UNIVERSITY OF CALIFORNIA
RIVERSIDE

Marked Temporal Point Processes for Irregular Time Series in the Context of Medical Data and Recommendation Systems

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Kazi Tasnif Islam

December 2020

Dissertation Committee:

Dr. Christian Shelton, Chairperson
Dr. Vagelis Hristidis
Dr. Vagelis Papalexakis
Dr. Vassilis Tsotras

The Dissertation of Kazi Tasnif Islam is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor Dr. Christian Shelton, for his continuous support of my Ph.D. study and related research, for his patience, motivation, and attention to detail. It would not have been possible to achieve the degree without his tireless support. I would also like to thank the rest of my committee members, Dr. Vagelis Hristidis, Dr. Vagelis Papalexakis, and Dr. Vassilis Tsotras, for their guidance throughout the long and arduous journey, and their insightful comments and encouragement. Thanks to my collaborators, Dr. Juan Casse, and Dr. Randall Wetzel, lab mates for the discussions, help, comments, insights, and the lovely time we spent together. Finally, I would like to thank my family and friends: my parents, my brother, my beautiful wife, and her family, for supporting me spiritually throughout writing this thesis and my life in general.

Thanks to the conference committee of the first conference on Machine Learning for Health Care (MLHC 17), and the editors of Journal of Machine Learning Research, Workshop and Conference Track, Volume 68, the workshop organizer committee of Workshop on Temporal Point Processes, NeurIPS 19, for generously accepting and publishing my research.

To my parents, my brother, and my lovely wife for all the support.

ABSTRACT OF THE DISSERTATION

Marked Temporal Point Processes for Irregular Time Series in the Context of Medical Data and Recommendation Systems

by

Kazi Tasnif Islam

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2020
Dr. Christian Shelton, Chairperson

Many real-world systems consist of large volumes of event data measured at irregularly spaced time intervals. The time interval between two events carries much information about the system dynamics, alongside the type of the event and other associated data. One such example of a system is the intensive care unit (ICU), where the captured patient records consist of sparse, noisy, incomplete, heterogeneous, and unevenly sampled patients' clinical data. Our first approach to model such data is a Bayesian structure learning based marked temporal point process model. We model the event streams, including vital signs and laboratory results in two different datasets using a piecewise-constant conditional intensity model (PCIM), a type of marked point process. Next, we model the stream of discrete clinical events in the continuous-time domain. We employ a neurally self-modulating marked temporal point process model that uses continuous-time long short-term memory (LSTM) cells as its building blocks. Our methods improve prediction performance in multiple tasks, including in-hospital mortality prediction, while providing suitable regularization and bypassing data imputation. We also experiment with a neural ordinary differential

equation-based framework that simultaneously models the temporal point process's intensity function and learn the underlying stochastic process that governs the distribution of the observed noisy measurements. This model achieves actual continuity as opposed to RNN based models, where the continuity is achieved through learnable decay rates of the hidden state, or through inputting the time between two observations as a scalar. Additionally, we apply a variational Bayes scheme to add uncertainty and regularization to the recurrent architectures on top of this framework. Another example of a real-world system containing a large amount of irregularly spaced time series-data is transaction systems, where the user's buying behavior can be considered a highly irregular temporal sequence. Predicting the next return time of a user based on the transaction history is useful for providing the recommendations at the right time. In this regard, we develop a hierarchical session-based recommendation system that combines a discrete recurrent intra-session architecture and a continuous LSTM based inter-session model. We train the model to simultaneously predict a new session's time using a temporal point process loss and recommend products within each session.

Contents

List of Figures **xi**

List of Tables **xii**

1 Introduction and Motivation **1**
1.1 Motivation . . . 1
1.2 Introduction . . . 2
1.3 Outline of the Thesis . . . 7

2 Related Work **12**
2.1 Traditional Learning . . . 13
2.1.1 Parametric models . . . 13
2.1.2 Non-Parametric models . . . 14
2.1.3 Structure Learning . . . 14
2.2 Neural Models . . . 16
2.2.1 Maximum Likelihood Estimation (MLE) . . . 16
2.2.2 Likelihood-free Learning . . . 24
2.3 Session time Prediction in Recommendation Systems . . . 27
2.4 Clinical Time Series Modeling . . . 29
2.4.1 Methods naturally capturing the irregularity . . . 29
2.4.2 RNNs adapted for irregular clinical time-series modeling . . . 31
2.4.3 Other Deep Learning Methods for EHR . . . 35

3 Marked Temporal Point Process with Structure Learning for Severity of Illness Assessment **36**
3.1 Summary . . . 36
3.2 Introduction . . . 37
3.3 Methods . . . 38
3.3.1 Piecewise-Constant Conditional Intensity Model . . . 39
3.4 Cohorts . . . 42
3.4.1 Cohort 1 . . . 43
3.4.2 Cohort 2 . . . 43

3.5 Experiments . . . 43
3.5.1 Data Pre-Processing and Experimental Setup . . . 44
3.5.2 Choice of Binary Tests . . . 45
3.5.3 Computing Severity of Illness Scores . . . 46
3.5.4 Predicting Mortality . . . 47
3.5.5 Baselines . . . 47
3.6 Results and Discussion . . . 47
3.7 Differences with existing Recurrent Neural Network based Methods . . . 50
3.8 conclusion . . . 50

4 Neural Hawkes Process for Classification of Irregularly Sampled Time Series 52
4.1 Summary . . . 52
4.2 Introduction . . . 53
4.3 Preliminaries . . . 54
4.4 Methods . . . 57
4.4.1 Neural Hawkes Process . . . 57
4.4.2 Mark Distribution . . . 60
4.4.3 Log-likelihood . . . 61
4.4.4 Jointly learning to classify . . . 61
4.5 MIMIC-III Benchmarks and Task Description . . . 63
4.5.1 In Hospital Mortality . . . 63
4.5.2 Phenotyping . . . 64
4.5.3 Decompensation . . . 64
4.5.4 Forecasting Length of Stay . . . 65
4.6 Experiments . . . 66
4.6.1 Experimental Setup . . . 66
4.6.2 Baselines and Our Model . . . 67
4.6.3 Results for In-Hospital Mortality Task . . . 70
4.6.4 Results for Length-of-stay Task . . . 72
4.6.5 Results for Phenotyping Task . . . 73
4.6.6 Results for Decompensation Task . . . 74
4.6.7 Joint Learning Prevents Overfitting . . . 75
4.6.8 Mixture Components lower Negative Log-likelihood Loss . . . 77
4.7 Summary . . . 78

5 Neural Stochastic Differential Equations with Bayesian Jumps for Marked Temporal Point Process 79
5.1 Summary . . . 79
5.2 Introduction and Related Work . . . 80
5.3 Preliminaries . . . 82
5.4 Proposed Framework . . . 85
5.4.1 ODE-GRU . . . 85
5.4.2 Maximizing the Log-likelihood . . . 87
5.4.3 Bayesian Jump . . . 87

5.4.4 Bayesian ODE-RNN 89
5.5 Experiments 90
5.6 Discussion and Future Work 93

6 A Joint Hierarchical Discrete RNN and Continuous Temporal Point Process Recurrent Model for Time and Item Predictions 94
6.1 Summary 94
6.2 Preliminaries and Problem formulation 95
6.3 The Joint Model 95
6.3.1 Context representation 96
6.3.2 Loss 97
6.3.3 Experiments and Results 98
6.4 Conclusion 100

7 Conclusions 101

Bibliography 104

List of Figures

3.1 Example decision trees representing a PCIM 41
3.2 Comparison of all methods for in hospital mortality prediction 48
3.3 Comparison of all methods for the scoring of labs and vitals only 48

4.1 Illustration of the neural Hawkes process architecture 56
4.2 Plots of the average performance and ranges of the models for the prediction tasks 71
4.3 Training and validation loss comparison as training progresses 76
4.4 Average validation loss decreases as the number of mixture components for the mark distribution is increased. 77

5.1 Illustration of Neural-ODE-RNN architecture with Bayesian jump 85

6.1 Time prediction results for the Last.FM dataset. Mean absolute errors are plotted versus the cumulative number of sessions with time gaps $\leq x$ days . 99

List of Tables

3.1 Binary Tests for learning PCIM. X denotes any measurement variable and τ and τ' are each one of a set of thresholds. Perhaps add more information on the thresholds? 46

4.1 Task Specific Sample Sizes for MIMIC-III benchmarking dataset 65
4.2 In-Hospital Mortality Prediction Task Results 72
4.3 Length-of-stay Task Results 73
4.4 Phenotyping Task Results 75
4.5 Decompensation Task Results 76

5.1 MIMIC-III Results 92

6.1 Statistics of the LastFM dataset after preprocessing 99

Chapter 1

Introduction and Motivation

1.1 Motivation

The real world has many examples of systems that evolve continuously in time and are interrupted by stochastic events, spaced by non-uniform time intervals. In many domains, such as social networks, global politics, computer systems, and the medical field, the underlying system's unknown dynamics can only be estimated by records of the events and their real-valued times. In addition to short term dependencies, these kinds of data can also have complicated long term dependencies on history. Data in the medical domain, such as electronic health records (EHRs), contain detailed descriptions of patients' symptoms, demographics, outcomes, and other encounter information. Electronic health records can additionally have real-valued noisy physiological measurements of different variables, including vital signs, drugs, and lab tests. These observations are inherently noisy due to charting errors, calibration, or device noise. Another example lies in real-world transaction data. User behavior in real-world transaction systems, for example, online marketplaces,

also produce asynchronous event sequences. Understanding the complex temporal dependencies in such sequences is essential for predicting future user activity, which is valuable for content recommendation and personalization. In this thesis, we mainly focus on two domains, medical applications and recommendation systems. However, the work is generalizable across all disciplines which produce asynchronous event sequences.

1.2 Introduction

The first application area is data generated in an Intensive Care Unit (ICU). In this thesis, we concentrate on records from stays in ICUs, because they provide an intense, well-monitored, and (relatively) short duration episodes. These attributes allow for the collection of a large number of patient trajectories with definite outcomes. We expect many of the modeling lessons learned in this setting to be transferable to other areas of patients' medical records.

We view the EHR data of a patient as a timeline of events happening throughout the patient's stay in the ICU, where an event for an individual patient is a new measurement of a particular physiological variable, a new lab test, a dosage of a particular drug, or a procedure performed, started or stopped. The measurements associated with such events are indicative of patient states and the subject of constant monitoring from clinicians. However, the temporal dynamics of the stream of events carry extra information for event forecasting and understanding the severity of illness. For example, abnormal values for blood pressure could indicate critical states of a patient, but a higher measurement *rate* could also indicate a high degree of urgency. Higher values of a particular physiological variable could instigate

the clinicians to order a particular lab test. High values of that lab result could result in the ordering of a related lab result, or initiate the dosage of a particular drug. These complex temporal dependencies that arise due to time, value, and types of different events throughout the timeline of a patient's stay in an ICU could be vital in understanding the temporal dynamics of the patient's state(e.g., the severity of illness). These sorts of data are not missing-at-random (or, more accurately, "measured-at-random").

One challenge in modeling medical data is the irregular arrival of different events. Not only are the temporal patterns of measurements indicative of the patients' conditions, but those patterns do not correspond to regular sampling intervals and are dependent on the previous history. Despite the irregular sampling property, traditionally, modeling is performed in the discrete-time domain by aggregating the data within fixed time intervals. However, just considering the sequence of events, without careful attention to modeling the time between events ignores much of the critical information in the datasets. These data often have long periods with no events, followed by bursts of events. Aligning the time to regular intervals, to allow for a discrete-time model, typically assume that the data is "missing at random," which is a simplified assumption for irregularly sampled data and assuredly inaccurate for medical data. Also, this artificially reduces the variability of signals because missing values are imputed with either zero, the population mean, or the value of the last recorded measurement of the signal. Therefore, modeling this irregular, noisy, and heterogeneous time series motivates models in the continuous-time domain, which does not require a fixed sampling rate.

In many real world systems including transaction systems, social networks, and many other systems, users act to produce many user activity data, which can be considered asynchronous event sequences. Each event has an irregularly sampled timestamp denoting the user activity time, accompanied by information related to the activity, such as the type of activity. Recommendation systems greatly benefit from predicting future user activity, including *when* they are going to return to the website along with *what* types of activities the user is going to perform next. The return time and the user's activity are jointly dependent on the history of the user's action, which can be represented as an asynchronous event sequence. Learning the temporal dynamics of such sequences would lead to predicting the time and type of the user's future action. Like the medical event sequences, this requires modeling the underlying process that generates the events in continuous time.

Marked temporal point process and intensity functions, with a well-established theoretical foundation, are a robust framework to model the event generation process. Next, we give a brief introduction of the temporal point process and their mathematical properties.

Temporal Point Processes Let samples from an unknown process controlling the dynamics of a system be available in the form of event streams containing a finite set of events of unbounded random size. Each event has a real-valued time and a mark consisting of a discrete-valued label specifying the type of the event. Additional marks can accompany each event, such as observed noisy measurements of a patient in an ICU. Formally, each sample $S = \{(t_1, k_1), (t_2, k_2), \ldots, (t_I, k_I)\}$ is an event sequence with non-uniform time intervals, where t_i is the time and k_i is the mark associated with event i, and $t_i < t_{i+1}$. For example, in a medical time series application, k_i can be composed of the pair

(e_i, v_i), where e_i corresponds to the event-type at time t_i, i.e.heart rate, blood pressure, etc., and v_i is the recorded noisy value of the measurement. The label, e_i, is drawn from a finite label set L, so $|L|$ represents the total number of event-types. We let I represent the total number of events, and T be the total length of the period ($T \geq t_I$).

Samples drawn from a temporal point process are a sequence of labeled events in continuous time, which enables continuous-time modeling of labeled time-series data with non-uniform time intervals without discretization and data imputation. Additionally, non-Markovian temporal point process models have a strong mathematical foundation. They can encode the entire previous history in the event sequence without specifying the order as required by Markovian models.

Conditional Intensity Function: A temporal point process can be represented as a counting process $N(t)$, which records the number of events before time t. The distribution over the next events conditioned on the history of events is specified using a **conditional intensity function**, $\lambda^*(t)$. Formally, for an infinitesimal time window , $[t, t + dt)$, the conditional intensity function, $\lambda^*(t)$ is the expected infinitesimal rate at which events are expected to occur in that window, conditioned on the history, $\mathcal{H}_t = \{(t_i, k_i)|t_i < t\}$, up to but not including the event at t:

$$\lambda^*(t) = \lim_{dt \to 0} \frac{\mathbb{E}\{N(t, t + dt)|\mathcal{H}_t\}}{dt} \tag{1.1}$$

where $\mathbb{E}\{N(t, t+dt)|\mathcal{H}_t\}$ is the expected number of events happening in the window $[t, t+dt)$ conditioned on the history $\mathcal{H}_t$. In case of a regular point process [108], two events coincide with likelihood 0.

The simplest case is a homogeneous Poisson process, which specifies the history independent intensity function as a constant: $\lambda^*(t) = \lambda$. In this case, the inter-arrival times are distributed independently and exponentially. If $P(t_i|\mathcal{H}_i)$ denotes the density of the next event occurring at time t_i given the history $\mathcal{H}_i$ prior to the event at t_i,

$$P(t_i|\mathcal{H}_i) = \lambda e^{-\lambda(t_i - t_{i-1})}. \tag{1.2}$$

More generally, $\lambda^*(t) = \lambda(t|\mathcal{H})$ is a function of the history which reduces the density of the next event occurring at time t_i to:

$$P(t_i|\mathcal{H}_i) = \lambda(t_i|\mathcal{H}_i) e^{-\int_{t_{i-1}}^{t_i} \lambda^*(s)ds}. \tag{1.3}$$

While $\lambda(t_i|\mathcal{H}_i)$ denotes the conditional intensity of an event at time t_i, the $e^{-\int_{t_{i-1}}^{t_i} \lambda^*(s)ds}$ term, known as the conditional survival function $S(t_i|\mathcal{H}_i)$, denotes the probability of no new event occurring up to time t_i since time t_{i-1}.

Popular Intensity Function Forms: Intensity functions are the most fundamental component in TPP modelling. Several forms of intensity functions have been proposed in the literature. Apart from the Poisson Process, other popular forms include,

- **Reinforced Point Process** [96]: Captures the rich get richer mechanism through an aging function. The form of the intensity function is $\lambda(t) = \lambda_0 f(t) i(t)$, where $f(t)$ captures the aging effect and $i(t)$ is the accumulation of history events

- **Hawkes Process** [53]: Describes an additive model to capture the self-excitation of previous events. The intensity function form is $\lambda(t) = \lambda_0 + \sum_{t_i < t} \kappa(t - t_i)$, where $\kappa(t - t_i)$ is the triggering kernel and captures the effect of history events on the intensity.

- **Reactive Point Process** [14]: Applies a self-inhibition term to the Hawkes process intensity function. The intensity function is denoted by, $\lambda_(t) = \lambda_0 + \sum_{t_i<t} \kappa(t-t_i) - \sum_{t_i<t} \gamma(t-t_i)$, where $\gamma(t-t_i)$ captures the inhibition of previous history

- **Self-correcting process** [58]: The background rate increases steadily, but decreases using a constant $e^{-\alpha} < 1$ when a new event is generated.

For a marked temporal point process observed within a time interval $[0, T)$, the number of observed events I is finite, and the continuous log-likelihood of such a finite event-stream is given by:

$$\log(P(S)) = \sum_i [\log \lambda^*(t_i) + \log P(k_i|\mathcal{H}_i, t_i)] - \int_{t=0}^{T} \lambda^*(t)\, dt. \tag{1.4}$$

The conditional mark distribution $P(k_i|\mathcal{H}_i, t_i)$ can be any distribution over the resulting mark, given the history and that the event occurred at time t_i. Popular choices for $P(e_i|\mathcal{H}_i, t_i)$ are Bernoulli distributions, while $P(v_i|\mathcal{H}_i, t_i)$ are treated as Gaussian.

1.3 Outline of the Thesis

Structure learning: Our first approach for modeling asynchronous medical time series data is based on a piecewise-constant conditional intensity model (PCIM) [50], a decision-tree-based marked temporal point processes model. The conditional intensity function for each event type in a PCIM is represented using a decision tree, learned through Bayesian structure learning. As medical time series data contain real-valued noisy measurements too, we augment the PCIM model to jointly learn the parameters of the value

distribution and the intensities of the different types of clinical events. We model the irregular event streams, including vital signs, laboratory results contained in two different datasets (MIMIC III - Medical Information Mart for Intensive Care clinical database) and data extracted from the EHRs of patients in a tertiary pediatric intensive care unit) and apply it for in-hospital mortality prediction, a highly critical clinical prediction task. Our experiments capture meaningful temporal dependencies and show improvement in in-hospital mortality prediction over traditional ICU scoring systems. One of the advantages of applying tree-based models in medical data is the added interpretability, which is a desirable aspect of machine learning models developed for deployment in the clinical setting. Our work is one of the first approaches in the literature applying marked temporal point process models in the context of clinical data and to our knowledge, the very first to apply point processes for a clinical classification task. In Chapter 3, we detail our proposed model, description of the cohort, experiments, and results.

Recurrent neural-networks-based temporal point process model for medical time-series classification: The overwhelming success of deep learning in multiple research areas combined with the fast development of deep learning theory and techniques, especially for recurrent neural network models have motivated researchers to apply deep models in temporal point process modeling. We apply some of the recently proposed RNN architectures for point processes [82, 31] in the context of medical time-series data. EHR data is used extensively for critical clinical problems, such as predicting patient acuity (the risk or required level of care) during hospital stays, phenotyping, or predicting the length of stays. While deep learning methods show superior performance in many tasks, including

image classification, machine translation, and speech recognition, they rely on large-scale datasets compared to traditional machine learning models. Medical data, however, is on a much smaller scale compared to the areas mentioned above. Therefore, careful regularization is vital for generalizability in clinical classification or regression tasks. We apply the RNN architectures to simultaneously learn the distribution of the temporal event sequences in the continuous-time domain and minimize the cross-entropy or the mean squared error loss for four clinical benchmark classification and regression tasks. The advantage is two-fold: First, we avoid discretization errors and the need for imputation in discrete-time RNN models for medical time-series classification. Second, jointly learning the temporal distribution of the sequence and performing classification tasks improves the testing performance of the later as solitary tasks. In Chapter 4, we provide details of the architectures, description of the clinical tasks performed, experiments, and results.

Ordinary differential equation based temporal point process model with a Bayesian filter: Many real-world systems evolve according to continuous dynamics and get interrupted by stochastic events, i.e., systems that both flow (often described by a differential equation) and jump. The stochastic event generation process can be modeled as samples generated from a marked temporal point process model, in the form of event sequences with non-uniform time intervals. Additionally, each event is marked with the type, and real-valued noisy measurements, such as blood pressure or heart-rate values, which are realizations of another stochastic process with an unknown stochastic differential equation (SDE). We present a framework that simultaneously models the intensity function of the temporal point process and learn the stochastic process governing the distribution of

the observed noisy measurements. This framework is based on neural ordinary differential equations [18], where the change in hidden state is expressed as a differential equation and specified by a neural network. We learn the distribution of the temporal point process by controlling the jump in the latent state at irregular observations by a regular RNN. Additionally, we learn the distribution of the SDE governing the values with a Bayesian filter, filtering the intrinsic noise in the observed measurements. The model achieves actual continuity as opposed to other RNN based models, where the continuity is achieved through learnable decay rates of the hidden state, or through inputting the time between two observations as a scalar.

