MODELING NON-STATIONARY DYNAMICS OF SPATIO-TEMPORAL SEQUENCES WITH SELF-ORGANIZING POINT PROCESS MODELS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Oguzhan Karaahmetoglu

June, 2021

Modeling Non-stationary Dynamics of Spatio-temporal Sequences with Self-Organizing Point Process Models
By Oguzhan Karaahmetoglu
June, 2021

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Süleyman Serdar Kozat(Advisor)

Sinan Gezici

Çağatay Candan

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

MODELING NON-STATIONARY DYNAMICS OF SPATIO-TEMPORAL SEQUENCES WITH SELF-ORGANIZING POINT PROCESS MODELS

Oguzhan Karaahmetoglu
M.S. in Electrical and Electronics Engineering
Advisor: Süleyman Serdar Kozat
June, 2021

We investigate the challenging problem of modeling the non-stationary dynamics of spatio-temporal sequences for prediction applications. Spatio-temporal sequence modeling has critical real-life applications such as natural disaster, social, and criminal event prediction. Even though this problem has been thoroughly studied, many approaches do not address the non-stationarity and sparsity of the spatio-temporal sequences, which are frequently observed in real-life sequences. Here, we introduce a novel prediction algorithm that is capable of modeling non-stationarity in both time and space. Moreover, our algorithm can model both densely and sparsely populated sequences. We partition the spatial region with a decision tree, where each node of the tree corresponds to a subregion. We model the event occurrences in different subregions in space with individual but interacting point processes. Our algorithm can jointly optimize the partitioning tree and the interacting point processes through a gradient-based optimization. We compare our approach with statistical models, probabilistic approaches, and deep learning based approaches, and show that our model achieves the best forecasting performance on real-life datasets such as earthquake and criminal event records.

Keywords: Spatiotemporal Modeling, Non-stationary Sequence, Time-Series Forecasting, Point Processes, Decision Trees.

ÖZET

KENDİNİ DÜZENLEYEN NOKTASAL SÜREÇ MODELLERİ İLE UZAY-ZAMANSAL DİZİLERİN DURAĞAN OLMAYAN DİNAMİKLERİNİ MODELLEME

Oguzhan Karaahmetoglu
Elektrik ve Elektronik Muhendisligi, Yüksek Lisans
Tez Danışmanı: Süleyman Serdar Kozat
June, 2021

Uzay-zamansal tahmin uygulamaları için serilerin değişken dinamiklerini modelleme problemini çalışmaktayız. Uzay-zamansal serilerin modellenmesi, doğal afet, sosyal medya ve suç olaylarını tahmin etme gibi birçok önemli uygulamada kullanılmaktadır. Bu problem bir çok araştırmaya konu olsa da birçok çalışma değişken dinamikler ve seyreklik gibi gerçek veriler üzerinde gözlemlenen durumları incelememektedir. Uzayda ve zamanda değişebilen dağılımları modelleyen yeni bir algoritma sunmaktayız. Algoritmamız verinin seyrekliğinden bağımsız olarak çalışabilmekte ve doluluk oranı değişebilen serilerde çalışabilmektedir. Uzaysal düzlemi bir karar ağacı ile, her düğüm bir bölgeye karşı gelecek şekilde, bölmekteyiz. Uzaysal düzlemdeki farklı yerlerdeki olay dağılımlarını bireysel fakat etkileşimli noktasal süreç modelleri ile modellemekteyiz. Algoritmamız hem karar ağaçlarını hem de noktasal süreç parametrelerini birlikte gradyan tabanlı bir süreç ile optimize etmektedir. Yaklaşımımızı istatistiksel, olasılıksal ve derin öğrenme tabanlı modeller ile karşılaştırmaktayız, ve deprem ve suç olayları kayıtları gibi gerçek veriler üzerinde en iyi tahmin performansını elde ettiğini göstermekteyiz.

