018. ISSN 0933-3657.

[69] Zitao Liu and Milos Hauskrecht. A regularized linear dynamical system framework for multivariate time series analysis. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[70] Zitao Liu and Milos Hauskrecht. Clinical time series prediction: Towards a hierarchical dynamical system framework. *Artificial Intelligence in Medicine*, 65(1):5–18, 2015.

[71] Zitao Liu and Milos Hauskrecht. Learning adaptive forecasting models from irregularly sampled multivariate clinical data. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1273–1279, 2016.

[72] Zitao Liu and Milos Hauskrecht. Learning linear dynamical systems from multivariate time series: A matrix factorization based framework. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 810–818. SIAM, 2016.

[73] Zitao Liu and Milos Hauskrecht. A personalized predictive framework for multivariate clinical time series via adaptive model selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1169–1177, 2017.

[74] Zitao Liu, Lei Wu, and Milos Hauskrecht. Modeling clinical time series using gaussian process sequences. In *SIAM International Conference on Data Mining*, 2013.

[75] Chris Lloyd, Tom Gunter, Michael Osborne, and Stephen Roberts. Variational inference for Gaussian process modulated Poisson processes. In *International Conference on Machine Learning*, pages 1814–1822, 2015.

[76] Chris Lloyd, Tom Gunter, Michael Osborne, Stephen Roberts, and Tom Nickson. Latent point process allocation. In *Artificial Intelligence and Statistics*, pages 389–397, May 2016.

[77] D. Luo, H. Xu, H. Zha, J. Du, R. Xie, X. Yang, and W. Zhang. You are what you watch and when you watch: Inferring household structures from IPTV viewing data. *IEEE Transactions on Broadcasting*, 60(1):61–72, March 2014. ISSN 0018-9316.

[78] Markos Markou and Sameer Singh. Novelty detection: A review—part 1: Statistical approaches. *Signal Processing*, 83(12):2481–2497, December 2003. ISSN 0165-1684.

[79] Markos Markou and Sameer Singh. Novelty detection: A review—part 2:: Neural network based approaches. *Signal Processing*, 83(12):2499–2521, December 2003. ISSN 0165-1684.

[80] Hongyuan Mei and Jason M. Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, pages 6757–6767, 2017.

[81] Hongyuan Mei, Guanghui Qin, and Jason Eisner. Imputing missing events in continuous-time event streams. *arXiv:1905.05570 [cs, stat]*, May 2019.

[82] Y. Ogata. On Lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, 27(1):23–31, January 1981. ISSN 0018-9448.

[83] Sang Min Oh, James M. Rehg, Tucker Balch, and Frank Dellaert. Learning and inference in parametric switching linear dynamic systems. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1161–1168. IEEE, 2005.

[84] Sang Min Oh, James M. Rehg, Tucker Balch, and Frank Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, May 2008. ISSN 0920-5691, 1573-1405.

[85] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954. ISSN 0006-3444.

[86] Vladimir Pavlovic, James M. Rehg, and John MacCormick. Learning switching linear models of human motion. In *NIPS*, volume 2, page 4, 2000.

[87] A. N. Pettitt. A non-parametric approach to the change-point problem. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(2):126–135, 1979. ISSN 0035-9254.

[88] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6: 1939–1959, December 2005. ISSN 1532-4435.

[89] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[90] Syama Sundar Rangapuram, Matthias W. Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[91] Vinayak Rao and Yee W. Teh. Gaussian process modulated renewal processes. In *Advances in Neural Information Processing Systems*, pages 2474–2482, 2011.

[92] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. MIT press Cambridge, 2006.

[93] Yves-Laurent Kom Samo and Stephen Roberts. Scalable nonparametric Bayesian inference on point processes with Gaussian processes. In *PMLR*, pages 2227–2236, June 2015.

[94] Ashish Sen and Muni S. Srivastava. On tests for detecting change in mean. *The Annals of Statistics*, 3(1):98–108, 1975. ISSN 0090-5364.

[95] Christian R. Shelton, Zhen Qin, and Chandini Shetty. Hawkes process inference with missing data. In *Thirty-Second AAAI Conference on Artificial Intelligence*, April 2018.