In Bayesian modeling, there are two main types of uncertainties: aleatoric and epistemic. Aleatoric uncertainty is concerned with the noise in the observations, like the noisy measurements recorded in an ICU setting. On the other hand, epistemic uncertainty represents the uncertainty that the model causes itself. We address epistemic uncertainty by Bayesian neural networks, where probability distributions are placed as priors on the parameters of the network, instead of the outputs. The motivations for introducing uncertainty on the weights are two fold. First, it achieves regularization via a compression cost on the weights, which is useful mainly for sparse and irregular data. Second, this allows cheap model averaging for better representations and predictions.Model-based uncertainty is useful for many different reasons, particularly in the context of irregularly spaced data. As there can be extended periods of no observations, the proposed RNN architectures are, therefore, prone to overfitting. In addition to the property of the data, the limited scale of the data in the medical domain makes regularization a crucial aspect for better generaliz-

ability.

We outline the details of our framework and experimental evaluation in Chapter 5.

Chapter 2

Related Work

The work in the temporal point process (TPP) literature can be divided into two threads: traditional modeling and neural modeling using deep networks. Most of the earlier work was based on statistical TPP models, with a pre-specified form of intensity function. While they are interpretable and require a small amount of data, the fixed form of intensity functions restrict model capacity. Since the overwhelming success of deep learning [46], recent work in TPP modeling focuses more on neural network-based models, with recurrent neural networks (RNNs) leading the way. Although less interpretable, neural models are more flexible for a broader range of applications as they can make use of a large amount of data available for learning the parameters of the model. Deep models can be further divided according to two aspects of learning: likelihood free learning and maximum-likelihood-estimation (MLE) based models. We discuss the related work in the literature along these lines.

2.1 Traditional Learning

Traditional temporal point process modeling can be divided into two areas: parametric models that require a fixed form of intensity function and non-parametric models which do not require explicit parameterization of the intensity function. We discuss several works in these two domains in the following subsections. Although fairly recent, we have included the structure-learning based models in this section too.

2.1.1 Parametric models

Parametric models for temporal point processes are mostly based on optimization of the log-likelihood or its lower bound. Parametric Hawkes process models usually require specific forms of the background term and the triggering kernel. Most methods assume constant background intensity and take the triggering kernel as an exponential. Ozaki et al. [93] explicitly computed the Hessian and the gradients of the log-likelihood function, but this leads to slow convergence rates. Due to the branching nature of a Hawkes process, the background rates and the triggering effect can be de-clustered via expectation-maximization (EM) [85]. Veen et al. [124] proposed an EM framework, where, at each iteration, the parameters are decoupled and can be solved independently to optimize for a bounded objective. For multi-dimensional processes (which are equivalent to marked temporal point processes), sparse low-rank regularization is added to mitigate the curse of dimensionality [136]. Other techniques, such as sampling-based methods, are adopted in other forms of intensity functions, e.g., the reactive point process (RPP) [14].

2.1.2 Non-Parametric models

Non-parametric models don't require explicit parameterization of the intensity function and therefore improve the model capacity. The model-independent stochastic declustering (MISD) method [80], proposed in the context of seismicity, iteratively learns the triggering and background weights and the seismic intensity (background rates and excitation of previous earthquakes). Lewis et al. [72] further proposed the maximum penalized likelihood estimation (MPLE) method with additional regularization to improve the MISD method in terms of both accuracy and data size requirements. Zhou et al. [137] proposed a method for learning the triggering kernels for multi-dimensional Hawkes process through decoupling the parameters by constructing a tight upper-bound on the objective function. Then, the Euler-Lagrange equation is applied to derive an ordinary differential equation (ODE) for the optimal triggering kernels to satisfy.

2.1.3 Structure Learning

A conditional intensity model (CIM) is a set of non-negative conditional intensity functions indexed by the labels or the event-types, $\{\lambda_l(t|\mathcal{H})\}$. The piecewise constant conditional intensity Model (PCIM) [50] is a class of CIM where the conditional intensity function is a piecewise-constant function of time for any history. For each label l, a PCIM represents the conditional intensity function using a local structure T_l and local parameters θ_l. The structure specifies regions in the timeline where the conditional intensity function is constant and the parameters for each label θ_l represent the values of the intensity function in those regions. Formally, PCIMs are composed of local structures $T_l = (\Sigma_l, \sigma_l(t, \mathcal{H}))$ and

local parameters $\theta_l = \{\lambda_{ls}\}_{s \in \Sigma_l}$, where Σ_l denotes the set of states where the conditional intensity function is constant, λ_{ls} are non-negative constants representing the intensities in those states, and σ_l is a piece-wise constant state function in time that maps a time and a history to Σ_l. In a PCIM, the state function for each label, σ_l, is represented using a decision tree where the states $s \in \Sigma_l$ are the leaves and the internal nodes are binary test functions, formally defined as basis state functions [50]. They map a time t and a history $\mathcal{H}$ to a subtree. If the test functions are picked to be piecewise-constant functions of time for any event history, the intensity function $\lambda_l(t|\mathcal{H}) = \lambda_{ls}$, where $s = \sigma_l(t, \mathcal{H})$ becomes piecewise-constant as well.

The piecewise assumption allows the marginalization of the parameters. θ, to be computed in the closed-form given the structure T using a product of gamma distributions as a conjugate prior for θ. The decision trees are learned greedily by imposing a structural prior, which allows a closed-form Bayesian score to be computed and locally optimized.

An extension to the PCIM model, known as conjoint piecewise constant conditional intensity model (CPCIM) [94] specifies the state function as a function of the label, whose intensity is being evaluated, in addition to the time and the history. Thus, the state function $\sigma(l, t, \mathcal{H})$ can be represented using a single decision tree, mapping the time, label, and history to the states σ_s or the leaves of the decision tree, which results in a shared representation of the intensity parameters λ_s across all event-types. Thus, a CPCIM learns a more compact representation of the conditional intensity function and can require fewer parameters.

2.2 Neural Models

The temporal point process literature on neural models can be divided into two threads of research: models that maximize the likelihood in Equation 1.4 and more recent approaches based on reinforcement learning and generative adversarial networks (GANs), that do not require any form of specification of the intensity function.

2.2.1 Maximum Likelihood Estimation (MLE)

Maximize likelihood estimation based models maximize the likelihood in Equation 1.4 and are mostly based on recurrent neural network architectures.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks and its variants, e.g., long short term memory networks (LSTM) [56], process inputs sequentially and at each timestamp produce a hidden state, which encodes all the inputs seen so far by the network. Therefore, RNNs can be used to encode the influences of the previous history in an event sequence. The intensity function and the mark distribution can then be obtained from the encoded history through separate non-linear functions.

The first RNN approach for modelling temporal point processes was proposed by Du et al. [31]. Given a set of sequences $\{S^i\}$ for $S^i = (t^i_j, e^i_j)_{j=1}^{n_i}$, the maximum likelihood objective to optimize by the proposed network is

$$\max \sum_i \sum_j (\log P(e^i_{j+1}|h_j) + \log P(t^i_{j+1}|h_j), \tag{2.1}$$

where h_j is the hidden state of the RNN after processing input subsequences $\{(t_k, e_k)|t_k <$

$t_j\}$. The distribution for the next event $P(e^i_{j+1}|h_j)$ can be specified as a multinomial distribution and the conditional intensity function $\lambda(t^i_{j+1}|\mathcal{H}^i_j)$ is obtained from h_j through a non-linear function:

$$\lambda(t^i_{j+1}|\mathcal{H}^i_j) = \exp\Bigg(\underbrace{w^{t^\intercal} \cdot h_j}_{\text{past influence}} + \underbrace{v^t(t^i_{j+1} - t^i_{j-1})}_{\text{present influence}} + \underbrace{b^t}_{\text{base intensity}} \Bigg). \tag{2.2}$$

By contrast, the hidden state, $h(t)$, is a continuous function of time in the Neural Hawkes Process model [82]. One of the major differences between this model and that of Du et al. is that the later models the intensity of any event-type using one single non-linear function. However, the former has separate intensity functions for each event type and can be calculated from the hidden state using separate non-linear functions. The continuous dynamics of the model is achieved from the cell state vector, $\mathbf{c}(\mathbf{t}) \in \mathbb{R}^D$ of a modified long short term memory (LSTM) network. At each input-processing step, t_i, the LSTM produces a new steady-state vector, $\bar{\mathbf{c}}_{i+1}$, a new starting cell vector, $\mathbf{c}_{i+1}$, and decay rates δ_{i+1}. These learnable values are updated using regular LSTM update equations. Given these values, the cell state vector smoothly changes over time as

$$\mathbf{c}(\mathbf{t}) = \bar{\mathbf{c}}_{i+1} + (\mathbf{c}_{i+1} - \bar{\mathbf{c}}_{i+1}) \exp(-\delta_{i+1}(t - t_i)) \tag{2.3}$$

for $t \in (t_i, t_{i+1})$, that is until the next event at t_{i+1}. The hidden state vector $\mathbf{h}(\mathbf{t}) \in (-1, 1)^D$ can be continuously obtained from $\mathbf{c}(\mathbf{t})$ through the output gates of the network. Label independent intensity functions are learned from the continuous hidden state $h(t)$ using the scaled softplus non-linearity.

Xiao et al. [132] modeled the temporal point process using two separate RNNs, a time-series RNN and an event sequence RNN. The motivation of using two separate RNNs

is to decouple the learning of the background rates of events from dynamic or continuously changing features, such as age, heart rate, etc., and the effect of the previous history from asynchronous events happening over longer time intervals. The encoded history learned from the two synergic RNNs can be used to learn the intensities and the event probabilities. The event probabilities were learned using a cross-entropy loss, while the density for the next event time was learned using a Gaussian penalty function with fixed variance.

Attention-based RNN models have been proposed [125] to improve the interpretability of RNN models. The attention mechanism captures the impact (either negative or positive) of the previous events to the current one, which leads to uncovering the mutual excitation or inhibition effects among different event types in the data.

Neural Ordinary Differential Equations

The recently proposed Neural ODEs [18] are a family of continuous-time models where the hidden state $h(t)$ is defined as a solution to an ODE initial-value problem (IVP):

$$\dot{h(t)} = f_\theta(h(t), t), \tag{2.4}$$

where $h(t_0) = h_0$ and $f_\theta(h(t), t)$ is a multilayer perception (MLP) with parameters θ.

The continuous hidden state $h(t)$ can be evaluated at any set of times using a numerical ODE solver:

$$h_0, ..., h_I = ODESolve(f_\theta, h_0, (t_0, ..., t_I)). \tag{2.5}$$

The ODE-based flow helps achieving model-based continuity, which is particularly suited for applications where time is asynchronous. Chen et al. [18] discussed deriving Poisson process likelihoods from the continuous hidden state by parameterizing the intensity using an MLP. However, in many situations, the latent state can jump to new states after each observation, which is not handled with this model. Also, notably, the derived Poisson process likelihood from the ODE solutions at the observation times doesn't depend on history.

The ODE-RNN model proposed by [107] addresses the issue of jumps in the continuous latent state flowed according to an ODE, using recurrent units. The hidden state flows continuously according to an ODE in between the observations, and the solution to the ODE at time t is the input to a Gated Recurrent Unit(GRU) [25]. As a result, the hidden state is jumped and it starts flowing from the new state throughout the next interval. Formally,

$$h_i = ODESolve(f_\theta, h'_{i-1}, (t_{i-1}, t_i)), \tag{2.6}$$

where $h'_{i-1} = GRU(h_{i-1}, k_{i-1})$. That is, h_i is the hidden state before the observation at time t_i, and h'_i is the new latent state after the RNN sees the observation at time t_i. Similar history-independent Poisson process likelihoods are evaluated using the ODE-RNN, as in the previous work.

Jia et al. [61] addressed the change in the latent state and also modeled the effect

of history in the conditional intensity function of point processes. Their model addresses how to adapt the adjoint method [99] for jumps in the latent state, which results in memory-efficient learning.

Transformers

Recurrent Neural Networks are computationally expensive and fail to capture long term dependencies. For example, in medical data setting, diagnosis of critical diseases like diabetes might be dependent on long-term dependencies of the diagnosis history and medications. Recently proposed transformer architectures [123] are more suitable to capture long term dependencies and are much more inexpensive to compute.

Self-attention or intra-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence. In case of a point process, embedding the sequence history requires self-attention between a particular event (t_i, k_i) and all the prior events, $(t_j, k_j) \in \mathcal{H}_i$. Additionally, the self-attention mechanism requires a positional encoding, which captures the relative order between elements in a sequence.

Formally, in a self attention module, the attention output P is computed by

$$P = \text{Softmax}(\frac{QK^T}{\sqrt{M_K}})V, \tag{2.7}$$

where $Q = XW^Q, K = XW^K, V = XW^V$. Q, K, V are the query, key, and value matrices obtained from linear transformations of X using the weight matrices, where X is an embedding of a sequence. Zuo et al. [138] proposed the Transformer Hawkes Process, where, $X \in \mathbb{R}^{L \times M}$, represents the embedding of the irregular sequence S, where each row is an

M dimensional encoding of event (t_i, k_i). $W^Q, W^K \in \mathbb{R}^{M \times M_K}$ and $W^V \in \mathbb{R}^{M \times M_V}$, are the weight matrices for the linear transformations.

To encode the times $t_1, t_2, t_3, \ldots, t_L$ in a sequence, Zou et al.used the following approach:

$$[z(t_j)_m] = \begin{cases} cos(t_j/10000^{\frac{(i-1)}{M}}), \text{if } m \text{ is odd} \\ \\ sin(t_j/10000^{\frac{i}{M}}), \text{if } m \text{ is even} \end{cases} \tag{2.8}$$

where $z(t_j) \in R^M$, $m \in M$ is a dimension for the M dimensional encoding.

Zhang et al. [134], on the other hand, used a different time-encoding approach, which is given by

$$z(t_j)_m = \sin(\omega_m \times j + w_m \times t_j), \tag{2.9}$$

where t_j is the time of the j-th event in a sequence, and ω_m and w_m are the specified angle frequencies for dimension $m \in M$. Multiple sinusoidal functions with different w_m and ω_m angular frequencies are used for encoding and are concatenated at the end. Even and odd dimensions are generated from the sine and cosine functions respectively.

The embedding of the sequence, S, is specified by $X = (UY + Z)^T$, where $Y = [k_1, k_2, k_3, \ldots, k_L] \in \mathbb{R}^{K \times L}$ is a collection of K dimensional one-hot encoding of the events. $Z = [z(t_1), z(t_2), \ldots, z(t_L)] \in \mathbb{R}^{M \times L}$ is the concatenation of time encodings and $U \in \mathbb{R}^{M \times K}$ is an embedding matrix for the events, where each column represent the M dimensional embedding for a particular event type. Each row in $X \in \mathbb{R}^{L \times M}$, corresponds to the embedding of a particular event in the sequence.

For the Transformer Hawkes Process described by Zhou et al., X is then passed

through the self attention module in Equation 2.7. Usually, multi-head attention is used for model flexibility, so different attention outputs P_1, P_2, P_H are obtained from different sets of $\{W_h^Q, W_h^K, W_h^V\}_{h=1}^H$ weight matrices.

The aggregated attention output is $P = [P_1, P_2, \ldots, P_H]W^O$, where W^O is an aggregation matrix. The j-th column in P expresses the dependence of the j-th event in the sequence on its history. To get the history embedding, the attention output is passed through further linear and non-linear transformations, $H = ReLU(PW_1^{FC} + b_1)W_2^{FC} + b_2$, using weight matrices W_1^{FC} and W_2^{FC}. The resulting matrix H is an $L \times M$ dimensional matrix containing hidden representations of all the events in the input sequence, and $h(t_j) = H(j, :)$, the particular representation for event at t_j.

The conditional intensity can then be obtained using

$$\lambda_k(t|\mathcal{H}_t) = f_k(\alpha_k((t - t_j)/t_j) + w_k^T h(t) + b_k), \tag{2.10}$$

where b_k is a parameter specifying base intensity, $\alpha_k((t - t_j)/t_j)$, captures the influence of the current observed time and the last observed time, which is weighted by a coefficient α_k, and then $w_k^T h(t)$ captures the influence of history, which is linearly transformed by the weight matrix w_k. The linear transformations then get passed through a non-linear softplus function, $f_k(.)$ to obtain the desired intensity, where $f_k(x) = \beta_k \log(1 + \exp(x/\beta_k))$ is the softplus function with "softness" parameter β_k.

After applying event embedding and adding it to the temporal encoding, the sequence embedding X is obtained similar to Transformer Hawkes Process, where x_i denotes the embedding for event i. Following the sequence

$\{(t_1, k_1), (t_2, k_2), (t_3, k_3), \ldots, (t_L, k_L)\}$, given a particular time t, to get the desired intensity for event type k, an embedding, x_{i+1} is first obtained for (t, u) using the embedding approach described above. Then, to capture the similarity with previous events, the following self-attention mechanism is applied:

$$h_{i+1}^k = (\sum_{j=1}^{i} f(x_{i+1}, x_j) g(x_j)) / \sum_{j=1}^{1} f(x_{i+1}, x_j), \tag{2.11}$$

where, $g(.)$ is a linear transformation using weight matrices and $f(.,.)$ is specified as an embedded Gaussian, with $f(x_i, x_j) = \exp(x_i x_j^T)$. This resembles Equation 2.7. However, instead of transforming the x_i-s using additional linear transformations, similarity is directly captured using the $f(.,.)$ and $g(.)$ functions.

The conditional intensity for event type k is calculated from the following equation:

$$\lambda_k(t|\mathcal{H}_t) = \text{softplus}(\mu_{k,i+1}) + (\eta_{k,i+1} - \mu_{k,i+1}) \exp(-\gamma_{k,i+1}(t - t_i)), \tag{2.12}$$

where $\mu_{k,i+1}$, $\eta_{k,i+1}, \gamma_{k,i+1}$ denote base intensity, peak intensity, and the decay rate respectively. They are obtained from the history embedding, h_{i+1}^k, through linear transformations using weight matrices W_μ, W_η, and W_γ, and then applying GeLU and softmax non-linearity functions.

Variational Auto Encoder

Mehrasa et al. [81] proposed a variational auto-encoder formulation for modelling temporal point processes. The input sequences of event types and the asynchronous inter-arrival times are encoded using a recurrent VAE model. At each time step, t_i, the model uses history, $\mathcal{H}_i$, to produce a latent distribution, z_i. The latent distribution in then used

to decode two probability distributions: one for the type of the next event and the other for the inter-arrival time for the next event. The latent distribution, z_i, is encoded using a time-dependent inference network, $q_\phi(z_i|\mathcal{H}_i)$, which is approximated using a conditional Gaussian with parameters, $\mu_{\phi(i)}$, and $\sigma_{\phi(i)}$. $q_\phi(z_i|\mathcal{H}_i)$ is matched closer to the prior, $p(z_i)$, using a KL divergence loss. Instead of using a fixed prior and ignoring the temporal dependencies between events, a time-varied prior is incorporated, which also takes into account the history, $\mathcal{H}_i$. At step t_i, the prior and posterior networks observe history, $\mathcal{H}_i$. However, the prior produces parameters for the conditional Gaussian distribution for next event at time, t_{i+1}. On the other hand, the posterior networks outputs the parameters for the current event in consideration, observed at time, t_i. Given the latent state, z_i, the model generates two conditionally independent distributions for the event type and for the inter-arrival time for the next event. The distribution over event types is modeled using a multinomial distribution, whereas the inter-arrival temporal distribution is modeled using an exponential distribution parameterized by $\lambda(z_n)$. The prior and posterior networks rely on an LSTM network to encode the history and then an additional network is used to calculate the respective priors and posteriors. Both, the temporal distribution and the event-type distributions are decoded using a multi-layer perceptron from hidden state, z_i.

2.2.2 Likelihood-free Learning

Maximum likelihood estimation is asymptotically equivalent to minimizing the Kullback-Leibler (KL) divergence between two distributions. Conventional point process models maximizing the likelihood are prone to noise and outliers, especially given multi-modal distributions. Some of the recent approaches in temporal point process learning

make use of reinforcement learning and generative adversarial networks [47], to devise an intensity free learning approach.

Reinforcement Learning

Li et al. [73] framed the problem of learning temporal point processes in the deep reinforcement learning (RL) setting, which results in a maximum-likelihood-estimation-free learning paradigm which doesn't suffer from model misspecification. The generation process of new events is treated as taking new actions in the environment by the agent in an RL setting. Formally, π_e is an **expert policy**, which generates the observed event-sequences. The goal is to learn a **learner policy** $\pi(a|\mathcal{H}_t)$, which specifies the probability of the next event given the history $\mathcal{H}_t := \{t_i\}_{t_i<t}$. $\pi(a|\mathcal{H}_t)$ is specified by a recurrent neural network, where the generated events are fed back into the RNN with stochastic neurons [86]. Additionally, at each time t_i, the event generation is associated with the value of a reward function $r(t)$ at time t_i. However, the reward function is unknown and can be learned via *inverse reinforcement learning* [87], which iterates between learning the reward function and the stochastic policy. The discrepancy between the expert and learner policy is minimized to avoid the problem of slow convergence in inverse reinforcement learning. The authors show that, if the function class of the reward is the unit ball in reproducing kernel Hilbert space (RKHS)[2], the optimal reward function can be specified in a non-parametric closed form. The optimal stochastic policy can then be learned using a customized policy gradient method with the optimal reward function.