Anahtar sözcükler: Uzay-Zamansal Modelleme, Değişken Dinamikli Seriler, Zamansal Tahmin, Noktasal Süreçler, Karar Ağaçları.

Acknowledgement

I would like to thank Prof. Süleyman Serdar Kozat for his wise supervision during my M.S. studies and the opportunity to work on real-life challenges. Thanks to his helpful guidance, I had the chance to develop real-life prediction and analysis tools using state-of-the-art technologies. I have had a fruitful education as I have been able to author studies for highly respected journals and conferences while working as a full-time Machine Learning Engineer. To many more scientific endeavors alike during my doctorate years.

I would like thank Prof. Sinan Gezici and Prof. Çağatay Candan as my examining committee members.

I would like to express my deepest gratitude to my family, who have always supported and encouraged me throughout my journey. I owe my passion and determination to my mother. Although I will be physically further away from them during my Ph.D., I know that they will continue supporting me for the rest of my journey.

I have been extremely fortunate to meet Alper Akyüz during my high school years. I wish him happiness and luck in his future business plans. I also feel lucky to meet Muhammet Furkan Kılıç, Kadir Bulut Özler and Şeref Kutay Yakut during my undergraduate years. We have been through the most challenging experiences together. I wish Kutay luck in his upcoming Ph.D. application.

I would like to give my best regards to my G, Deniz Umut Yıldırım (Doenisz). I believe our friendship has been, and will be, mutually beneficial as we have valuable and enlightening conversations. I am especially grateful for his unending help through my undergraduate and M.S. years. I sincerely believe that he will achieve his academic and business goals.

I also would like to thank Doğan Can Çiçek, Ahmet Ercem Bulut, Ertuğ Ufuk and Yunus Emre Özertaş. I have really enjoyed the memorable experiences we had and the opportunity to work closely with you. I appreciate your support and

friendship that made Databoss a better workplace for me. I hope our paths will cross in the future once again. I wish Doğan Can Çiçek and Selim Furkan Tekin much success in their upcoming Ph.D. applications. I would like to thank Emir Ceyani for his aid through my thesis stage.

Last but not least, I would like to wish success and luck to my brother, Gökhan. I hope you will preserve your determination to learn and read as you will soon graduate and become a Computer Engineer. Never forget that your courage will always be rewarded.

Contents

1 Introduction **1**

1.1 Preliminaries 1

1.2 Prior Art and Comparisons 3

1.3 Contributions 4

1.4 Thesis Outline 5

2 Problem Description **6**

2.1 Problem Formulation 6

2.2 Representation of Sample Observations with Point Processes . . . 8

3 Novel Spatio-temporal Point Process Formulation For Modeling Non-stationary Dynamics **11**

3.1 Spatial Partitioning Mechanism With A Tree-Based Structure . . 12

3.2 Adaptive Partitioning of the Spatial Region with Decision Trees . 14

3.3 Sample Time and Location Prediction 17

3.4 Model Parameter Optimization via Likelihood Maximization . . . 20

3.5 Online Model Optimization 24

4 Experiments and Simulations **26**

4.1 Real-life Datasets For Spatio-Temporal Prediction 26

4.2 Benchmark Models For Performance Comparison 28

4.2.1 Event Count Estimation Models 28

4.2.2 Event Time and Location Prediction Models 29

4.2.3 Hybrid Approaches 30

4.3 Evaluation of the Presented Algorithm 31

4.3.1 Prediction Horizon Effect 31

4.3.2 Past Temporal Window Length Effect 33

4.3.3 Parallel and Separate Training 35

4.3.4 Spatial Subregion Number Effect 35

4.4 Performance Comparison With Benchmark Models 40

5 Conclusion **44**

List of Figures

2.1 Illustration of a spatio-temporal sequence with 4 samples, $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4\}$. Each sample has a location stamp $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$ and a timestamp $\{t_1, t_2, t_3, t_4\}$. . 7