[96] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer Science & Business Media, 2010.

[97] Tomáš Šingliar and Miloš Hauskrecht. Learning to detect incidents from noisily labeled data. *Machine Learning*, 79(3):335–354, June 2010. ISSN 0885-6125, 1573-0565.

[98] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Conditional anomaly detection. *Knowledge and Data Engineering, IEEE Transactions on*, 19(5): 631–645, 2007.

[99] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[100] William Trouleau, Jalal Etesami, Matthias Grossglauser, Negar Kiyavash, and Patrick Thiran. Learning Hawkes processes under synchronization noise. In *International Conference on Machine Learning*, pages 6325–6334, May 2019.

[101] Ruey S. Tsay. Time series model specification in the presence of outliers. *Journal of the American Statistical Association*, 81(393):132–141, 1986. ISSN 0162-1459.

[102] Ruey S. Tsay. Outliers, level shifts, and variance changes in time series. *Journal of Forecasting*, 7(May 1987):1–20, 1988. ISSN 02776693.

[103] Ryan Turner, Marc Deisenroth, and Carl Rasmussen. State-Space Inference and Learning with Gaussian Processes. In *PMLR*, pages 868–875, March 2010.

[104] Michal Valko and Milos Hauskrecht. Feature importance analysis for patient management decisions. *Studies in Health Technology and Informatics*, 160(Pt 2):861, 2010.

[105] Michal Valko, Branislav Kveton, Hamed Valizadegan, Gregory F. Cooper, and Milos Hauskrecht. Conditional anomaly detection with soft harmonic functions. In *Proceedings of the 2011 IEEE International Conference on Data Mining*, June 2011.

[106] Yichen Wang, Bo Xie, Nan Du, and Le Song. Isotonic Hawkes processes. In *International Conference on Machine Learning*, pages 2226–2234, 2016.

[107] Jeremy Weiss, Sriraam Natarajan, and David Page. Multiplicative forests for continuous-time processes. In *Advances in Neural Information Processing Systems*, pages 458–466, 2012.

[108] Jeremy C. Weiss and David Page. Forest-based point process for event prediction from electronic health records. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer, 2013.

[109] Mike West, P. Jeff Harrison, and Helio S. Migon. Dynamic generalized linear models and Bayesian forecasting. *Journal of the American Statistical Association*, 80(389): 73–83, March 1985. ISSN 0162-1459.

[110] Weng Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *International Conference on Machine Learning*, pages 808–815, August 2003.

[111] Adam Wright, Thu-Trang T. Hickman, Dustin McEvoy, Skye Aaron, Angela Ai, Jan Marie Andersen, Salman Hussain, Rachel Ramoni, Julie Fiskio, Dean F. Sittig, and David W. Bates. Analysis of clinical decision support system malfunctions: A case series and survey. *Journal of the American Medical Informatics Association: JAMIA*, March 2016. ISSN 1527-974X.

[112] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, pages 3247–3257, 2017.

[113] Hongteng Xu and Hongyuan Zha. THAP: A matlab toolkit for learning with Hawkes processes. *arXiv:1708.09252 [cs, stat]*, August 2017.

[114] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. Learning Granger causality for Hawkes processes. In *International Conference on Machine Learning*, pages 1717–1726, 2016.

[115] Hongteng Xu, Dixin Luo, and Hongyuan Zha. Learning Hawkes processes from short doubly-censored event sequences. In *International Conference on Machine Learning*, pages 3831–3840, July 2017.

[116] Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 676–681, New York, NY, USA, 2002. ACM. ISBN 978-1-58113-567-1.

[117] Rui Zhang, Christian Walder, Marian-Andrei Rizoiu, and Lexing Xie. Efficient non-parametric Bayesian Hawkes processes. *arXiv:1810.03730 [cs, stat]*, October 2018.

[118] Ke Zhou, Hongyuan Zha, and Le Song. Learning triggering kernels for multi-dimensional Hawkes processes. In *International Conference on Machine Learning*, pages 1301–1309, 2013.

[119] Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. In *Artificial Intelligence and Statistics*, pages 641–649, 2013.