Upadhyay et al. [121] specified both the actions taken by an RL agent and the feedback received from the environment as asynchronous stochastic discrete events param-

eterized by two separate temporal point processes. Unlike Li et al., the reward function is known and can be arbitrarily complex. Formally, the action events, $\mathcal{A} = \{(t_i, a_i)\}$, are drawn from the distribution, $p^*_{\mathcal{A},\theta}$, specified by a temporal point process with conditional intensity function λ^*_θ and the conditional mark distribution m^*_θ. Similarly, the feedback events, $\mathcal{F} = \{(t_i, k_i)\}$, are realizations of the temporal point process, $p^*_{\mathcal{F},\phi} = (\lambda^*_\phi, m^*_\phi)$, with conditional intensity λ^*_ϕ and conditional event distribution, m^*_ϕ. The joint densities of the action and the feedback sequence depend on the joint history of events $\mathcal{H}_t = \{\mathcal{A}_t \cup \mathcal{F}_t\}$. Furthermore, the agent receives a stochastic reward $R^*(T)$ after period T, which depends on it's stochastic actions and the feedbacks from the environment. The action distribution i.e. the policy, $p^*_{\mathcal{A},\theta} = (\lambda^*_\theta, m^*_\theta)$, is parameterized using a recurrent neural net architecture, similar to the Du model [31]. This reduces the objective function to

$$J(\theta) = \mathbb{E}_{\mathcal{A}_T \sim p^*_{\mathcal{A},\theta}(\cdot), \mathcal{F}_T \sim p^*_{\mathcal{F},\phi}(.)}[R^*(T)]. \tag{2.13}$$

The optimal policy $p^*(\mathcal{A}, \theta)$ is derived by maximizing the expected reward function $J(\theta)$ using stochastic gradient descent (SGD). It is shown that the REINFORCE trick [128] can be adopted for deriving the gradients. However, additional differentiable regularizers are added to the reward function to impose a budget on the number of action events and the marks.

Generative Adversarial Networks (GANs)

Generative adversarial networks (GAN) [47], have a strong theoretical foundation and empirical verification. Notably, recent improvements in the Wasserstein GAN (WGAN) [3], which replaces the Jensen-Shannon (JS) distance adopted in the original GAN by the

earth moving (i.e., Wasserstein) distance, makes the model more sensitive to the underlying geometry structure of samples and robust to issues like mode-dropping.

The Wasserstein GAN Temporal Point Process (WGANTPP) model proposed by Xiao et al. [130] aims to learn a parameterized generator g_θ whose generated distribution is close to the distribution generating temporal sequences. The distance is minimized using the Wasserstein distance, and the unconditional generative model g_θ takes as input ξ_l, a random sequence sampled from a Poisson process. Unconditional generative models learn the overall distribution of the training set, which cannot be directly used for sequence modeling. Xiao et al. [131] proposed a conditional generative model, which is more suitable for individual sequence level prediction. The observed history of each sequence is encoded by a sequence-to-sequence recurrent network (seq2seq LSTM). A gradient-based Lipsitz regularizer is added to the Wasserstein loss, which minimizes the distance between two distributions, while the likelihood loss is also used in the final loss function to make the best of two worlds.

2.3 Session time Prediction in Recommendation Systems

While predicting the right product to the right user is an essential task in recommendation systems, knowing when is the appropriate time for the return of a user and curating the recommendations based on that is also extremely valuable. The return times of users in online platforms can be considered as an irregular event sequence, where each event can represent the beginning or end of a particular session and the mark k_i can represent an L-dimensional multi-hot encoding vector representing all the clicks or purchases within

that session.

Formally, session data for user u can be represented by, $S_u = \{(b_1, e_1, a_1),$ $(b_2, e_2, a_2), \ldots, (b_{n_u}, e_{n_u}, a_{n_u})\}$, where b_i represents the begin time of session i, e_i represents the end time of session i, a_i is the K dimensional sparse action vector, and n_u is the total number of sessions that user u participated in. The goal is to predict gap time $b_i - e_{i-1}$, from the history, H_i.

Using the temporal point process terminology, the log-likelihood for the session data of all the users can be written as,

$$L_p(S, T) = \sum_u \sum_{i=1}^{n_u} ((\int_{e_{i-1}}^{b_i} \lambda_u(t)dt - \log \lambda_u(b_i)) + \log P(a_i))) + \int_{t_{n_u}}^{T} \lambda_u(t)dt, \tag{2.14}$$

where T is the terminal time up to which user data is available, $\lambda_u(t)$ is the intensity function for user u for generating a new event, either a session-start or a session-end event, and $P(a_i)$ is the probability of action a_i for user u.

Jing et al. [62] represents the continuous function $\int_{e_{i-1}}^{b_i} \lambda_u(t)dt$ as an approximation by a piece-wise constant function: $\sum_{t=e_{i-1}}^{t=b_i; t+=\delta(t)} \lambda_u(t)\delta(t)$.

Embedded user representation u, the gap time embedding $d_{i-1} = (b_{i-1} - e_{i-2})$, user action embedding, a_{i-1} are concatenated and passed as an input to an LSTM, which embeds the history h_i. The desired intensity is then obtained from h_i, by passing through a linear transformation and then applying a softrelu non-linearity.

Vassøy et al. [122] frames the problem using a different terminology. Instead of representing all the items in a session as a multi-hot encoded vector a_i, each item is represented separately. However, a discrete-time intra-session RNN captures the dependencies

within a session and that gets passed to an outer RNN, which is responsible for predicting the session time. This leads to a joint hierarchical model, which tries to maximize the log-likelihood for the next session time using a regular point process loss and simultaneously improve the recommendation score of the items within each session using the discrete time intra RNN. The two losses are optimized jointly using a weighted loss function.

2.4 Clinical Time Series Modeling

Approaches to clinical time series modeling can be categorized into two ways of handling the temporal irregularity of measurements: i. methods that naturally capture the irregularity, and ii. methods that adapt recurrent or convolutional neural networks to the irregular time setting.

2.4.1 Methods naturally capturing the irregularity

Weiss et al. [127] extended the piecewise-constant conditional intensity models (PCIMs) to a multiplicative forest-based temporal point process model (MFPP), building on ideas from multiplicative-forest continuous-time Bayesian networks (mfCTBNs) [126]. MFPPs extends the tree structure of PCIMs to regression forests, using a multiplicative-forest technique developed in CTBNs. The multiplicative property allows a concise representation of composite rates of events without sacrificing the flexibility to model rates with complicated dependencies. They apply their model for predicting the onset of myocardial infarction from irregular and heterogeneous clinical time-series data. They conduct experiments on two forecasting setups: first, where no labels are provided in the forecasting

region, and second, where the outcomes in the forecasting region are used in training to predict whether at least one event of myocardial infraction would occur in the forecasting region for a patient in the test set.

Fauber et al. [37] used piecewise constant conditional intensity models (PCIMs) [50] to model the data in continuous time for the reconstruction of monitor readings of patient vitals from sparse EHR data. For the probabilistic inference of continuous monitor values, a new Reversible Jump MCMC inference method [48] was developed around the augmented PCIM, where the distribution of the values learned on the leaves of the tree.

Ranganath et al. [102] proposed a hierarchical survival model in the context of the EHR, where all the observations, including patient covariates, are modeled jointly conditioned on a rich latent structure, based on deep exponential families (DEF) [103]. DEFs can capture a hierarchy of dependencies between latent variables and can be generalized to many settings through exponential families. Instead of an arbitrary time zero as in traditional survival analysis, the authors propose to align the observations by their failure time instead. Thus, risk can be evaluated at any particular time, not only in timepoints that align with the particular synchronization event like the entry to a clinical trial. The authors applied their method in patient stratification on the risk of developing coronary heart disease (CHD) on 313,000 patients corresponding to 5.5 million months of observations.

MedGP [19] is a Bayesian nonparametric model based on Gaussian process (GP) regression for hospital patient monitoring. The model is based on a highly structured Gaussian process kernel that supports tractable computation over tens of thousands of time points. Regularization was performed using sparsity inducing priors, which also allows in-

terpretability and computational tractability. Additionally, the model can be tailored to specific patients using nonparametric density estimation techniques. They evaluate the method over 6,000 patients from three disease groups with more than four million measurements.

Futoma et al. [42] employed a multi-output Gaussian process RNN for early sepsis detection. Latent values from a Gaussian process at fixed grid points are fed into an RNN model for classification. While this interpolates the irregularly spaced values, it does not model the timing of the events directly. Other methods that naturally capture the irregularity include variants of Gaussian Processes [74, 112].

Razavian et al. [105] proposed a multitask model based on the temporal convolutional network to predict disease onsets from sparse lab measurements. One of the contributions of the approach was a differentiable multivariate kernel estimation for imputing missing values, which can be learned jointly with the parameters of the CNN. The method was applied for the onset of multiple disease prediction from the observed window of 36 months on a dataset containing lab measurement and diagnosis information for 298,000 individuals.

2.4.2 RNNs adapted for irregular clinical time-series modeling

Most of the work in temporal clinical time series modeling falls in the second class of methods involving RNNs and CNNs. The first example of the use of RNNs for EHR data [75] used long short-term memory recurrent neural networks (LSTM-RNNs) for disease phenotyping. They employed a discrete (hourly) time-window approach, and imputed values in the missing windows using either a forward or back-filling strategy. For

added regularization, they replicate the targets at each window, instead of having the target only at the last time-step and then back-propagating the error. In a subsequent work [76], they experiment with different imputation strategies, notably imputing missing values with 0s, using binary indicator variables, or using both. Experiments suggest a combination of both strategies perform the best for an LSTM.

Doctor AI [21], is a temporal predictive model for diagnosis prediction using recurrent neural networks (RNN). The model was applied to longitudinal time-stamped EHR data from a large cohort of 260K patients and 2,128 physicians over eight years. The inputs to the model are a multi-hot encoding of the detailed patient encounter records, including diagnosis, medication codes, or procedure codes. Additionally, the duration between the previous and the current visit is added as an input to the model. The RNN learns to predict the time of the next visit and the diagnosis and medication categories associated with it.

Aczon et al. [1] developed a dynamic mortality risk prediction model using LSTM-RNNs for pediatric ICU data. At any time t, where at least one variable is observed, the RNN takes as input the vectored observations (missing values are imputed with 0s) and a controllable Δt term. The RNN then learns to predict the risk score of the patient at $t + \Delta t$. The Δt term and no discretization allows a dynamic model, which can adapt to new observations, and the risk score can be computed at any time.

Che et al. [16] introduced trainable decays for both the input and the hidden states of Gated Recurrent Units (GRUs) to capture the temporal structure of the missing values. If an input variable is missing, instead of directly replacing it with it's last measured value x_l, or the empirical mean x_m, the value of the missing variable becomes, $\delta_v x_l + x_m - \delta_v x_m$, where

δ_v, is a learnable decay parameter for variable v. Also, a decay parameter is introduced to the GRU update equations, by decaying the previous hidden state h_{t-1} as $h_{t-1} = \gamma_{h_t} \circ h_{t-1}$, before computing the new hidden state h_t.

Habiba et al. [51] extends on the works of Che et al. by incorporating a continuous neural-ODE network. The proposed models use the continuous differential equation solvers in order to compute the hidden dynamics of the model. The imputed data is calculated using differential equation solvers, where the missing observations of variables are replaced by the derivative of the value of the available observations of the corresponding variables. Additionally, the decay rates are computed as derivatives of time t. Unlike Che at al. the time series are treated as a continuous function of t, not of a discrete sequence.

Ma et al. [78] experimented with attention-based recurrent models for diagnosis prediction. They experimented with three different attention strategies for computing the attention weights: computing the weights directly from the previous hidden states h_i, concatenating the current hidden state h_t with h_i, and with adding a weight matrix as $h_i W_\alpha h_t$. Song et al.[116] used the seminal transformer model [123] in the context of clinical time series data for four clinical benchmark prediction tasks.

Tan et al. [119] proposed an end-to-end network named DATA-GRU, which consists of a time-aware structure to handle the irregular time intervals. The value for a particular missing variable at timestamp, $t*$, is imputed using a conditional Gaussian distribution. However, the imputed values are not always reliable compared with the actual records. Additionally, different imputed data can have different degrees of reliability. This motivates to set an "unreliability score" for each value, where the score, $U[x*]$ for value $x*$

is 0 if it's an actual record, or it's set as $Con[x*] > 0$, if it's an imputed value, where $Con[x*]$ denotes the co-variance functions for the Gaussian distributions imputing the values. The GRU then takes inputs the times of the measurements, the values of the measurements, and additionally the unreliability scores to compute the hidden state. Additionally, time intervals are incorporated into the network to adjust the hidden states. The following decay function is used in the paper and then multiplied with the hidden state to get the decayed hidden state

$$w_{\Delta t} = 1/\log(e + \Delta t) \tag{2.15}$$

$$h^d_{t-1} = h_{t-1} \odot w_{\Delta t} \tag{2.16}$$

, where h^d_{t-1} denotes the decayed hidden state and $w_{\Delta t}$ is the corresponding weight for interval Δt. To weight the reliabilities of different types of data, i.e, imputed data and real data, an unreliability-aware attention mechanism is proposed to adjust the weights assigned to different data and ensure high-quality data play more important roles in the prediction performance. Additionally, to incorporate the medical knowledge into missingness, a symptom aware attention mechanism is proposed.

Harutyunyan et al. [52] curated a benchmarking dataset of four clinically relevant tasks and proposed comparisons between LSTMs and other linear and non-linear classifiers such as logistic regression. In a subsequence work, they proposed two other methods using LSTMs, one using a bidirectional LSTM preprocessor and the other using replicated targets at each timestep. For the four benchmark tasks, these two methods have the current state-of-the-art performance.

2.4.3 Other Deep Learning Methods for EHR

Researchers have used deep learning methods in various other tasks such as concept embedding, augmenting the health records data, and EHR data privacy. While phenotyping is a particular case of concept embedding, as various data elements are mapped to a particular phenotype, other concept embedding models like med2vec [22] also provide vector representations of the phenotypes. Concept embedding is often trained in an unsupervised learning setup, leveraging massive datasets to extract patient representations or embeddings [22, 84]. Autoencoders and their variants (sparse and denoising) are the preferred deep learning methods used in concept embedding. Generative adversarial networks (GANs) [47] have been used in the medical domain for data augmentation and synthesis by generating continuous medical time series data [35] and discrete codes [23, 15].

EHR data privacy is concerned about creating de-identified patient records. Recently, LSTMs [30] and hybrid LSTM models [77] have been used in this domain with success. As this a large area, a full review is out of the scope. We refer to [129] for a good review of different directions of research for applying deep learning methods for EHR data.

Chapter 3

Marked Temporal Point Process with Structure Learning for Severity of Illness Assessment

3.1 Summary

Electronic Health Records (EHRs) consist of sparse, noisy, incomplete, heterogeneous, and unevenly sampled clinical data of patients. They include physiological signals, lab test results, procedural events, clinical notes. Such data can be treated as a temporal stream of events of varied types occurring at irregularly spaced time points. Our first approach to model such data is a Bayesian structure learning based marked temporal point process model named PCIM [50]. We model the event streams, including vital signs, laboratory results contained in two different datasets (MIMIC III — 'Medical Information Mart

for Intensive Care' clinical database — and data extracted from EHRs of patients in a tertiary pediatric intensive care unit) using a piecewise-constant conditional intensity model (PCIM), a type of marked point process. Our experiments capture meaningful temporal dependencies and show improvement in in-hospital mortality prediction over traditional ICU scoring systems.

3.2 Introduction

Our general goal is to explore how modeling the temporal dynamics of the raw EHR data stream in the continuous-time domain could facilitate specific critical decision-making tasks performed in an ICU. To achieve that goal, we have performed mortality prediction in the ICU using the first 24 hours of ICU data from two different datasets, including the publicly available MIMIC-III database [65].

The severity of illness (SOI) assessment and mortality modeling is a broad area of research in health informatics. SOI and mortality scores are used to predict patient outcomes and often used as the principal measures of quality-of-care comparison among ICUs and stratification for clinical trials. Scoring systems like SAPS-I (Simplified Acute Physiology Score) [69], SAPS-II [68], PIM2 (Pediatric Index of Mortality) [115], and PRISM3 [98] are the accepted current practices in ICU acuity scoring, but are based on static snapshots of certain clinical variables over a patient's stay in the ICU and ignore the rapidly evolving temporal dynamics of the clinical variables. Several works in the literature have focused on boosting the predictive power of such scoring systems using valuable clinical information present in an EHR such as clinical notes [43, 70].

While the temporal dynamics of the topics derived from the notes using traditional topic modeling techniques are useful [63, 44] for patient risk assessment, the temporal information present in patient trajectories of vital signs or lab measurements has not been extensively studied. We conclude that temporal modeling of clinical variables, including vital signs and lab tests, can complement the standard scoring systems and improve mortality prediction performance.

In this chapter we discuss the following contributions.

- We model the temporal dependency in streams of measurement data using piecewise-constant conditional intensity models (PCIMs).
- For modeling the measurement values, we extend the PCIM model to allow marks that contain continuous-values, in addition to the discrete-valued labels.
- From the learned models, we assign risk scores to each patient trajectory with vital signs and lab tests and use these as inputs to a feedforward neural network [57] to predict mortality.

3.3 Methods

We chose PCIM, which is a class of marked point process, due to its flexible representation of the rate function as a decision tree. This promotes an interpretable and concise representation of the temporal dependencies.

3.3.1 Piecewise-Constant Conditional Intensity Model

Assume events are drawn from a finite label set L, representing the different event types. An event can then be represented by a pair: a time stamp t and a label $l \in L$. An event sequence x is $\{(t_i, l_i)\}_{i=1}^{n}$, where $0 < t_1 < \ldots < t_n$. We use $h_i = \{(t_j, l_j) | (t_j, l_j) \in x, t_j < t_i)\}$ for the history of event i. Let $t(y)$ for an event sequence y be the time of the last event in y, such that $t(h_i) = t_{i-1}$.

A conditional intensity function $\lambda(t|x)$ associated with a temporal point process is the expected instantaneous rate at which events are expected to occur at time t given the history before t. A conditional intensity model (CIM) is a set of such non-negative conditional intensity functions indexed by the labels $\{\lambda_l(t|x, \theta)\}_{l \in L}$. The likelihood of event sequence x can then be written as

$$p(x|\theta) = \prod_{l \in L} \prod_{i=1}^{n} \lambda_l(t_i|h_i; \theta)^{\mathbf{1}_l(l_i)} e^{-\Lambda_l(t_i|h_i;\theta)}, \tag{3.1}$$

where $\Lambda_l(t|h; \theta) = \int_{t(h)}^{t} \lambda_l(\tau|h; \theta) d\tau$. If $l' = l$, the indicator function $\mathbf{1}_l(l')$ is one, and it is zero otherwise. $\lambda_l(t|h; \theta)$ is the expected rate of event l at time t given h and model parameters θ. As it is conditional on the entire history, the process is non-Markovian.

A PCIM is a class of CIM where the conditional intensity function is a piecewise-constant function of time for any history. For each label l, a local structure S_l specifies regions in the timeline, where the conditional intensity function is constant and local parameters for each label θ_l represent the values of the intensity function in those regions. Formally, PCIMs are composed of local structures $S_l = (\Sigma_l, \sigma_l(t, x))$ and local parameters $\theta_l = \{\lambda_{ls}\}_{s \in \Sigma_l}$, where Σ_l denotes the set of states where the conditional intensity function is constant, λ_{ls} are non-negative constants representing the intensities in those states, and

σ_l is a piecewise constant state function in time that maps a time and a history to Σ_l. In a PCIM, the state function for each label, σ_l, is represented using a decision tree where the states $s \in \Sigma_l$ are the leaves and the internal nodes are binary test functions, formally defined as basis state functions [50]. They map a time t and a history h to a subtree. If the test functions are picked to be piecewise-constant functions of time for any event history, the intensity function $\lambda_l(t|h) = \lambda_{ls}$, where $s = \sigma_l(t, h)$ becomes piecewise-constant as well. The resulting likelihood of the event sequence x can then be written as

$$p(x|S, \theta) = \prod_{l \in L} \prod_{s \in \Sigma_l} \lambda_{ls}^{c_{ls}(x)} e^{-\lambda_{ls} d_{ls}(x)} \ , \tag{3.2}$$

where $S = \{S_l\}_{l \in L}$, $\theta = \{\theta_l\}_{l \in L}$. c and d are the sufficient statistics for likelihood calculation. $c_{ls}(x)$ is the total number of events of label l occurring in x that map to state s, and $d_{ls}(x)$ is the total duration when the trajectory for l is mapped to s. An example of a PCIM model is given in Figure 3.1.

The basis state functions or the test functions need to be carefully chosen to control the capacity of the resulting model. One of the approaches is to index the functions based on predefined time windows and thresholds. Some examples are

- Is the time of day between 6am and 9am?
- Is the number of events with the label B in the past half an hour greater than a threshold?
- Was the most recent event of label A?

The piecewise-constant assumption allow for efficient inference and learning of the model. Gunawardana et al. showed that the marginal likelihood of an event sequence, x, can be computed in closed form given the structure S using a product of gamma distributions

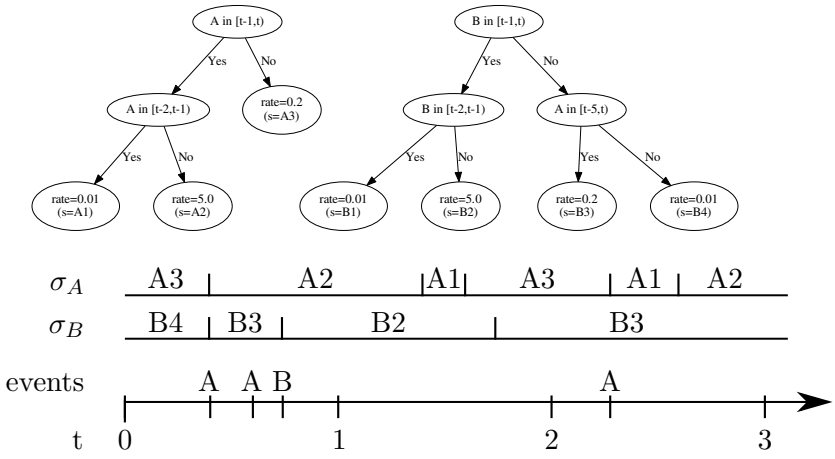


Figure 3.1: Top: Example decision trees representing a PCIM for two labels A and B. Bottom: Example trajectory and the corresponding piecewise-constant assignment of time to states (leaves of the PCIMs).

as a conjugate prior for θ.

For learning the decision trees greedily, imposing a structural prior allows a closed form Bayesian score to be computed. Given a structure, the rates associated with the states (leaves) can be selected using maximum a posteriori or maximum likelihood estimation.

In the previous sections, we have represented an event sequence as a sequence of pairs, where each pair consists of the time stamp and label representing the time and type of a particular event. For EHR data, we take the label to be the type of measurement (pulse, for example). However, these events also have associated values (the heart rate measurement, for example). In this thesis, we have focused only on events with numerical values. However, we expect our modeling lessons to be transferable to other forms of event data including clinical notes which can be represented in numerical format, such as topic proportions.

Incorporating the values, an event is now a triple: a time stamp t, a label $l \in L$ and the value of the event $v \in \mathbb{R}$. The notation for an event sequence x then becomes $\{(t_i, l_i, v_i)\}_{i=1}^{n}$, where $0 < t_1 < \ldots < t_n$. We have extended the original PCIM model to model the values in each state $s \in \Sigma_l$. We impose a Gaussian distribution $\frac{1}{\sigma_{ls}\sqrt{2\pi}} e^{-\frac{1}{2\sigma_{ls}^2}(v-\mu_{ls})^2}$ on the value v associated with an event from a particular state s, represented by a leaf in the decision tree.

The parameter set θ_l is now $\{\lambda_{ls}, \mu_{ls}, \sigma_{ls}\}_{s \in \Sigma_l}$, and, after incorporating the product of Gaussians with the product of the exponential distributions in Equation 3, the likelihood is

$$p(x|S_l, \theta_l) = \prod_{s \in \Sigma_l} \Lambda_{ls} \lambda_{ls}^{c_{ls}(x)} e^{-\lambda_{ls} d_{ls}(x)} \ , \tag{3.3}$$

where

$$\Lambda_{ls} = \left(\frac{1}{\sigma_{ls}\sqrt{2\pi}}\right)^{c_{ls}(x)} \ e^{-\frac{1}{2\sigma_{ls}^2}(u_{ls}(x) - 2\mu_{ls} m_{ls}(x) + \mu_{ls}^2 c_{ls}(x))} \ . \tag{3.4}$$

Here, $u_{ls}(x) = \sum_{v \leftarrow s} v^2$, $m_{ls}(x) = \sum_{v \leftarrow s} v$ are the sufficient statistics where $v \leftarrow s$ indicates that v is the value of an event of the portion of x that has been mapped to state s. We add a normal-gamma distribution as the (independent) prior over each μ_{ls}-σ_{ls} pair of parameters to allow for closed-form Bayesian scores for structure learning via the greedy tree-growing algorithm.

3.4 Cohorts

Mortality modeling is often performed among heterogeneous population of patients to compare quality of care between different ICUs. To demonstrate the generality of our proposed modeling approach, we focus on two different cohorts, a cohort of pediatric patients

in a tertiary PICU and a cohort of adult patients.

3.4.1 Cohort 1

We used data from the EHR archive of PICU at Children's Hospital Los Angeles (CHLA), which consists of 11684 patient episodes, collected over a period of 10 years. It includes demographics, outcomes, PIM2 and PRISM3 scores, and times and value for measurements of vital signs, interventions, drugs, and lab tests.

3.4.2 Cohort 2

The MIMIC-III database is the largest publicly available clinical dataset. It integrates de-identified clinical data of patients admitted to the Beth Israel Deaconess Medical Center in Boston between 2001 and 2012 [65]. The dataset includes demographics, vital signs, laboratory results, medications, charted observations, clinical notes, and more. Our second cohort consists of adult patients in MIMIC-III, aged 18 years or older.

3.5 Experiments

To show the effectiveness of our proposed temporal dependency modeling approach, we conducted experiments to predict in-hospital mortality based on the first 24 hours of clinical data for both the cohorts. The pre-processing step involves aggregating multiple related variables and normalization. Next, using a hold-out validation set, we select the final set of variables to be included in the PCIM models, based on the univariate performance of each variable. The final set of variables correspond to the set of event types

or the label set L. Critical components of the models such as the basis state functions were also chosen using the validation set. Next, we learned two separate sets of PCIM models on the training set for the two classes of patients: patients who died in hospital and those who survived. Finally, we obtain severity of illness scores from the two sets of models using the log-odds ratio and use the scores as mortality predictors. Details for each of the steps in our experimental process are explained in the following subsections.

3.5.1 Data Pre-Processing and Experimental Setup

For both datasets, some of the most commonly used clinical variables used in ICU scoring systems (PRISM3, SAPS-I, SAPS-II) were used to compose the primary sets of variables to perform our experiments. For the MIMIC-III dataset, we aggregated multiple related variables into one using a publicly available codebase[1]. The same codebase was also used to extract patient mortality outcomes, SAPS-I, SAPS-II score, and measurements for the first 24 hours. The CHLA data were similarly mapped.

For the cohort of pediatric patients, we normalized the values of age-dependent variables, such as heart rate, dividing by the median value among healthy children of the same age range and sex, according to published tables [39]. Then, we performed z-score normalization for all variables and removed data falling outside of ± 4 standard deviations from the mean of each variable. For adult patients there is relatively low variability on variables across different age groups and therefore we performed only z-score normalization.

The two class labels are "death" and "survival." We take the former to be the positive class. For both datasets, we randomly perform a (65-15-20) training-validation-test

[1]https://github.com/MIT-LCP/mimic-code

split. The validation set was used for hyper-parameter tuning and selection of the set of binary tests (both type and parameters). In order to make comparisons with PIM2 and PRISM3, we ran our final experiments for the CHLA dataset on patient episodes with both PIM2 and PRISM3 scores reported in the dataset (PRISM3 scores of 0 were dropped). This resulted in a final dataset containing 4601 patient episodes (mortality rate 6.2%). Constructing only the test set with patient episodes having both scores reported would have introduced bias. For MIMIC-III, the final dataset included 34971 patient episodes corresponding to the first ICU and hospital stay of the patient (in hospital mortality rate 11.67%).

3.5.2 Choice of Binary Tests

The base binary tests available for internal nodes for the PCIM are listed in Table 3.1. The tests for checking whether the total number of measurements within some time interval is greater than a particular threshold (τ) is to allow the model to capture the burstiness in measurements of certain vital signs and lab tests. These tests allow modeling of self-excitation (burstiness) and self-suppression. We also introduce a test to check whether the number of measurements with value $\geq \tau'$ is greater than a particular threshold (τ) within a particular time interval. This allows the rate of subsequent events to depend on the values of previous events (e.g. measurements with extreme values of a particular variable causing more events of the same or a different variable in near future).

The full bank of binary tests are chosen by selecting the parameters from predetermined sets of values. In particular, "past t hours" uses $t \in \{1/60, 1/12, 1/4, 1/2, 1, 24\}$; "value $\geq \tau'$" uses $\tau' \in \{-3.0, -2.5, -2, -1.5, 1.5, 2, 2.5, 3.0\}\times$std. dev+

number of measurements of variable X in past t hours is $\geq \tau$
number of measurements of variable X in past t hours with value $\geq \tau'$ is $\geq \tau$
current time t is within a particular interval within the day

Table 3.1: Binary Tests for learning PCIM. X denotes any measurement variable and τ and τ' are each one of a set of thresholds. Perhaps add more information on the thresholds?

mean; "number of measurements $\geq \tau$" uses $\tau \in \{0, 1, 5, 10, 20, 30, 50\}$; and "current time is within a particular interval of the day" uses range of one of "within δt minutes of the top of the hour," where $\delta t \in \{1, 2, 5, 15\}$ or within the first t hours of the day where $t \in \{1, 2, 3, \ldots, 24\}$.

3.5.3 Computing Severity of Illness Scores

Denote the set of models learned for the "died" class as M_{death} and the set for the "survived" class as M_{survive}. Both sets include PCIM models learned on the measurement timings and values of the selected variables. While each PCIM tree models the rate for one particular variable, the binary tests in the tree can look at all available variables (both vitals and labs) to capture the temporal dependencies with other variables. If the corresponding PCIM trees for a particular variable v in the two sets M_{death} and M_{survive} are denoted by m_{vd} and m_{vs} respectively then, the SOI score corresponding to variable v for patient with event sequence x is calculated by the log-odds: $\log \frac{p(m_{vd}|x)}{p(m_{vs}|x)}$. These individual SOI scores for each variable can be summed up to form a single SOI score or can be used as inputs along with other features to a final classifier model.

3.5.4 Predicting Mortality

We use a feedforward neural network as the final classifier using the computed severity of illness scores (as above) and other features. For MIMIC-III, the static features include age, gender, minimum Glasgow Coma Scale, minimum ratio of partial pressure arterial oxygen and fraction of inspired oxygen, total urine output, co-morbidity, and reception of ventilation, all collected in the first 24 hours. For the CHLA dataset, we used PIM2 score as a feature.

For both datasets, we use a feedforward architecture with 3 hidden layers with 256 hidden units each. Number of hidden layers and number of hidden units were selected based on performance on validation data. Dropout [117] rate of 0.5 was used for each hidden layer. All the models were trained with the Adam optimization method [67] using early stopping. We used Keras Deep Learning Library [24] for the implementation.

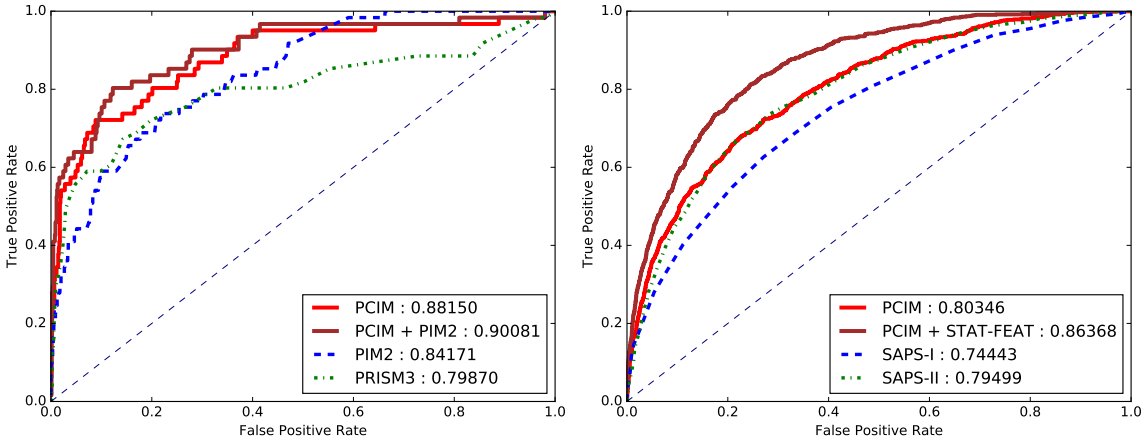
3.5.5 Baselines

For evaluating the predictive performance of our approach, we compare with PIM2 and PRISM3 scores for Cohort 1, and SAPS-I and SAPS-II scores for Cohort 2.

3.6 Results and Discussion

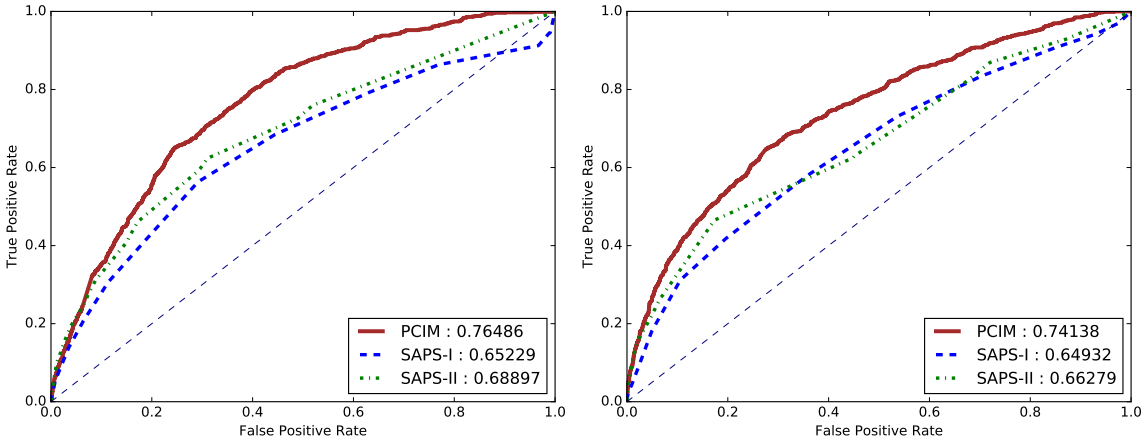
In this section, we present a comparative evaluation of our method with the standard ICU scoring systems. Statistical significance of the difference between the ROC curves presented was measured using MedCalc statistical software[2] [29].

[2]https://www.medcalc.org



(a) Cohort 1 (CHLA) (b) Cohort 2 (MIMIC-III)

Figure 3.2: Comparison of all methods for in hospital mortality prediction



(a) Scores from lab variables only (MIMIC-III) (b) Scores from vital signs only (MIMIC-III)

Figure 3.3: Comparison of all methods for the scoring of labs and vitals variables in MIMIC-III

Figure 3.2a shows the comparison of ROC curves for PIM2, PRISM3, and SOI scores computed from PCIM models. SOI scores computed from the learned PCIM models outperform standard scoring system baselines, PIM2 and PRISM3 ($p(\text{PCIM} \sim \text{PRISM3}) < 0.024$ and $p(\text{PCIM} \sim \text{PIM2}) = 0.146$). The reason for relatively higher p-values is highly likely to be caused by the small test set size of the Cohort 1. PCIM models do not directly incorporate information such as admission baseline features which capture how ill a child was at the time of admission. PIM2 is largely based on such features calculated within two hours of admission and outperforms PRISM3 in our experiments on the holdout testset. Best performance is achieved when PIM2 is combined with the SOI score obtained from the PCIM models by adding PIM2 as an extra feature to the feedforward net.

ROC curves for SAPS-I, SAPS-II, and SOI scores computed from PCIM models on the MIMIC-III dataset are presented in Figure 3.2b. Similar to the CHLA dataset, pre-admission and in-ICU information, such as ventilation and comorbidity were not directly incorporated in the PCIM models. When added to our method (again, as features to the feedforward net), we outperform both SAPS-I and SAPS-II with statistical significance (AUROC difference has $p < 0.0001$).

For MIMIC-III, Figure 3.3a shows the ROC curves using only labs or only vital signs. Scores computed from the four vital variables significantly outperform (AUROC, $p < 0.0001$) the isolated vitals scores of both SAPS-I and SAPS-II, computed using certain ranges of values, Figure 3.3b. The area under the ROC curve difference computed only using the lab variables is similarly statistically significant ($p(\text{PCIM} \sim \text{SAPS-I}) < 0.0001$ and $p(\text{PCIM} \sim \text{SAPS-II}) = 0.0001$), Figure 3.3a.

3.7 Differences with existing Recurrent Neural Network based Methods

While existing RNN based approaches [75, 1, 52, 76, 16] can strongly capture the temporal trend of the values of the variables, our proposed methodology is also taking into account the intensity of different events by directly modeling the generation of a new event of a particular type with respect to previous events' values, timings, and types. One particular advantage to directly model the event generation process in continuous time domain is to avoid discrete window based aggregation and any form of data imputation. Another advantage of our method is the model interpretability, added flexibility of model selection through the choice of basis state functions.

3.8 conclusion

Our contribution has been to demonstrate the efficacy of directly modeling the continuous time temporal dependencies of discrete events recorded in an EHR. Our results indicate that a severity of illness score computed using our proposed modeling technique improves the performance of hospital mortality prediction.

One of the limitations of our approach is the use of a generative model for the task of classification, using the log-odds calculated from the representative models for the two class labels, learned separately. However, careful choice of the basis state functions from a validation set resulted in significant discrimination between learned models of the two classes and improved prediction performance.

Our method provided one prominent feature for the final classifier. We added other static measurements (age, minimum Glasgow Coma Scale, minimum ratio of partial pressure arterial oxygen and fraction of inspired oxygen, total urine output, Pa02, and co-morbidity) as features to the classifier. These could have also been added as possible values for binary tests in the PCIM model. This, particularly in conjunction with disciminative PCIM training, might work better as a classifier. We also focused only on nine variables for MIMICIII because those are used in SAPS-II. The model might also be improved with an more expansive variable set. For the Cohort 1 data, we used more variables and saw a larger improvement. In one case, this was not statistically significant; we expect this is because of the relatively smaller sample size.

Chapter 4

Neural Hawkes Process for Classification of Irregularly Sampled Time Series

4.1 Summary

We model the stream of discrete clinical events in an EHR in continuous time using a **neurally self-modulating marked temporal point process model** which uses **continuous-time long short-term memory (LSTM)** cells as its building blocks. Our experiments show that joint learning of the distribution of the temporal event sequence in continuous time setting, along with minimizing the cross-entropy for four clinical benchmark classification tasks improves testing performance of the later as solitary tasks.

4.2 Introduction

One challenge in modeling such a heterogeneous event stream is the irregular time spacing of different events. Despite the irregular nature of the data, traditionally modeling is performed in the discrete-time domain by aggregating the data within fixed time intervals. Typically, these approaches assume that the data is "missing at random," which is a simplified assumption for irregularly sampled data and assuredly inaccurate for medical data. Also, this artificially reduces the variability of signals because missing values are imputed with either zero, the population mean, or the value of the last recorded measurement of the signal.

Therefore, we directly model the underlying generation process of clinical events using methods that naturally capture the temporal irregularity. Motivated by the recent success in deep learning in the health-care domain, we base our model architecture on a neural continuous-time marked point process model [82] in which the distribution of the event sequence evolves according to a continuous-time LSTM network, a variant of the highly popular discrete-time LSTM [56].

A general assumption in the temporal point process framework is that only one type of event can occur at a particular instant, which does not hold in many real-world event sequences. For example, multiple vital signs can be recorded at once in a patient's ICU record. Therefore, we design our neural network architecture to capture the dynamics of event sequences where multiple event-types can co-occur at any particular time instant.

Most approaches in the literature for clinical classification tasks involve discretizing patient sequences into hourly time intervals, feeding the data into a discrete LSTM

architecture, and training the model using the cross-entropy loss for the prediction problem at hand. Because our method provides a full joint distribution over the event sequence (not just a final output prediction), we can train on both the sequence likelihood and the output prediction. We show that the former improves the latter, as it provides suitable regularization.

Overall, by modeling the data using a generative continuous-time point process, we can 1) avoid making any missing-at-random assumptions, 2) model the entire sequence without data aggregation, and 3) train on the full likelihood. Taken together, these provide new state-of-the-art results on standard ICU benchmark tasks. We demonstrate our approach's performance on in-hospital mortality prediction, length-of-stay prediction, patient "phenotyping," and decompensation prediction.

4.3 Preliminaries

Formally, our goal is to model a stream of events $S = \{(t_1, k_1), (t_2, k_2), \ldots, (t_I, k_I)\}$, where t_i is the time and k_i is the mark (the type of event and the corresponding recorded value) associated with event i. We let I represent the total number of events and T be the total length of time ($T \geq t_I$). For all cases, $t_i < t_{i+1}$. Because EHR data frequently records multiple measurements at the same time (for instance, vital signs like heart rate, blood pressure, and respiratory rate), our marks need to account for multiple values. To do this, the mark k_i consists of the pair (e_i, v_i). The first component is a multi-hot encoding of the set of all L measurement types (heart rate, glucose, blood pH, etc.): $e_i \in \{0, 1\}^{|L|}$. The second, v_i, is a vector of values with one element for each of the L measurements: $v_i \in \mathbb{R}^{|L|}$.

Event types that co-occur have non-zero values in the multi-hot encoding representation and the corresponding elements in v_i are the associated measurements. Elements of the vector v_i corresponding to the event-types that did not occur at time t_i are represented with 0s.

For a marked point process observed within a time interval $[0, T)$, the number of observed events I is finite, and the continuous log-likelihood of such a finite event-stream is given by

$$\sum_{i=1}^{I} (\log p(t_i|\mathcal{H}_i) + \log p(k_i|\mathcal{H}_i, t_i)) + \log p(t_{I+1} > T|\mathcal{H}) \tag{4.1}$$

where $\mathcal{H}_i$ denotes the history prior to event i, $\{(t_1, k_1), (t_2, k_2), \ldots, (t_{i-1}, k_{i-1})\}$, and $p(t_i|\mathcal{H}_i)$ is the density of the next event occurring at time t_i given the history through event $i-1$.

A continuous-time marked point process quantifies the distribution over the next event using an **intensity function**, $\lambda(t)$. The simplest case of a homogeneous Poisson process specifies the intensity function as a constant: $\lambda(t) = \lambda$. In this case, the inter-arrival times are distributed exponentially: $p(t_i|\mathcal{H}_i) = \lambda e^{-\lambda(t_i - t_{i-1})}$.

More generally, we let $\lambda(t)$ be a function of the history and

$$p(t_i|\mathcal{H}_i) = \lambda(t_i, \mathcal{H}_i) e^{-\int_{t_{i-1}}^{t_i} \lambda(s, \mathcal{H}_i) ds} \ . \tag{4.2}$$

The conditional mark distribution $p(k_i|\mathcal{H}_i, t_i)$ can be any distribution over the resulting mark, given the history and that the event occurred at time t_i. We use a continuous-time LSTM model to capture $\lambda(t)$ and describe the mark distribution $p(k_i|\mathcal{H}_i, t_i)$ by a neural network with inputs from the LSTM's state and outputs parameterizing a mixture of Bernoulli (for e_i) and Gaussian (for v_i) distributions.

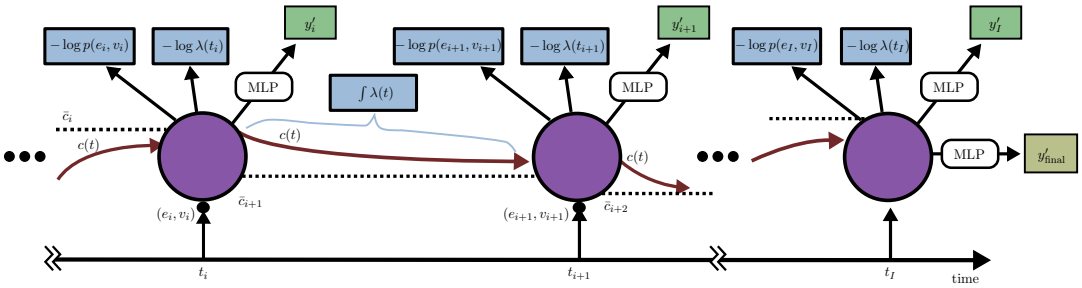


Figure 4.1: Illustration of the neural Hawkes process architecture using our mixture components. The model takes e_i, v_i at time t_i to update the initial cell state c_{i+1} (not shown in diagram) and the new target cell state $\bar{c}_{i+1}$. The continuous hidden state $h(t)$ is dictated by the continuous cell state $c(t)$ within the interval (t_i, t_{i+1}). $h(t_{i+1})$ dictates the intensity at time t_{i+1} and the mixture parameters $(\{\pi_m, P_m^{(e)}, p_m^{(v)}\}_m)$ for the distribution of $k_{i+1} = (e_{i+1}, v_{i+1})$. Components in blue represent terms used in the (negative log) likelihood loss (Equation 4.11). Components in green represent loss terms used for intermediate predictions and the component in yellow represents the loss term for predictions based on the entire sequence (Equations 4.12 and 4.13).

4.4 Methods

4.4.1 Neural Hawkes Process

The **Hawkes process**[53] is a well-known example of a marked point process in which past events in the history $\mathcal{H}$ have additive influence to the rate of future events. This influence decays over time according to a kernel function. However, real world influences can be both excitatory and inhibitory. The neural Hawkes process [82] removes this restriction by using a recurrent neural network (RNN) to learn a complex dependency of the intensity functions on the order, number, and timing of past events. The fundamental building block of the architecture is a novel continuous-time long short-term memory cell. Figure 4.1 schematically shows some components of the process, along with our additions for the distribution of multiple simultaneous marks and per-event and per-sequence predictions.

The continuous-time LSTM cells control the dynamics of the distribution of the temporal event stream through the cell state vector $c(t) \in \mathbb{R}^D$. This cell state changes continuously and smoothly over time, with jumps at event times. At such event times, a new steady state value for the cell state vector, $\bar{c}$, is outputted by the RNN as well as the rate at which $c(t)$ will approach this steady state value, δ. After event i, the RNN produces a new steady-state vector, $\bar{c}_{i+1}$, and a new starting cell vector, c_{i+1}, for the next time interval. We give the details of how these are calculated below. Given these values, the cell state smoothly changes over time as

$$c(t) = \bar{c}_{i+1} + (c_{i+1} - \bar{c}_{i+1}) \exp(-\delta_{i+1}(t - t_i)) \tag{4.3}$$

for $t \in (t_i, t_{i+1})$, that is, until the next event at t_{i+1}. This is shown in Figure 4.1 by the red

arrows, decaying from c_{i+1} to $\bar{c}_{i+1}$.

The hidden state vector $h(t) \in (-1, 1)^D$, also a continuous function of time t, is continuously obtained from $c(t)$. At each time $t > 0$ the hidden state $h(t)$ can be computed from $c(t)$ using the output gates o_{i+1} (also described below) from the previous event time t_i as

$$h(t) = o_{i+1} \odot (2\sigma(2c(t)) - 1), \qquad t \in (t_i, t_{i+1}] \ . \tag{4.4}$$

Given the history up through t, the conditional intensity of an event at time t, $\lambda(t)$, is continually obtained from the continuous hidden state $h(t)$ through a non-linear transfer function g with parameters W_t:

$$\lambda(t) = g(W_t^\intercal h(t)) \ . \tag{4.5}$$

This intensity function describes when the next event will occur (see Equation 4.2). Once it occurs, the hidden state at the event time, $h(t_{i+1})$, is used to determine the distribution over the mark, k_{i+1}, as described in Section 4.4.2.

At an event time t_i, the cell state changes discontinuously, based on the mark produced, $k_i = (e_i, v_i)$; the current value of the cell, $c(t_i)$; and the target steady-state value for the cell computed at the last event, $\bar{c}_i$. The updates of the fundamental components of

the continuous-time LSTM network are given through the equations

$$i_{i+1} \leftarrow \sigma(W_i e_i + W_{vi} v_i + W_{hi} h(t_i) + b_i) \tag{4.6a}$$

$$\bar{i}_{i+1} \leftarrow \sigma(W_{\bar{i}} e_i + W_{v\bar{i}} v_i + W_{h\bar{i}} h(t_i) + b_{\bar{i}}) \tag{4.6b}$$

$$f_{i+1} \leftarrow \sigma(W_f e_i + W_{vf} v_i + W_{hf} h(t_i) + b_f) \tag{4.6c}$$

$$\bar{f}_{i+1} \leftarrow \sigma(W_{\bar{f}} e_i + W_{v\bar{f}} v_i + W_{h\bar{f}} h(t_i) + b_{\bar{f}}) \tag{4.6d}$$

$$z_{i+1} \leftarrow 2\sigma(W_z e_i + W_{vz} v_i + W_{hz} h(t_i) + b_z) - 1 \tag{4.6e}$$

$$o_{i+1} \leftarrow \sigma(W_o e_i + W_{vo} v_i + W_{ho} h(t_i) + b_o) \tag{4.6f}$$

$$c_{i+1} \leftarrow f_{i+1} \odot c(t_i) + i_{i+1} \odot z_{i+1} \tag{4.6g}$$

$$\bar{c}_{i+1} \leftarrow \bar{f}_{i+1} \odot \bar{c}_i + \bar{i}_{i+1} \odot z_{i+1} \tag{4.6h}$$

$$\delta_{i+1} \leftarrow g(W_d e_i + W_{vd} v_i + W_{hd} h(t_i) + b_d) \quad . \tag{4.6i}$$

Ws and bs denote parameters of the model. Other quantities with subscripts exist only at event times. Subscripts of $i+1$ denote those quantities produced just after event i and are used to describe the rate and distribution for event $i+1$. These equations are represented by the purple circles in Figure 4.1.

These updates resemble the discrete-time LSTM except for the extra update equations for the initial and target cell state and the decay rates (the last three equations). The fundamental difference, however, is that the previous hidden state obtained just after reading the input tuple (k_{i-1}, t_{i-1}) is not used in the update equations, rather it uses the decayed value of $h(t_i)$ at time t_i, after it has decayed for the period $(t_i - t_{i-1})$ through the decay rates δ_i.

For all non-linear transfer functions denoted by g in the above equations, we have used scaled softplus function $g(x) = s\log(1+\exp(x/s))$, each of which has scale s as separate learnable parameters.

4.4.2 Mark Distribution

The mark distribution $P(e_i, v_i|\mathcal{H}_i, t_i)$ is also obtained from the decayed hidden state $h(t_i)$. It is modeled as a mixture of joint Bernoulli-Normal distributions:

$$p(e_i, v_i|\mathcal{H}_i, t_i) = \sum_{m=1}^{M} \pi_m P_m^{(e)}(e_i|h(t_i)) p_m^{(v)}(v_i|h(t_i)) \tag{4.7}$$

where

$$\pi_m = \frac{e^{W_m^\intercal h(t)}}{\sum_{m'} e^{W_{m'}^\intercal h(t)}} \ . \tag{4.8}$$

The mixture weights are denoted by π_m. Given a mixture component and the history, the distribution over the measurement types recorded, e, and the values recorded for those measurements, v, are independent. The probability $P_m^{(e)}(e|h)$ is a Bernoulli distribution:

$$P_m^{(e)}(e|h) = \prod_{j=1}^{|L|} [\sigma(W_{jm}^\intercal h)]^{e_j} [1 - \sigma(W_{jm}^\intercal h)]^{1-e_j}, \tag{4.9}$$

where σ is the sigmoidal transfer function and W_{jm} controls the probability of event type $j \in L$ under mixture component m. The density for the values under mixture component m is a normal distribution:

$$p_m^{(v)}(v|h) = \prod_{j=1}^{|L|} \mathcal{N}(v_j, W_{\mu_j m}^\intercal h, g(W_{\sigma_j m}^\intercal h)), \tag{4.10}$$

where the value of event type j under mixture component m is modeled using a normal distribution with mean $W_{\mu_j m}^\intercal h$ and variance $g(W_{\sigma_j m}^\intercal h)$. The function g is a non-linear

transfer function keeping the variance positive, and $W_{\mu_j m}$ and $W_{\sigma_j m}$ are the parameters of the linear and non-linear transfer functions respectively.

4.4.3 Log-likelihood

The log-likelihood of the event stream S over the finite time interval T is given by

$$\log(p(S|\mathcal{W})) = \sum_i [\log \lambda(t_i; \mathcal{W}) + \log p(e_i, v_i | h(t_i); \mathcal{W})] - \int_{t=0}^{T} \lambda(t; \mathcal{W})\, dt \tag{4.11}$$

where $\mathcal{W}$ is the set of all parameters including the LSTM parameters and the parameters controlling the dynamics, such as the mixture weights, event intensities, and the joint density of the events and their values. The $-\int_{t=0}^{T} \lambda(t)dt$ term integrates the log-probability of the infinitely many non-events that did not occur within the the time interval T. This loss is represented by the blue boxes in Figure 4.1.

The integral can be approximated by Monte Carlo sampling. The total log-likelihood can be maximized by any stochastic gradient method to learn the parameters in $\mathcal{W}$. In our experiments, we have minimized $-\log(p(S|\mathcal{W}))$, or the negative log-likelihood loss (nllh).

4.4.4 Jointly learning to classify

Given a finite event stream, $S_T = (t_1, k_1), (t_2, k_2), \ldots (t_T, k_T)$ as input to a continuous-time LSTM network, the hidden state $h(t)$ is updated continuously as the network processes the input stream. At any point in time it can be passed through a multilayer-perceptron (MLP) network to obtain desired outputs for solving a classification or regression problem.

Or, the final $h(T)$ obtained after processing sequences up to a finite time T can be used to sample times, values, and the types of future events in a sequence.

We use an sigmoidal transfer function for the output layer of the MLP network for a binary classification task, and a softmax transfer function for a multilabel task. We evaluate our method in the context of four clinically relevant classification tasks by jointly minimizing the negative log-likelihood loss $-\log(p(S|\mathcal{W}))$, denoted by $\mathcal{L}_S$ and the cross-entropy losses for the respective tasks. In case of binary classification, the classification loss is

$$\mathcal{L}_{pb} = -y \cdot \log y' + (1-y) \cdot \log(1-y'), \tag{4.12}$$

where $y' = \sigma(W_{pb}^{\intercal} h(t'))$ is the predicted probability of the positive class at time t' obtained from a multilayer perceptron network with weights W_{pb}, $\sigma()$ is the sigmoidal transfer function, and $h(t')$ is the hidden state at prediction time t'.

In case of a multilabel classification problem with K labels, the cross-entropy loss is

$$\mathcal{L}_{pm} = \frac{1}{K} \sum_{k=1}^{K} -y_k \cdot \log y'_k + (1-y_k) \cdot \log(1-y'_k), \tag{4.13}$$

where $y'_k = \text{softmax}(W_{pm}^{\intercal} h(t'))$ is the output probability of the k^{th} class at time t' obtained from the softmax function, and W_{pm} is the weights of the MLP network. This loss is represented by the yellow box in Figure 4.1.

For tasks that require intermediate predictions, we employ the same multilayer perceptron, using the intermediate value of $h(t)$ for prediction time $t < T$. The losses are then averaged across all prediction times. This loss is represented by the green boxes in Figure 4.1.

The final loss to optimize then becomes

$$\mathcal{L} = w_l \cdot \mathcal{L}_S + w_c \cdot \mathcal{L}_c, \tag{4.14}$$

where $\mathcal{L}_c$ is the suitable classification loss and $\mathcal{L}_S$ is the negative log-likelihood. The weights w_l and w_c are the tunable weights to control the relative importance of these two terms. The model is trained to minimize L in an end-to-end fashion.

4.5 MIMIC-III Benchmarks and Task Description

In this section, we describe the MIMIC-III benchmarking tasks used to evaluate our method and the corresponding benchmarking datasets.

Measuring the progress in machine learning for health care research is often hindered due to a lack of benchmarking tasks and datasets. To alleviate the problem, [52] created four clinically relevant benchmarking tasks and curated data for the tasks from the MIMIC-III database, described in section 3.4.2. We have used this benchmarking dataset of 33,798 unique patients with a total of 42,276 hospital admissions and ICU stays and time-series of 17 physiologic variables. We briefly describe the tasks and evaluation metrics below:

4.5.1 In Hospital Mortality

In-hospital mortality prediction is a highly relevant clinical task that is used primarily for patient outcome prediction, triage, stratification of clinical trials, and quality-of-care comparison among ICUs. It involves predicting the outcome o the patients, i.e., death during the hospital stay or discharge as alive. Among the benchmark cohort, the mortality

rate was 13%. We frame the problem as a binary classification task, and the input to our models is data from the first 48 hours of ICU-stay of the patient. To compare to previous methods, we also used the Area Under the Curve for the Receiver Operating Characteristic plot (AUC-ROC) and the Area Under the Curve for the Precision-Recall plot (AUC-PRC) as evaluation metrics.

4.5.2 Phenotyping

Another benchmarking task used in our experiments is phenotyping, which is the classification of acute-care phenotypes. The task involves retrospectively predicting 25 common conditions the patient might have had using all data from admission to discharge. The conditions include 12 critical conditions such as respiratory/renal failure, eight chronic conditions such as diabetes, and 5 "mixed" conditions such as liver infections. As the patient can be involved with multiple conditions, we frame the task as a multilabel classification problem. Following [75] and [52], we use macro and micro-AUC-ROC as our evaluation metrics. While the former averages the AUC for all tasks, the latter combines all tasks and calculates a single AUC.

4.5.3 Decompensation

Treatment planning for deteriorating patients requires early warning scores to be computed based on accurate prediction of mortality within a fixed time window, for example, 24 hours after assessing a patient. We frame the task as a binary classification problem, where, at each hour, the task is to predict whether the patient would die in the next 24

Table 4.1: Task Specific Sample Sizes for MIMIC-III benchmarking dataset

Task	Train	Validation	Test
Mortality	14681	3222	3236
Phenotyping	29242	6371	6281
Decompensation	29242	6371	6281
Length of Stay	29242	6371	6281

hours. In our experiments, we perform this binary prediction at timepoints closest to the top of the hour mark. Thus, we roughly capture the same distribution used in [52], where the measurements are divided into hourly bins, and a binary prediction is performed at each of these hourly windows. We used the same evaluation metrics for the in-hospital mortality task hold in the decompensation prediction task.

4.5.4 Forecasting Length of Stay

Forecasting length of stay is an essential clinical task as early identification of patients who would need lengthy stays help to manage hospital resources. For each measurement time, the remaining length of stay is transformed into one of ten buckets to pose the problem as a multilabel classification task. The buckets are one bucket for less than a day, seven buckets of the length of a day for each day of the first week, and two outlier buckets — one for stays more than a week but less than two weeks, and one for stays higher than two weeks [52]. Similar to the decompensation task, we only predict timesteps closest to the top of the hour mark. We use Cohen's linear weighted kappa [26] for our evaluation metric, which measures the correlation between ordered items.

4.6 Experiments

We discuss the experimental setup, datasets, baseline models, and the results in this section.

4.6.1 Experimental Setup

A 85%-15% training-test split is defined by the benchmarking data. We further split the training set into 82% for model training and 18% for model hyper-parameter validation. The total number of predictions for each task are shown in Table 4.1. We kept a strict divide between training and testing data, and the final test set was not used in *any* part of our experiment until our final evaluation, which is reported in this thesis.

We selected the best possible hyperparameter combination by performing a grid search over possible hyperparameter values and choosing the best configuration based on performance in the validation set. The set of hyperparameters included the number of cells in the continuous-time LSTM layers, number of layers (one or two) in the dense network, number of units in the dense network, number of mixture components, learning rate, dropout probability, and weights of the two losses, w_c and w_l in case of joint learning (Equation 4.14). We pick the number of cells in the continuous-time LSTM layers from $\{64, 128, 256, 512\}$, the number of units in the dense layers from $\{16, 32, 64\}$ and the number of mixture components from $\{1, 4, 8, 12\}$. In all cases we use the Adam optimizer [67] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$. We used dropout [117] only in the fully connected layers. After cross-validation, we set the learning rate to 0.001 for in-hospital mortality tasks and 0.005 for all other tasks. For all of our experiments using the validation set, we have found

adjustments to w_c and w_l have minimum effect on prediction performance in case of joint learning and as a result set both of them to 1. We use early stopping based on the validation set accuracy and train our models for a maximum of 30 epochs.

4.6.2 Baselines and Our Model

We compare our methods with the state-of-the-art models provided in [52] including generalized linear models such as logistic regression and deep learning methods based on LSTM networks. Below we briefly describe these methods, along with the two versions of our modified neural Hawkes process that we tested.

Logistic Regression (LR)

Six different sample statistic features on seven different subsequences of the 17 time series variables are used as features to a logistic regression model. The statistical hand-engineered features from each sub-sequence include minimum, maximum, mean, standard deviation, skew, and the number of measurements within the subsequences.

Standard LSTM (S)

Each time series is split into regularly spaced hourly intervals. Only the value of the last measurement is used if multiple measurements exist in the same interval. Missing values are imputed using the value of the last measurement if exists and a pre-specified 'normal' value otherwise. Binary masks are also used as inputs which indicate which values are 'true' and which are imputed measurements. Z-score normalization is performed after missing value imputation.

Channel-wise LSTM (C)

This is a modified LSTM model which uses additional bi-directional LSTM layers for pre-processing. Separate bi-directional LSTM layers independently pre-process the masks and each imputed time-series split into regularly spaced hourly intervals. Then, the outputs of these LSTM layers are concatenated and fed into the final LSTM model for classification.

Standard LSTM with Deep Supervision (S+DS)

For in-hospital mortality and phenotype predictions tasks, deep supervision is performed by replicating the labels at each time-step. A modified cost function controls the strength of supervision and learns to predict the replicated timesteps too. This helps to mitigate the difficulty of passing information across many time-steps, which is particularly an issue in case of long sequences. For length-of-stay and decompensation prediction tasks, both the standard LSTM model and the channel-wise model perform predictions on multiple prediction instances created from the same ICU stay (each hourly prediction is treated as a separate prediction instance). Deep supervision in this case is achieved when these samples derived from the same ICU stay are grouped together and prediction is performed in a single pass instead of treating them as separated prediction instances.

Channel-wise LSTM with Deep Supervision (C+DS)

The deep supervision is similar to the above, however the channel-wise LSTM model is the base architecture instead of a regular LSTM network.

Neural Hawkes maximizing prediction only (NH:pred)

We use the same architecture as neural Hawkes process described in section 4.4.1, however instead of minimizing $-\log(p(S|\mathcal{W}))$ in 4.14, we only learn to minimize the cross-entropy losses for the classification tasks. In terms of Figure 4.1, we only use loss associated with the green (for intermediate predictions) or yellow (for sequence predictions) losses. Thus, we directly optimize the predictions, and do not consider how well the underlying neural Hawkes process matches the event sequences.

Neural Hawkes maximizing prediction score and sequence likelihood (NH:pred+llh)

In this version, we jointly learned to minimize $-\log(p(S|\mathcal{W}))$ along with the cross-entropy losses described in 4.4.4 (that is, the blue loss functions from Figure 4.1, as well as the green or yellow boxes). The final loss function to minimize is Equation 4.14. Note that deep supervision is implicitly built into this method as in case of decompensation and length-of-stay, we do not create multiple instances from the same stay but predict them in a single pass and also have the loss functions for estimating $p(e, v)$ and $\lambda(t)$. For in-hospital mortality and phenotyping, we do not replicate the labels but the model learns to predict the next timestep, events, and the values of the immediate next measurement.

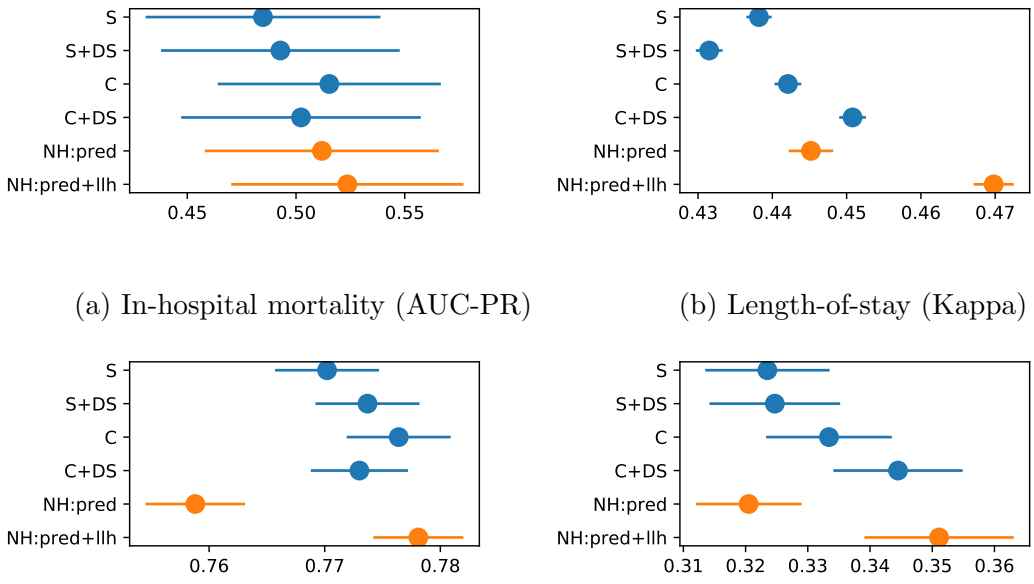
We do not include the multitask baselines proposed in [52] in our discussion because, in the later version of that paper, apart from the in-hospital mortality task, the channel-wise and deep supervision techniques outperform multitask learning. The reason is the regularization capacity for multitask learning gets mitigated by channel-wise preprocessing and deep supervision for single task models. Similarly, in our case, learning the

distribution of the sequence alongside minimizing the prediction loss acts as a surrogate task to improve the prediction performance. We discuss this in detail in section 4.6.7.

In the following subsections, we provide comparisons among the above baselines and discuss the results for the four benchmarking tasks. We also provide a 95% confidence interval estimation in the results tables. Following [52], we resample the test set K times and use 2.5 and 97.5 percentiles of the predicted scores in the resampled set as our confidence interval estimate. For in-hospital mortality and phenotype prediction tasks, we select K as 10000, and for decompensation and length-of-stay, K is 1000.

4.6.3 Results for In-Hospital Mortality Task

The results for the in-hospital mortality task are provided in Table 4.2 and Figure 4.2(a). We notice that the neural Hawkes process, trained only on the prediction loss (NH:pred) has a similar AUC-ROC score to the discrete LSTM (S). However, it achieves much better area under the precision-recall curve. The neural Hawkes model which jointly learns the distribution of the sequence alongside the cross-entropy loss for prediction (NH:pred+llh) has similar AUC-ROC with the channel-wise model, which improves upon the discrete LSTM baseline. However, the NH:pred+llh model achieve the best area under the precision-recall curve among all competing methods. Also, the performance improvement by joint learning is noticeable as NH:pred+llh is significantly better than NH:pred according to both comparison metrics. The best performing NH model has one recurrent layer with 128 units, two layered MLP network at the end with 16 units each with no dropout and 12 mixture components.



(a) In-hospital mortality (AUC-PR)

(b) Length-of-stay (Kappa)

(c) Phenotyping (Macro AUC-ROC)

(d) Decompensation (AUC-PR)

Figure 4.2: Plots of the average performance and ranges for (a) in-hospital mortality AUC-PR, Table 4.2, (b) length-of-stay Kappa, Table 4.3, (c) phenotyping Macro AUC-ROC, Table 4.4, and (d) decompensation AUC-PR, Table 4.5. We omitted logistic regression (LR), as it consistently did not perform as well as the other methods.

Table 4.2: In-Hospital Mortality Prediction Task Results

Model	AUC-ROC	AUC-PR
LR	0.8485 (0.8279, 0.8682)	0.4744 (0.4188, 0.5293)
S	0.8547 (0.8349, 0.8732)	0.4848 (0.4308, 0.5372)
S + DS	0.8558 (0.8362, 0.8750)	0.4928 (0.4379, 0.5486)
C	**0.8623 (0.8436, 0.8807)**	0.5153 (0.4640, 0.5680)
C + DS	0.8543 (0.8340, 0.8734)	0.5023 (0.4472, 0.5544)
NH:pred	0.8547 (0.8350, 0.8737)	0.5119 (0.4581,0.5632)
NH:pred+llh	0.8619 (0.8433, 0.8801)	**0.5236 (0.4702, 0.5730)**

4.6.4 Results for Length-of-stay Task

Results for length-of-stay prediction are provided in Table 4.3 and Figure 4.2(b). As mentioned previously, we perform this prediction at timesteps that are closest to the hourly mark. As many hourly intervals do not contain any measurement and we haven't performed any imputation, the comparison with the baseline models requires a small modification: For hours without measurements, we carried our most recent previous prediction forward to the vacant hour. This allows us to make predictions on exactly the same set of testing data. From the table, we can see that the NH:pred model performs slightly better than the channel-wise model. However, it is slightly worse than the channel-wise model with deep supervision. However, the NH:pred+llh model significantly outperforms all the baseline methods and achieves a kappa score of 0.4698. Similar to in-hospital mortality, the utility of joint learning is evident by comparing the performance of NH:pred and

Table 4.3: Length-of-stay Task Results

Model	Kappa
LR	0.4024 (0.4006, 0.4043)
S	0.4382 (0.4365, 0.4400)
S + DS	0.4315 (0.4297, 0.4334)
C	0.4421 (0.4403, 0.4438)
C + DS	0.4508 (0.4490, 0.4527)
NH:pred	0.4452(0.4422, 0.4461)
NH:pred+llh	**0.4698(0.4671, 0.4716)**

NH:pred+llh. As mentioned previously, the NH:pred is already performing deep supervision by predicting all values for a single ICU stay in one pass. However, jointly learning to minimize the negative log-likelihood (nllh) loss for the sequence distribution further boosts performance. The best performing NH model has one recurrent layer with 512 hidden units, a two-layered MLP network with 32 units and 8 mixture components. Dropout didn't improve performance significantly.

4.6.5 Results for Phenotyping Task

We provide the results for the phenotyping task in Table 4.4 and Figure 4.2(c). The neural Hawkes model which jointly learns the distribution of the sequence alongside the cross-entropy loss for prediction (NH:pred+llh) has similar micro AUC-ROC with the channel-wise model, which improves upon the discrete LSTM baselin. However, it is slightly

better in macro AUC-ROC. The performance of the NH model trained only on the prediction loss is relatively poor among all models. We believe because phenotyping is performed on full patient history and we do not perform discrete time aggregation, the average sequence length is quite high compared to a time-windowed regular LSTM network. This results in difficulty to back-propagate the error without suitable supervision in the intermediate steps. As a result, adding the llh loss to the final loss function causes a performance boost. The best performing NH: pred+llh model has one recurrent layer with 512 hidden units and a two-layered MLP network with 32 units each and no mixture components. Note that the performances of all the baseline models and our NH:pred+llh model in this task are highly similar. This could because phenotyping is a multi-label classification problem where the model is trained to simultaneously predict the probabilities for all the 25 phenotypes. This can be considered a multi-task learning and has a regularization effect itself. As a result, all the other regularization techniques do not provide significantly better performance boost over a regular LSTM model.

4.6.6 Results for Decompensation Task

We provide the results for the decompensation task in Table 4.5 and Figure 4.2(d). Similar to the length of stay model, to make comparisons for hours without measurements, we carried our most recent previous prediction forward to the vacant hour. The neural Hawkes model (NH:pred+llh) has the best AUC-ROC AUC-PR among all models. The NH:pred model has similar performance with the regular LSTM model. The channel-wise model with deep supervision has the best performance among the other baseline models.

Table 4.4: Phenotyping Task Results

Model	Macro AUC-ROC	Micro AUC-ROC
LR	0.7385 (0.7338, 0.7431)	0.7995 (0.7961, 0.8030)
S	0.7702 (0.7657, 0.7747)	0.8212 (0.8177, 0.8246)
S + DS	0.7737 (0.7692, 0.7781)	0.8231 (0.8196, 0.8265)
C	0.7764 (0.7719, 0.7806)	**0.8251 (0.8217, 0.8284)**
C + DS	0.7730 (0.7688, 0.7773)	0.8224 (0.8191, 0.8257)
NH:pred	0.7588 (0.7545, 0.7610)	0.8114 (0.8096, 0.8239)
NH:pred+llh	**0.7781(0.7742, 0.7825)**	0.8240 (0.8199, 0.8273)

Again, the utility of joint learning is evident from the performance improvement obtained from the NH:pred+llh model over the NH:pred model.

4.6.7 Joint Learning Prevents Overfitting

To investigate the utility of jointly minimizing the negative log-likelihood of the sequences along with minimizing the binary cross-entropy loss for in-hospital mortality prediction, in Figure 4.3 we plot the evolution of training and validation losses as training progresses. As the figure suggests, without the intermediate losses at each timestep for learning the distribution $P(e, v|h)$ and the intensity function $\lambda(t)$, the NH:pred model minimizing only the binary cross entropy loss quickly starts to overfit and prediction performance suffers. However, jointly optimizing both losses helps prevent over-fitting and the

Table 4.5: Decompensation Task Results

Model	AUC-ROC	AUC-PR
LR	0.8700 (0.8666, 0.8734)	0.2138 (0.2054, 0.2227)
S	0.8917 (0.8886, 0.8950)	0.3235 (0.3135, 0.3326)
S + DS	0.9039 (0.9008, 0.9069)	0.3247 (0.3142, 0.3349)
C	0.9056 (0.9026, 0.9086)	0.3334 (0.3233, 0.3440)
C + DS	0.9106 (0.9076, 0.9133)	0.3445 (0.3341, 0.3541)
NH:pred	0.8905 (0.8830, 0.8935)	0.3205 (0.3120, 0.3317)
NH:pred+llh	**0.9186(0.9135, 0.9230)**	**0.3511 (0.3391, 0.3627)**

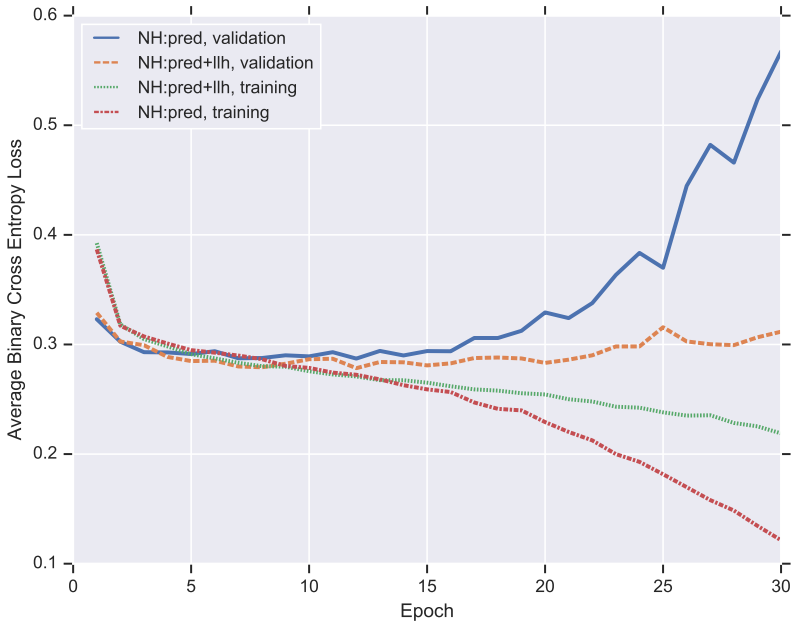


Figure 4.3: Training and validation loss comparison as training progresses

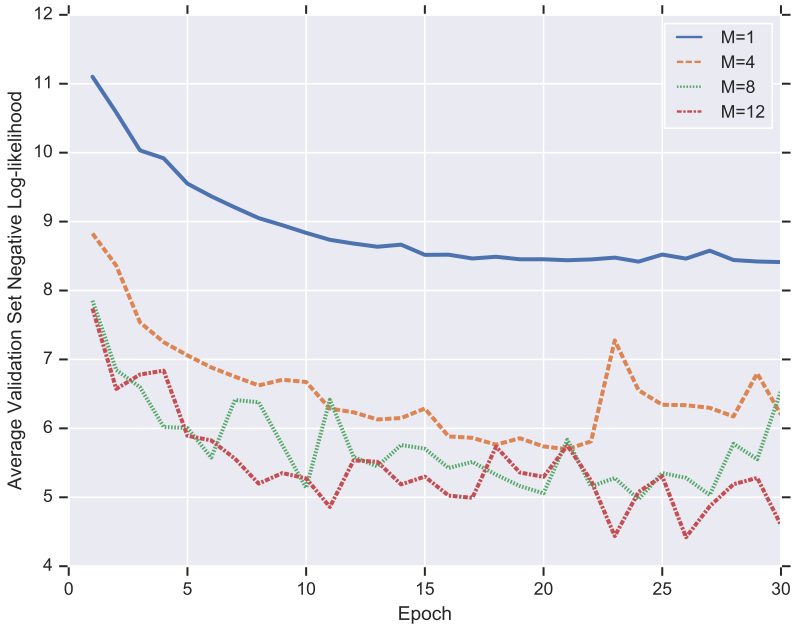


Figure 4.4: Average validation loss decreases as the number of mixture components for the mark distribution is increased.

negative log-likelihood term $-\log(p(S_T|\mathcal{W}))$, denoted by $\mathcal{L}_S$ in the final loss function in Equation 4.14, implicitly acts as a regularizer. This explains the superior performance of the NH:pred+llh model in classification over the NH:pred model.

4.6.8 Mixture Components lower Negative Log-likelihood Loss

For all the tasks, we have chosen the number of mixture components on the basis of performance on the validation set. However, particularly for the in-hospital mortality and the phenotyping task, we noticed that changing the number of mixture components didn't have a significant effect on classification performance. This could be due to the fact that the $\mathcal{L}_S$ term in the final loss function in Equation 4.14 has similar regularization capacity for any number of mixture components. However, the utility of adding the mixture components

is more pronounced when considering the generative capacity of the models. To investigate this, for the in-hospital mortality task, we plot the average negative log-likelihood (nllh) loss in the validation set for different number of mixture components, M, as training progresses. As Figure 4.4 shows, without the mixture components, the nllh loss is significantly higher. However, as the number of mixture components is increased, the nllh loss drops is the lowest at $M = 12$. The convergence behavior is similar for all cases.

4.7 Summary

We have demonstrated new state-of-the-art results on benchmark clinical prediction tasks. Our results use a neural Hawkes process [82] which we modified to allow for multiple simultaneously recorded measurements values, which we augmented for prediction tasks (not just fitting sequence log-likelihood).

Fitting a combination of prediction loss and log-likelihood loss allowed for better testing prediction performance. Previous approaches to this clinical prediction problem did not build on generative models and did not address the irregularly spaced time intervals directly. Therefore, they were not able to benefit from the implicit regularization of including the sequence likelihood in the loss function. We demonstrated the value of generative models, particularly for clinical time series classification problems in which data are (relatively) sparse and irregular.

Chapter 5

Neural Stochastic Differential Equations with Bayesian Jumps for Marked Temporal Point Process

5.1 Summary

Many real-world systems evolve according to continuous dynamics and get interrupted by stochastic events i.e. systems that both flow (often described by a differential equation) and jump. If the equation of the continuous motion is unknown, the stochastic event generation process can be modeled as samples generated from a *marked temporal point process*, in the form of event sequences with *non-uniform* time intervals. Additionally, each event is marked with the type of the event and a real-valued *noisy* measurement. The noisy measurements themselves are the realizations of a stochastic process with an unknown

stochastic differential equation (SDE). We present a framework that simultaneously models the intensity function of the temporal point process and learn stochastic process governing the distribution of the observed noisy measurements. Similar to the underlying system of interest, the latent dynamics of our framework evolves continuously according to ordinary differential equations (ODEs) while, the jumps at the observations are controlled by Gated Recurrent Units (GRUs) through a *Bayesian* update, which accounts for the observation noise. We present data fitting results on a real-world medical dataset.

5.2 Introduction and Related Work

The real world has many examples of systems that evolve continuously in time and are interrupted by stochastic events, spaced by non-uniform time intervals. In many domains, such as social networks, global politics, and computer systems, the "equations of motion" are unknown and the unknown dynamics of the underlying system can only be estimated by records of the events and their real-valued times. Data in the medical domain, for example, electronic health records (EHRs), can additionally contain real-valued *noisy* physiological measurements of different variables including vital signs, drugs, and lab tests. These observations are inherently noisy due to charting errors, calibration, or device noise. Our goal is to model the dynamics of such systems where in addition to the real-valued irregularly sampled times, each event is accompanied by the type of the event and also a real-valued noisy measurement.

Given such event sequences with non-uniform time intervals, marked temporal point process and intensity functions are a powerful mathematical framework to model

the event generation process. The measurement variables can be modeled by the Fokker-Plank equation [106] and the measurements viewed as noisy observations of this underlying stochastic process. Recurrent Neural Networks (RNNs) are not a natural fit for modeling such data, as they naturally assume the observations are regularly spaced across the temporal dimension. Usual tricks to adapt RNNs for irregular time series data divide the timeline into equally-spaced time-intervals [76, 52, 118], use the time difference between the observations as an input to the model [16, 31], or decay the hidden state exponentially when no observation is made [82, 16]. However, these approaches either make the simplified "missing at random" assumption, fail to capture the continuously evolving dynamics of the underlying system or use a fixed form of evolution in between observations.

We present a framework based on the recently proposed neural ordinary differential equations (Neural ODEs) [18] for jointly learning the two intertwined stochastic processes, that is the temporal point process generating the stochastic events and the stochastic process generating the associated real-valued noisy measurements. We encode the state of the underlying system in a latent state vector $h(t)$, which evolves piecewise-continuously according to a neural ODE. The jumps at the observation times are controlled by GRUs [20], which changes the trajectory of $h(t)$. Additionally, a Bayesian update term accounts for the noise in the measurements to learn the true distribution of the underlying multivariate stochastic process of the measurement variables. The intensity function of the point process and the mark distributions are generated from $h(t)$ with additional neural nets.

In a neural ordinary differential equation (Neural ODE) framework, the differential equation expressing the flow dynamics is parameterized by a neural network without consid-

ering potential jumps of the latent state. Jia et al. [61] address the jumps using a neural net and also an adjoint-operator formulation of the gradient for learning in constant memory and model the intensity of point processes and distribution of real-valued features. However, they ignore the stochasticity of the underlying process generating the noisy measurements. Brouwer et al. [13] proposes a Bayesian update network using GRUs to regularize the neural ODE to better track a true belief state for the underlying system without directly addressing the event generation process concerning event history. Rubanova et al. [107] proposes an ODE-RNN hybrid where the hidden state dynamics flow according to a neural ODE and the jumps according to a GRU unit. They applied their model for learning the intensity function without accounting for the event history and did not address the noise in the measured observations. We address all these issues in modeling noisy, irregularly-sample time series altogether using a unified framework combining the above ideas.

Finally, we present preliminary data fitting results on a real-world electronic health records from the MIMIC-III ('Medical Information Mart for Intensive Care') database [65], a publicly available critical care database.

5.3 Preliminaries

We want to model the dynamics of a hybrid system whose samples contain a finite set of events of unbounded random size; each event has a real-valued time and a mark consisting of a discrete-valued label and a real-valued noisy measurement of that label. We use the same notations as 4.3 for the subsequent discussion.

We postulate that the measurements $\boldsymbol{v_i}$ are observations of an L-dimensional continuous stochastic process $\mathcal{V}(t)$, which is driven by an unknown stochastic differential equation (SDE):

$$d\mathcal{V}(t) = M(\mathcal{V}(t))dt + \Sigma(\mathcal{V}(t))dW(t), \tag{5.1}$$

where $dW(t)$ is a Wiener process. The distribution of $\mathcal{V}(t)$ evolves according to the Fokker-Planck equation. The continuous mean and the covariance function for the probability density function (PDF) of $\mathcal{V}(t)$ is denoted by $M(t)$ and $\Sigma(t)$ in equation 5.1. From the noisy measurements observed at irregular times, we need to model the two unknown mean and the covariance function. Note that, observations from all dimensions are not sampled at the same time, which results in the missing values in v_i. Additionally, we assume the observation noise in the sampled measurement for dimension l is drawn from a Gaussian distribution with zero mean and standard deviation of σ_l^{obs}.

We postulate that the observation times are drawn from a general continuous-time point process, whose intensity may depend on the history of both $\mathcal{V}$ and the previous observation times. Its conditional intensity function, $\lambda(t, \mathcal{H})$, may depend on both the raw time t, and the history up through time t, $\mathcal{H}$. If we let $\mathcal{H}_i$ be the observed history up to but not including event i, $\mathcal{H}_i = \{(t,k) \mid (t,k) \in S \wedge t < t_i\}$, the density of the distribution over the time of event i, given the history is:

$$P(t_i \mid \mathcal{H}_i) = \lambda(t_i, \mathcal{H}_i) e^{-\int_{t_{i-1}}^{t_i} \lambda(s, \mathcal{H}_i)\, ds} \,. \tag{5.2}$$

For a marked point process observed within a time interval $[0, T)$, the number of observed events I is finite, and the continuous log-likelihood of such a finite event-stream

is given by,

$$\sum_{i=1}^{I} [\log P(t_i \mid \mathcal{H}_i) + \log P(k_i \mid \mathcal{H}_i, t_i)] + \log P(t_{I+1} > T \mid \mathcal{H}) \ . \tag{5.3}$$

The middle $P(k_i \mid \mathcal{H}_i, t_i)$ term is a combination of the distribution for the event label, $\boldsymbol{e_i}$, and the complex marginal distribution over $\mathcal{V}(t_i)$ conditioned on $\mathcal{H}_i$, as implied by the Fokker-Planck equation. We assume, the distribution for the event label, e_i, is drawn from its own distribution conditioned on history: $P(\boldsymbol{e_i} \mid \mathcal{H}_i, t_i)$. As the labels are discrete-valued, this distribution could be folded in to the intensity, to create a separate intensity for each event label possibility (with the total intensity being the sum of the intensities over all possible labels). However, for our model, a global intensity and a multinomial distribution over labels is more natural.

If the SDE (Equation 5.1) were known, the inference would depend on tracking the belief state of $\mathcal{V}$: $P(\mathcal{V}(t) \mid \mathcal{H})$. Rather than parameterizing and estimating the SDE and then developing an (almost assuredly approximate) belief-state filtering method for $\mathcal{V}$, we propose to learn the belief-state filter directly, never explicitly modeling the underlying SDE. To this end, we next describe a continuous-time neural ODE process with jumps that we train to track both the SDE and the point process. Thus, its hidden state, h, serves as a sufficient summary of the history to model the belief state of $\mathcal{V}$ and the history, $\mathcal{H}$, as necessary for the conditional intensity, $\lambda(t, \mathcal{H})$.

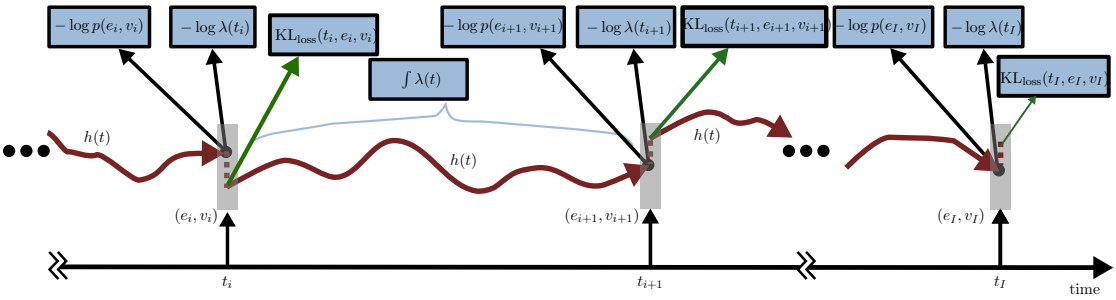


Figure 5.1: Illustration of our proposed framework. Just before processing input observation (e_i, v_i), the negative log-likelihood loss for the intensity and the mark distribution is calculated from the continuous hidden state, obtained from the solution of the ODE solver at t_i. After the GRU (shown by the gray box) processes the observation, the hidden state jumps to the new state (indicated by the brown dots) and the KL divergence loss (shown by the green arrows) is computed (Equation 5.13).

5.4 Proposed Framework

Figure 5.1 schematically shows the components of our framework. We describe each component in the sections below.

5.4.1 ODE-GRU

Recently proposed Neural ODEs [18] are a family of continuous-time models where the hidden state $h(t)$ is defined as a solution to an ODE initial-value problem (IVP):

$$\dot{h(t)} = f_\theta(h(t), t), \tag{5.4}$$

where $h(t_0) = h_0$ and $f_\theta(h(t), t)$ is a multi-layer perception (MLP) with parameters θ.

The continuous hidden state $h(t)$ can be evaluated at any set of times using a

numerical ODE solver:

$$h_0, ..., h_I = ODESolve(f_\theta, h_0, (t_0, ..., t_I)). \tag{5.5}$$

Similar to the ODE-RNN model [107], the hidden state in our framework evolves according to an ODE in between the noisy observations, and the solution to the ODE at time t is the input to an RNN unit that provides the jump at observations. For the RNN unit, we choose the Gated Recurrent Unit (GRU) [20]. This jump allows the hidden state to incorporate the observation and track the belief state of the underlying system. Formally,

$$h_i = ODESolve(f_\theta, h'_{i-1}, (t_{i-1}, t_i)), \tag{5.6}$$

where $h'_i = GRU(h_i, k_i)$. That is, h_i is the hidden state before the observation at time t_i, and h'_i is the hidden state after the observation at time t_i.

The hidden state is coupled with the processes through four MLPs, g^λ, g^e, g^μ, and g^{var}:

$$\lambda(t) = g^\lambda(h(t)) \tag{5.7}$$

$$p(\boldsymbol{e_i} \mid t_i, h(t_i)) = \mathcal{B}\left(\boldsymbol{e_i}; g^e(h(t_i))\right) \tag{5.8}$$

$$p(\boldsymbol{v_i} \mid t_i, h(t_i)) = \mathcal{N}\left(\boldsymbol{v_i}; g^\mu(h(t_i)), g^{\text{var}}(h(t_i))\right) \tag{5.9}$$

where g^λ maps h to the intensity of an observation event (a non-negative scalar), g^e maps h to $|L|$ independent Bernoulli distributions (that is, the $|L|$ independent probabilities that each label appears at an observation), g^μ maps h to $|L|$ means, and g^{var} maps h to $|L|$ variances. For $\boldsymbol{v_i}$, we assume that, conditioned on the hidden state h, each observed label's

value is drawn independently from a normal distribution. Thus, $g_l^e(h_i)$ is the model's predicted probability that label l appears in the observation at time t_i. Similarly, $g_l^\mu(h_i)$ and $g_l^{\text{var}}(h_i)$ are the mean and variance of the predicted normal distribution over the value associated with label i, if it appears in the ith observation.

5.4.2 Maximizing the Log-likelihood

We train the model using maximum likelihood estimation. The log-likelihood of the event stream S over the finite time interval T is derived from Equation 5.2 and 5.3 as:

$$\log(p(S|\mathcal{W})) = \sum_i [\log \lambda(t_i \mid h(t_i); \mathcal{W}) + \log p(k_i \mid h(t_i), t_i; \mathcal{W})] - \int_{t=0}^{T} \lambda(t)\, dt, \qquad (5.10)$$

where $\mathcal{W}$ is the set of all parameters including the ODE-GRU parameters and the MLP parameters controlling the dynamics, such as the event intensities, and the joint density of the events and their values. The $-\int_{t=0}^{T} \lambda(t)\, dt$ term integrates the log-probability of the infinitely many non-events that did not occur within the the time interval T.

The model evolves according to Equation 5.6 where h_i is equivalent to $h(t_i)$, the history, $\mathcal{H}_i$, prior to event i. The intensity for event i and the mark distribution $p(k_i|h(t_i), t_i)$ in Equation 5.10 therefore is obtained from h_i, which is the solution of the ODE-solver with initial value h'_{i-1} (state after the GRU processes event $i-1$), at time t_{i-1}.

5.4.3 Bayesian Jump

We add the Bayesian Jump framework [13] to regularize the system by encouraging its output to track the Bayes filter and account for the noise in the observed measurements. Consider the observation of the lth label's value at time t_i. Just before the observation

event, the model's belief over its value has a mean and variance of $g_l^{\mu}(h_i)$ and $g_l^{\text{var}}(h_i)$. After making the observation, the hidden state jumps to h_i'. The new belief has a mean and variance of $g_l^{\mu}(h_i')$ and $g_l^{\text{var}}(h_i')$. We would like these beliefs to be consistent with a Bayesian update of a belief state, assuming a known zero-mean observation noise with variance σ_{obs}^2.

Assuming we observe value $\boldsymbol{v_{i,l}}$, the posterior distribution over the value of event type l should have mean and variance

$$\mu_{\text{Bayes},l} = \frac{\sigma_{\text{obs}}^2 \cdot g_l^{\mu}(h_i)}{g_l^{\text{var}}(h_i) + \sigma_{\text{obs}}^2} + \frac{g_l^{\text{var}}(h_i) \cdot \boldsymbol{v_{i,l}}}{g_l^{\text{var}}(h_i) + \sigma_{\text{obs}}^2} \tag{5.11}$$

$$\sigma_{\text{Bayes},l}^2 = \frac{g_l^{\text{var}}(h_i) \cdot \sigma_{\text{obs}}^2}{g_l^{\text{var}}(h_i) + \sigma_{\text{obs}}^2} \ . \tag{5.12}$$

If we let $p_{\text{Bayes},l}$ be this Bayesian update normal distribution (with mean $\mu_{\text{Bayes},l}$ and variance $\sigma_{\text{Bayes},l}^2$) and let $p_{i,l}'$ be the normal distribution predicted by the model *after* the GRU update (with mean $g^{\mu}(h_i')$ and variance $g^{\text{var}}(h_i')$), we add a KL-divergence loss to encourage the two distributions to be the same:

$$\text{KL}_{\text{Loss}}(t_i, \boldsymbol{e_i}, \boldsymbol{v_i}) = \sum_l \boldsymbol{e_{i,l}} D_{\text{KL}}(p_{\text{Bayes},l} \parallel p_{i,l}') \,, \tag{5.13}$$

which sums over the individual event type observations, because the model assumes that each component is independent given h. The joint loss to optimize then becomes

$$-\log(p(S|\mathcal{W})) + \sum_i \text{KL}_{\text{Loss}}(t_i, \boldsymbol{e_i}, \boldsymbol{v_i}) \,, \tag{5.14}$$

which can be optimized using gradient descent.

To aid the network in making the correct update, we augment its input. For the i^{th} event, instead of directly feeding the mark k_i to the GRU, $\boldsymbol{v}_i$ is appended with the means and variances of the predicted outputs ($g_l^{\mu}(h_i)$ and $g_l^{\text{var}}(h_i)$ for all l) and the normalized

error terms $(\boldsymbol{v}_{i,l} - g_l^{\mu}(h_i))/\sqrt{g_l^{\text{var}}(h_i)}$ (for all l), which are then multiplied by a dimension specific weight vector W_l and passed through a ReLU non-linearity to produce $\boldsymbol{q}_i$. Non-observed dimensions in $\boldsymbol{q}_i$ is zeroed out and is appended with $\boldsymbol{e}_i$ to be passed as input to the GRU. While this extra information is already available in h_i, the prevents the GRU from needing to learn to recalculate it.

5.4.4 Bayesian ODE-RNN

Instead of only adding the Bayesian jump framework at the observations, we also experiment with a Bayesian architecture that adds uncertainty to the network weights instead. The goal is to achieve regularization through model based uncertainty, and not just observational uncertainty as in the Bayesian Jump framework.

We apply Bayes by Backprop [40, 11] to learn the posterior distributions $\theta' \in \mathbb{R}^d$ of the overall ODE-RNN architecture. θ' consists of the parameters of the GRU architecture, which controls the jump and the parameters of the feed forward net f_θ, which controls the flow of the ODE-architecture. The posterior distribution of the parameters are drawn from a Gaussian with mean parameters. $\mu \in \mathbb{R}^d$ and standard deviation, $\sigma \in \mathbb{R}^d$. The network is trained to minimize the variational free energy:

$$L(\theta') = \mathbb{E}_{q(\theta')}[\log \frac{q(\theta')}{p(y|\theta', x)p(\theta')}], \tag{5.15}$$

where, $p(\theta')$ is a prior on the parameters and $p(y|\theta', x)$ is the log-likelihood of the model. Equation 5.15 is equivalent to maximizing the log-likelihood, $p(y|\theta', x)$, subject to

a KL divergence loss on the parameters of the network that acts as a reguraliser:

$$L(\theta') = -\mathbb{E}_{q(\theta')} \log p(y|\theta', x) + KL[q(\theta')||p(\theta')]. \tag{5.16}$$

Here, $\log p(y|\theta', x)$ is equivalent to the log-likelihood of the sequence S, with respect to the parameters of the whole network $\mathcal{W}$, $p(S|\mathcal{W})$, where, $\theta' \in \mathcal{W}$. So, 5.14, which represents the total loss to optimize now becomes, $\{-\mathbb{E}_{q(\theta')} \log p(S|\mathcal{W}) + KL[q(\theta')||p(\theta')]\}$

5.5 Experiments

One of the prominent use-cases of irregular time series modeling is patient time series data observed in an Intensive Care Unit (ICU) setting, which are highly noisy, sparse, and irregularly sampled. We perform our experiments on a subset of the publicly available MIMIC-III database curated as a benchmark for evaluating machine learning models in healthcare settings [52], containing 17 physiological time-series variables. Unlike the proposed methods in the corresponding paper [52], we do not discretize the data, instead, take into account each time where at least one variable is measured, and represent the measurement variables and their values as e_i and v_i (section: 5.3).

We fit our models on the first 24 hours of data on a training set containing 5000 samples. We report the data-fitting performance on a separate hold-out set with 1000 samples. To properly use batching and achieve speedup in training, we rounded the observation times into the nearest minute and took the aggregate of measurements taking place in the same rounded minute. This results in $(24 * 60 + 1)$ or 1441 possible measurement times. Note that, rounding to the minute results in very little information loss compared to hourly aggregation. All the ODEs in a minibatch are solved continuously for each 1441 measure-

ment times, however, the jumps using the RNN occur only at the observation times for each sample. A separate boolean mask matrix indicates the times where an observation occurred for each sample in the minibatch.

We used ODE Solvers from torchdiffeq python package [18], particularly the fifth-order "dopri5" solver with adaptive step size. The relative and absolute tolerances were set as 1e-3 and 1e-4 respectively. The adjoint method described in [18] and [61] can be used to reduce the memory use, with the cost of added computational time. We used Adam optimizer [67] for learning the parameters of all the MLPs and the RNN. The integral term in equation 5.10 can be directly obtained by augmenting the ODE with an integral over $\lambda(t)$, which is obtained from the hidden state using g^{λ}.

We refer to our model as ODE-RMTPP.It has three versions: ODE-RMTPP without the observational or model uncertainty, ODE-RMTPP + Bayes is the version with the bayesian jump framework, while the Bayesian ODE-RMTPP is the model described in section 5.4.4. We report the average negative log-likelihood (NLLH) loss and also the individual losses: the intensity loss, event loss, and the value loss. LSTM refers to the model proposed by Du et al. [31], and the CTLSTM model refers to the model proposed by [82], based on continuous time Long Short Term Memory cells, described in section 4.4.1.

Results (Table: 5.1) show that adding the Bayes filter improves the data-fitting performance, as the NLLH in the test set reduces for each distribution. The most significant reduction in the NLLH loss occurs for the value distribution. This makes sense, as the Bayes Filter is particularly tracking the belief state for the values. However, the KL divergence loss also implicitly acts as a regularizer, which results in loss reduction for both the intensity

Table 5.1: MIMIC-III Results

Model	Avg. Sequence NLLH	Avg. Event NLLH	Avg. Intensity NLLH	Avg. Value NLLH
ODE-RMTPP	13.98	6.58	3.41	4.12
ODE-RMTPP + BAYES	**13.76**	6.54	**3.39**	**3.83**
Bayesian ODE-RMTPP	13.85	6.43	3.55	3.87
LSTM	14.37	6.72	4.1	3.90
Bayesian LSTM	14.29	6.54	3.76	3.99
CTLSTM	14.22	**6.38**	3.88	3.96
Bayesian CTLSTM	14.15	6.31	3.81	4.1

and the event distribution. Additionally, model uncertainty also helps improve the data fitting performance for all the models, including the baseline LSTM and CTLSTM.

5.6 Discussion and Future Work

We have developed a framework based on Neural Ordinary Differential Equations, with a Bayesian Jump Framework to simultaneously capture the irregularity and noise in time-series data. Also, we experimented with a Bayesian architecture, where uncertainties are imposed on the model weights themselves. Our model is best suited for temporal point process modeling in time-series data where, the irregularly sampled temporal event-sequences are marked with a real-valued noisy measurement, in addition to the type of the event e.g. medical time series.

Chapter 6

A Joint Hierarchical Discrete RNN and Continuous Temporal Point Process Recurrent Model for Time and Item Predictions

6.1 Summary

User behavior in real world transaction systems such as, transaction or music websites can be considered as an irregularly spaced temporal sequence. However, when an user joins a session and engages in the website, the user behavior within each session is likely to be discrete, but the return time of a user to the system or the start of the next session is irregular. We propose a hierarchical recommendation system, where the behavior of users

within each session is captured using a discrete recurrent network architecture and the user return time is modeled using a continuous-time recurrent architecture, which treats each session's starts as samples drawn from a temporal point process. The models are jointly learned to predict the items that the user is likely to consume within each session and when the user is going to return back in the system and create a new session.

6.2 Preliminaries and Problem formulation

The return times of users in online platforms can be considered as an irregular event sequence, where each event can represent the beginning or end of a particular session.

Formally, session data for user u can be represented by, $S_u = \{(b_1, e_1, a_1), (b_2, e_2, a_2), \ldots, (b_{n_u}, e_{n_u}, a_{n_u})\}$, where b_i represents the begin time of session i, e_i represents the end time of session i, a_i is the K dimensional sparse action vector performed by the user within each session, and n_u is the total number of sessions that user u participated in. The goal is to predict gap time $b_i - e_{i-1}$, the initial and subsequent recommendation scores for all the items in session i, from the history $\mathcal{H}_i$.

6.3 The Joint Model

We propose a continuous hierarchical recurrent model (CHRNN), that consists of a discrete recurrent architecture enhanced with a continuous time Long Short Term Memory (CTLSTM) model for predicting the return times of an user. The representations of previous sessions together with contextual information are inputted to the hierarchical model, followed by the item representations within each session.

The CHRNN model is inspired by [122], and consists of an inter-session and an intra-session RNN. The inter-session RNN which is responsible for predicting the return times, using temporal point process loss, is composed of CTLSTM cells [82]. This model is fed with a fixed number of preceding session representations, along with embedding of the gap times of the previous sessions, and other contextual embedding such as, user embedding. Together, they constitute the history $\mathcal{H}_i$, for session i. Conditioned on $\mathcal{H}_i$, the CTLSTM model predicts the next gap time, $b_i - e_{i-1}$, assuming it's drawn from a temporal point process model, and the first item predictions for the next session i.

The intra-sesssion discrete RNN architecture takes the last hidden state of the inter-session RNN, uses it as it's initial state, and take the item embeddings as inputs. For each item embedding, the corresponding output of the discrete RNN is passed to a linear layer, which calculates the scores for each target item. The k items with the highest scores are then selected as recommendations to be given to the user.

6.3.1 Context representation

The context within each session is embedded along with the last hidden state of the intra-session RNN, which then gets inputted to the inter-session RNN. We consider three types of contextual representations, similar to the work of Vassoy et al. [122]. They are:

- User embedding: Unique user embedding to capture the user behavior across all sessions. This helps capturing the effect of long term user behavior in case, some session data are noisy and sporadic

- Item embedding: Item embeddings represent each unique item in the dataset and gets inputted to the intra-session RNN.

- inter-session gap time embedding: The time gaps between each session are also embedded according to [122]. The gaps are first normalized and then divided into unique buckets, IDs of which are then used as index to the embedding layers.

6.3.2 Loss

Similar to [122], we use temporal point process as our temporal model to predict the next session time, conditioned on the history. However, instead of a discrete RNN architecture, we employ the continuous Long Short Term Memory model to continuously learn the underlying intensity function.

Using the temporal point process terminology, the log-likelihood for the session data of all the users can be written as,

$$L_p(S,T) = \sum_u \sum_{i=1}^{n_u} (\int_{e_{i-1}}^{b_i} \lambda_u(t)dt - \log \lambda_u(b_i))) + \int_{t_{n_u}}^{T} \lambda_u(t)dt, \tag{6.1}$$

where T is the terminal time up to which user data is available. The goal is to maximize this likelihood in the training. In prediction time, we can use the learned intensity function to use forward sampling to predict the next return time of the user.

The total loss to optimize then becomes a weighted loss, weighted by hyperparmeter β,

$$L_{total} = -\beta L_p(S,T) + (1-\beta) \sum_{j,l,u} L_{rec}(s_{j+1}, l), \tag{6.2}$$

where $L_{rec}(s_{j+1}, l)$, is the softmax for item l, given session representation at time $j+1$, i.e., s_{j+1}, for each user u. The intra-rnn discrete time model is responsible for optimizing the item prediction loss $\sum_{j,l,u} L_{rec}(s_{j+1}, l)$.

New session times can be forecast from the learned intensities and sample new times in forward using Poisson-process thinning.

6.3.3 Experiments and Results

Dataset

The evaluation is performed on the LastFM dataset [7], containing the listening habits of users on the music website Last.FM. The data was pre-processed according to [122]. First, noisy and irrelevant data were removed. Next, user data were divided into sessions based on an fixed inactivity time period, δt. That is, if an user's consecutive actions happened within the time limit, then they belong to the same session. Formally, for a specified time limit, δt, and a list of user actions, $\{a_{t_0}, a_{t_1}, a_{t_2}, \ldots, a_{t_{nu}}\}$, consecutive interactions a_{t_i} and $a_{t_{i+1}}$ belong to the same session. if $t_{i+1} \leq t_i + \delta t$. Also, if any session consisting of greater than M items are split into two sessions. The resulting dataset statistics are given in table 6.1.

Time Prediction Results

To evaluate how the model is performing for predicting the next time, we train on 80 percent of each user's session, and try to predict the rest of the sessions' timings. We present the time prediction results in Figure 6.1 and compared with [122], who employs a

Number of users	977
Number of sessions	630,774
Sessions per user	645.6
Average session length	8.1

Table 6.1: Statistics of the LastFM dataset after preprocessing

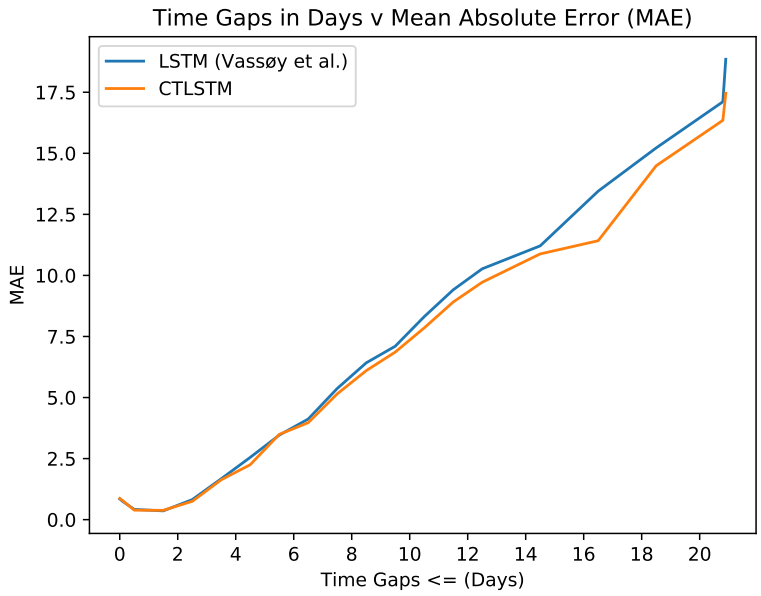


Figure 6.1: Time prediction results for the Last.FM dataset. Mean absolute errors are plotted versus the cumulative number of sessions with time gaps $\leq x$ days

discrete LSTM architecture for the time prediction. As we can see, the continuous LSTM based model as the inter-session gap time predictor outperforms the discrete LSTM based model, as more and more sessions with larger gap times are included. This makes sense, as for longer time periods, the continuous decay property of the architecture is more suitable for capturing long term signals in the data and improve prediction performances for session with longer time gaps.

6.4 Conclusion

We present a hierarchical recurrent neural architecture to jointly improve item prediction performance and predict the next time an user is going to return to the system and join a new session. We improve on the existing literature by replacing the time-predictor inter-session discrete RNN, with a continous LSTM architecture. Our model improves time prediction performance, particularly when user return times are longer.

Chapter 7

Conclusions

Our research's purposes have been two-fold: extend some of the state of the art recurrent neural network architectures and structure learning-based models for temporal point processes and apply the extensions to two application areas: recommendation systems and electronic health records.

From the architectural extension perspective, we extend the structure learning frameworks proposed by [50, 94] to simultaneously model the distributions for the measurement values and the intensities of the events in the same PCIM structure. Then, we extend some of the existing recurrent neural architectures to account for multiple recorded events at the same time. Additionally, we take the architectures' hidden states and learn the parameters of a mixture of Gaussian-Bernoulli distributions for the mark distributions. For the recently proposed ODE-RNN architectures, we propose extensions in the form of observational and model uncertainty. For the observational uncertainty, we combine the works of [107] and [13] to learn the point process intensities and account for the inherent

noise in the measurements. Additionally, we experiment with model-based uncertainties by introducing priors on the weights and optimizing using Bayes by Backprop proposed by [11, 40].

From the perspective of the application, a large chunk of our work has been applied to medical time-series, where patient history in an Intensive Care Unit are stored in the form of electronic health records. First, we apply our structure learning work on an in-hospital-mortality task. Secondly, we apply the continuous RNN-architecture and the extension to jointly learn the temporal point process losses and the classification losses for four benchmarking tasks [52]. We apply the ODE-RNN architecture and the extensions on data fitting of the MIMIC-III dataset. Lastly, we learn a hierarchical architecture to optimize the item prediction losses within each user session at a music website and optimize a temporal point process loss to predict the users' return time on the website, using a continuous LSTM architecture.

One of the weaknesses of our proposed work in Chapter 3 was using a generative model to learn a discriminative task. The weaknesses were partially solved using designing more discriminative tests for the model to choose from. This weakness was resolved using deep neural network architectures, where we simultaneously optimized for the generative and discriminative losses in an end-to-end fashion. The discrete nature of the LSTM architecture, and the fixed form of decay in the CTLSTM architectures, were resolved by using the neural ODE architectures, where the infinitesimal change in hidden state is represented as a neural network, which allows the model to flow in between the observations continuously, without a fixed parametric form.

This thesis can be extended in quite a few directions. First, one of the weaknesses of the ODE based architectures is it is slow training time, which makes it challenging to deploy the models in real-world settings. Lessons learned from the works of [38], can be applied to the extended ODE architectures to reduce training time. Additionally, lessons learned from our modeling techniques can be extended to more recent transformer-based architectures to see whether classification and modeling performances improve over the RNN based models. PCIMs can be extended to allow discriminative training, which is more suitable for the classification tasks we performed. Further works can also be done on the explainability of the models by combining the predictive power of the structure learning-based models and the predictive capacity of the deep learning architectures. Additionally, research in the explainable AI domain can be applied solely on the RNN architectures to explain the RNN models' behaviors.

Bibliography

[1] Melissa Aczon, David Ledbetter, L Ho, Alec Gunny, Alysia Flynn, Jon Williams, and Randall Wetzel. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *arXiv preprint arXiv:1701.06675*, 2017.

[2] Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ArXiv*, abs/1701.07875, 2017.

[4] Alan R Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

[5] Emmanuel Bacry, Anamaria Iuga, Matthieu Lasnier, and Charles-Albert Lehalle. Market impacts and the life cycle of investors orders. *Market Microstructure and Liquidity*, 1(2), 2015.

[6] Inci M. Baytas, Cao Xiao, Xi Sheryl Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. Patient subtyping via time-aware LSTM networks. In *KDD*, 2017.

[7] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR*, 2011.

[8] D. Blei, L. Carin, and D. Dunson. Probabilistic topic models. *IEEE Signal Processing Magazine*, 27(6):55–65, Nov 2010.

[9] David M Blei and John D Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10(71):34, 2009.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[11] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. 2015. In Proceedings of the 32nd International Conference on Machine Learning (ICML 2015).

[12] Michael S. Branicky. Introduction to hybrid systems. In *Handbook of Networked and Embedded Control Systems*, 2005.

[13] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *CoRR*, abs/1905.12374, 2019.

[14] C.Ertekin, Cynthia Rudin, and Tyler H. McCormic. Reactive point processes: A new approach to predicting power failures in underground electrical systems. 2015.

[15] Zhengping Che, Yu Cheng, Shuangfei Zhai, Zhaonan Sun, and Yan Liu. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. *2017 IEEE International Conference on Data Mining (ICDM)*, pages 787–792, 2017.

[16] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *arXiv preprint arXiv:1606.01865*, 2016.

[17] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. Hierarchical deep generative models for multi-rate multivariate time series. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 784–793, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[18] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

[19] Li-Fang Cheng, Gregory Darnell, Bianca Dumitrascu, Corey Chivers, Michael Draugelis, Kai Li, and Barbara E. Engelhardt. Sparse multi-output Gaussian processes for medical time series prediction. 2017.

[20] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *ArXiv*, abs/1406.1078, 2014.

[21] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Doctor AI: predicting clinical events via recurrent neural networks. In *Proceedings of the 1st Machine Learning in Health Care, MLHC 2016, Los Angeles, CA, USA, August 19-20, 2016*, pages 301–318, 2016.

[22] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1495–1504, New York, NY, USA, 2016. ACM.

[23] Edward Choi, Siddharth Biswal, Bradley A Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete electronic health records using generative adversarial networks. *CoRR*, abs/1703.06490, 2017.

[24] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[25] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv*, abs/1412.3555, 2014.

[26] J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

[27] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes. Vol. I.* Probability and its Applications (New York). Springer-Verlag, New York, second edition, 2003. Elementary theory and methods.

[28] Thomas L Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *AAAI*, pages 524–529, 1988.

[29] Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988.

[30] Franck Dernoncourt, Ji Young Lee, Özlem Uzuner, and Peter Szolovits. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association : JAMIA*, 24 3:596–606, 2016.

[31] 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.

[32] 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, New York, NY, USA, 2015. ACM.

[33] Nan Du, Le Song, Manuel Gomez-Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3147–3155, USA, 2013. Curran Associates Inc.

[34] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. Time-sensitive recommendation from recurrent user activities. In *NIPS*, 2015.

[35] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *CoRR*, abs/1706.02633, 2018.

[36] Mehrdad Farajtabar, Nan Du, Manuel Gomez-Rodriguez, Isabel Valera, Hongyuan Zha, and Le Song. Shaping social activity by incentivizing users. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2474–2482, Cambridge, MA, USA, 2014. MIT Press.

[37] Jacob Fauber and Christian R. Shelton. Modeling "presentness" of electronic health record data to improve patient state estimation. In *MLHC*, 2018.

[38] C. Finlay, J. Jacobsen, L. Nurbekyan, and Adam M. Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. *arXiv: Machine Learning*, 2020.

[39] Susannah Fleming, Matthew Thompson, Richard Stevens, Carl Heneghan, Annette Plüddemann, Ian Maconochie, Lionel Tarassenko, and David Mant. Normal ranges of heart rate and respiratory rate in children from birth to 18 years of age: a systematic review of observational studies. *The Lancet*, 377(9770):1011–1018, 2011.

[40] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *ArXiv*, abs/1704.02798, 2017.

[41] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, RECOMB '00, pages 127–135, New York, NY, USA, 2000. ACM.

[42] Joseph Futoma, Sanjay Hariharan, Katherine A. Heller, Mark Sendak, Nathan Brajer, Meredith Clement, Armando Bedoya, and Cara O'Brien. An improved multi-output Gaussian process rnn with real-time validation for early sepsis detection. In *MLHC*, 2017.

[43] Marzyeh Ghassemi, Tristan Naumann, Finale Doshi-Velez, Nicole Brimmer, Rohit Joshi, Anna Rumshisky, and Peter Szolovits. Unfolding physiological state: Mortality modelling in intensive care units. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 75–84. ACM, 2014.

[44] Marzyeh Ghassemi, Marco AF Pimentel, Tristan Naumann, Thomas Brennan, David A Clifton, Peter Szolovits, and Mengling Feng. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in ICU with sparse, heterogeneous clinical data. In *AAAI*, pages 446–453, 2015.

[45] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML*, 2011.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[47] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[48] Peter John Green and David I. Hastie. Chapter 1 Reversible jump MCMC. 2009.

[49] Tom Griffiths. Gibbs sampling in the generative model of latent Dirichlet allocation, 2002.

[50] 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.

[51] M. Habiba and Barak A. Pearlmutter. Neural ODEs for informative missingess in multivariate time series. *2020 31st Irish Signals and Systems Conference (ISSC)*, pages 1–6, 2020.

[52] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*, 2017.

[53] Alan Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58:83, 04 1971.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[55] Xinran He, Theodoros I. Rekatsinas, James R. Foulds, Lise Getoor, and Yan Liu. Hawkestopic: A joint model for network inference and topic modeling from text-based cascades. In *ICML*, 2015.

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

[57] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[58] Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and their Applications*, 8(3):335 – 347, 1979.

[59] Kazi T. Islam, Christian R. Shelton, Juan I. Casse, and Randall C. Wetzel. Marked point process for severity of illness assessment. In *MLHC*, 2017.

[60] Yacine Jernite, Yoni Halpern, Steven Horng, and David Sontag. Predicting chief complaints at triage time in the emergency department. In *NIPS 2013 Workshop on Machine Learning for Clinical Data Analysis and Healthcare*, 2013.

[61] Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations. *CoRR*, abs/1905.10403, 2019.

[62] How Jing and Alex Smola. Neural survival recommender. In *WSDM '17*, 2017.

[63] Yohan Jo, Natasha Loghmanpour, and Carolyn Penstein Rosé. Time series analysis of nursing notes for mortality prediction via a state transition topic model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1171–1180, New York, NY, USA, 2015. ACM.

[64] Alistair EW Johnson, Andrew A Kramer, and Gari D Clifford. A new severity of illness scale using a subset of acute physiology and chronic health evaluation data elements shows comparable predictive accuracy. *Critical care medicine*, 41(7):1711–1718, 2013.

[65] Alistair EW 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, 2016.

[66] Rohit Joshi and Peter Szolovits. Prognostic physiology: modeling patient severity in intensive care units using radial domain folding. In *AMIA Annual Symposium Proceedings*, volume 2012, page 1276. American Medical Informatics Association, 2012.

[67] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[68] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (SAPS II) based on a european/north american multicenter study. *Jama*, 270(24):2957–2963, 1993.

[69] Jean-Roger Le Gall, Philippe Loirat, Annick Alperovitch, Paul Glaser, Claude Granthil, Daniel Mathieu, Philippe Mercier, Remi Thomas, and Daniel Villers. A simplified acute physiology score for ICU patients. *Critical care medicine*, 12(11):975–977, 1984.

[70] Li-Wei H Lehman, Mohammed Saeed, William J Long, Joon Lee, and Roger G Mark. Risk stratification of ICU patients using topic models inferred from unstructured progress notes. In *AMIA*. Citeseer, 2012.

[71] Ted Petrie Leonard E. Baum. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[72] Erik Lewis and George O. Mohler. Research article a nonparametric em algorithm for multiscale Hawkes processes. 2011.

[73] Shuang Li, Shuai Xiao, Shixiang Zhu, Nan Du, Yao Xie, and Le Song. Learning temporal point processes via reinforcement learning. In *Proceedings of the 32nd Advances in Neural Information Processing Systems*, NIPS'18, pages 10804–10814, USA, 2018. Curran Associates Inc.

[74] Steven Cheng-Xian Li and Benjamin Marlin. Classification of sparse and irregularly sampled time series with mixtures of expected Gaussian kernels and random features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, pages 484–493, Arlington, Virginia, United States, 2015. AUAI Press.

[75] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with LSTM recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[76] Zachary C Lipton, David C Kale, and Randall Wetzel. Modeling missing data in clinical time series with rnns. *Machine Learning for Healthcare*, 2016.

[77] Zengjian Liu, Buzhou Tang, Xiaolong Wang, and Qingcai Chen. De-identification of clinical notes via recurrent neural network and conditional random field. *Journal of biomedical informatics*, 75S:S34–S42, 2017.

[78] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1903–1911, New York, NY, USA, 2017. ACM.

[79] Benjamin M. Marlin, David C. Kale, Robinder G. Khemani, and Randall C. Wetzel. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2Nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, pages 389–398, New York, NY, USA, 2012. ACM.

[80] David Marsan and Olivier Lengliné. Extending earthquakes' reach through cascading. *Science*, 319(5866):1076–1079, 2008.

[81] Nazanin Mehrasa, Akash Abdu Jyothi, T. Durand, Jiawei He, L. Sigal, and G. Mori. A variational auto-encoder model for stochastic point processes. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3160–3169, 2019.

[82] Hongyuan Mei and Jason Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6757–6767, USA, 2017. Curran Associates Inc.

[83] Qiaozhu Mei, Kristina Klinkner, Ravi Kumar, and Andrew Tomkins. An analysis framework for search sequences. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1991–1994, New York, NY, USA, 2009. ACM.

[84] Riccardo Miotto, Li Li, Brian A. Kidd, and Joel T Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. In *Scientific reports*, 2016.

[85] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, Nov 1996.

[86] Berndt Muller, Joachim Reinhardt, and Michael T. Strickland. *Stochastic Neurons*, pages 38–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

[87] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *ICML*, 2000.

[88] P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh. `Deepr`: A convolutional net for medical records. *IEEE Journal of Biomedical and Health Informatics*, 21(1):22–30, Jan 2017.

[89] Uri Nodelman, Christian R. Shelton, and Daphne Koller. Continuous time Bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pages 378–387, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[90] Uri Nodelman, Christian R. Shelton, and Daphne Koller. Expectation maximization and complex duration distributions for continuous time Bayesian networks. *CoRR*, abs/1207.1402, 2012.

[91] Yosihiko Ogata. Space-time point-process models for earthquake occurrences. *Annals of the Institute of Statistical Mathematics*, 50:379–402, 06 1998.

[92] Adam Oliner and Jon Stearley. What supercomputers say-an analysis of five system logs. In *IEEE/IFIP Conf. Dep. Sys. Net*, 2007.

[93] T. Ozaki. Maximum likelihood estimation of Hawkes' self-exciting point processes. *Annals of the Institute of Statistical Mathematics*, 31(1):145–155, Dec 1979.

[94] Ankur P Parikh, Asela Gunawardana, and Christopher Meek. Conjoint modeling of temporal dependencies in event streams. In *UAI Bayesian Modelling Applications Workshop*. Citeseer, 2012.

[95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[96] Robin Pemantle. A survey of random processes with reinforcement. *Probability Surveys*, 4(0):1–79, 2007.

[97] Xuan-Hieu Phan and Cam-Tu Nguyen. Gibbslda++: A c/c++ implementation of latent Dirichlet allocation (LDA). *Tech. rep.*, 2007.

[98] Murray M Pollack, Kantilal M Patel, and Urs E Ruttimann. PRISM III: an updated pediatric risk of mortality score. *Critical care medicine*, 24(5):743–752, 1996.

[99] Lev Semenovich Pontryagin, V G Boltyanskii, R V Gamkrelidze, and E F Mishchenko. *The mathematical theory of optimal processes*. Wiley, New York, NY, 1962.

[100] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[101] Shyamsundar Rajaram, Thore Graepel, and Ralf Herbrich. Poisson-networks: A model for structured point processes. In *Proceedings of the 10th international workshop on artificial intelligence and statistics*, pages 277–284. Citeseer, 2005.

[102] Rajesh Ranganath, Adler J. Perotte, Noémie Elhadad, and David M. Blei. Deep survival analysis. In *MLHC*, 2016.

[103] Rajesh Ranganath, Linpeng Tang, Laurent Charlin, and David M. Blei. Deep exponential families. *ArXiv*, abs/1411.2581, 2014.

[104] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[105] Narges Razavian and David Sontag. Temporal convolutional neural networks for diagnosis from lab tests. *CoRR*, abs/1511.07938, 2015.

[106] H. Risken and H. Haken. *The Fokker-Planck Equation: Methods of Solution and Applications Second Edition*. Springer, 1989.

[107] Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent odes for irregularly-sampled time series. *CoRR*, abs/1907.03907, 2019.

[108] Izhak Rubin. Regular point processes and their detection. *IEEE Trans. Information Theory*, 18:547–557, 1972.

[109] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care II (MIMIC-II): a public-access intensive care unit database. *Critical care medicine*, 39(5):952, 2011.

[110] Suchi Saria, Daphne Koller, and Anna Penn. Learning individual and population level traits from clinical temporal data. In *Proc. Neural Information Processing Systems (NIPS), Predictive Models in Personalized Medicine workshop*. Citeseer, 2010.

[111] M. B. Short, G. O. Mohler, P. J. Brantingham, and G. E. Tita. Gang rivalry dynamics via coupled point process networks. *Discrete and Continuous Dynamical Systems Series B*, 19(5):1459–1477, 2014.

[112] Satya Narayan Shukla and Benjamin M. Marlin. Interpolation-prediction networks for irregularly sampled time series. *ICLR*, 2019.

[113] Aleksandr Simma, Moisés Goldszmidt, John MacCormick, Paul Barham, Richard Black, Rebecca Isaacs, and Richard Mortier. CT-NOR: representing and reasoning about events in continuous time. *CoRR*, abs/1206.3280, 2012.

[114] Aleksandr Simma and Michael I. Jordan. Modeling events with cascades of Poisson processes. *CoRR*, abs/1203.3516, 2012.

[115] Anthony Slater, Frank Shann, and Gale Pearson. PIM2: a revised version of the paediatric index of mortality. *Intensive Care Medicine*, 29(2):278–285, 2003.

[116] Huan Song, Deepta Rajan, Jayaraman J. Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *AAAI*, 2018.

[117] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[118] Harini Suresh, Nathan Hunt, Alistair Edward William Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical intervention prediction and understanding using deep networks. *CoRR*, abs/1705.08498, 2017.

[119] Qingxiong Tan, Mang Ye, Baoyao Yang, S. Liu, A. J. Ma, T. Yip, G. Wong, and Pongchi Yuen. Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. In *AAAI*, 2020.

[120] Wilson Truccolo, Uri T Eden, Matthew R Fellows, John P Donoghue, and Emery N Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005.

[121] Utkarsh Upadhyay, Abir De, and Manuel Gomez-Rodrizuez. Deep reinforcement learning of marked temporal point processes. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 3172–3182, USA, 2018. Curran Associates Inc.

[122] Bjørnar Vassøy, Massimiliano Ruocco, Eliezer de Souza da Silva, and Erlend Aune. Time is of the essence: A joint hierarchical rnn and point process model for time and item predictions. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019.

[123] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[124] Alejandro Veen and Frederic P Schoenberg. Estimation of space-time branching process models in seismology using an em-type algorithm. *Journal of the American Statistical Association*, 103(482):614–624, 2008.

[125] Yongqing Wang, Huawei Shen, Shenghua Liu, Jinhua Gao, and Xueqi Cheng. Cascade dynamics modeling with attention-based recurrent neural network. In *IJCAI*, 2017.

[126] Jeremy C. Weiss, Sriraam Natarajan, and David Page. Multiplicative forests for continuous-time processes. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 458–466, USA, 2012. Curran Associates Inc.

[127] Jeremy C Weiss and David Page. Forest-based point process for event prediction from electronic health records. In *Machine learning and knowledge discovery in databases*, pages 547–562. Springer, 2013.

[128] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256, 1992.

[129] Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. In *JAMIA*, 2018.

[130] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3250–3259, USA, 2017. Curran Associates Inc.

[131] Shuai Xiao, Hongteng Xu, Junchi Yan, Mehrdad Farajtabar, Xiaokang Yang, Le Song, and Hongyuan Zha. Learning conditional generative models for temporal point processes. In *AAAI*, 2018.

[132] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M. Chu. Modeling the intensity function of point process via recurrent neural networkss. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 1597–1603. AAAI Press, 2017.

[133] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. Learning Granger causality for Hawkes processes. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1717–1726, New York, New York, USA, 20–22 Jun 2016. PMLR.

[134] Qiang Zhang, Aldo Lipani, Ömer Kirnap, and E. Yilmaz. Self-attentive Hawkes processes. *ArXiv*, abs/1907.07561, 2019.

[135] Kaiping Zheng, Wei Wang, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, and James Wei Luen Yip. Capturing feature-level irregularity in disease progression modeling. In *CIKM*, 2017.

[136] Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. In *AISTATS*, 2013.

[137] Ke Zhou, Hongyuan Zha, and Le Song. Learning triggering kernels for multi-dimensional Hawkes processes. In *ICML*, 2013.

[138] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and H. Zha. Transformer Hawkes process. *ArXiv*, abs/2002.09291, 2020.