3.1 Diagram of a 2-level tree, which has three nodes m_1, m_2, m_3. At every node, a decision function is computed, which are shown as m_{1d}, m_{2d}, m_{3d}. Subregion scores $\boldsymbol{\rho}(\mathbf{l}_i)$ are computed for 4 branch leaves for the input samples $\mathbf{n}_i = [t_i, \mathbf{l}_i]$. Each leaf node is associated with a spatial subregion, which are L_1, L_2, L_3 and L_4 for the 4 branches. 15

4.1 Prediction horizon performance comparison for our algorithm on two datasets. 32

4.2 Past temporal window length comparison for different horizon lengths on two datasets. 34

4.3 Parallel and Separate training comparison on two different datasets. 36

4.4 Performance comparison for different number of spatial subregions on the crime and earthquake datasets. 37

4.5 Performance comparison in both time and location estimations on the earthquake dataset. 41

List of Tables

4.1 Experiment results on crime and simulated datasets for different horizon lengths. 39

4.2 RMSE Test results on the Chicago Crime Dataset 43

4.3 RMSE Test results on the Earthquake Dataset 43

4.4 Test results for the time and location estimations on the earthquake dataset. 43

List of Abbreviations

RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
LSTM	Long-Short Term Memory
CNN	Convolutional Neural Network
MPP	Marked Point Process
MCI	Monte Carlo Integration
ADAM	Adaptive Moment Estimation
NLL	Negative Log-Likelihood
NEIC	National Earthquake Information Center
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
RSTPP	Recurrent Spatio-temporal Point Process
RMTPP	Recurrent Marked Temporal Point Process
MSE-t	Mean Squared Error in Time
MSE-l	Mean Squared Error in Location

List of Symbols and Notation

x	Scalar value representation
$\mathbf{x}$	Vector representation
$\mathbb{1}_N$	Column vector of ones with length N
$\mathbf{X}$	Matrix Representation
$\tilde{t}_i$	Random variable mapping the next event time to probabilities.
$\tilde{\mathbf{l}}_i$	Random variable mapping the next event location to probabilities.
$f_{\tilde{t}_i}$	Probability Density function of the next event time
$f_{\tilde{\mathbf{l}}_i}$	Probability Density function of the next event location

Chapter 1

Introduction

1.1 Preliminaries

Effective processing of spatio-temporal data carries vital importance due to a wide range of problem setups for applications such as sequence prediction, dynamic system modeling, and data assimilation [1], [2], [3]. These setups are frequently studied due to their critical applications like criminal and social event prediction, and predictive maintenance [2], [4], [5], [6]. Spatio-temporal sequences consist of samples ordered by their occurrence time and tagged along with their exact locations in a 2-D space. The aim is to predict the number of events that will take place in future spatio-temporal intervals or estimate the precise location and time of the next event using the past event times and positions up to an observation point. Accurately forecasting the distribution of criminal events or precisely estimating the time and location of the next event could save many lives and expenses [7], [3], [4]. However, certain difficulties such as the non-stationary dynamics or sparse distribution of the events restrict the application of standard approaches to this problem directly [8], [9]. Thus, we present a point-process-based approach to address these issues. Our algorithm can partition the data into subregions, where non-stationary dynamics of event sequences are modeled in each subregion with individual but interacting processes.

Deep learning-based forecasting approaches have been shown to be superior compared to other methods. Particularly, Recurrent Neural Network (RNN) achieved exceptional performances in many domains [8], [9], [10]. These significant improvements are due to the inherent memory structure in RNN models [11]. RNN also has two other variants, namely, Gated Recurrent Unit (GRU) and Long-Short Term Memory (LSTM) [12]. These variants improve the state transition mechanism of the standard RNN model by introducing gate mechanisms that prevent exploding or vanishing gradients problems [12]. Convolutional Neural Networks (CNN) are also applied to the same problem setup [13], [14], [15]. CNNs are specifically tailored to capture complex spatial patterns in data, which plays a crucial role in predicting the number of future events [16], [10]. Despite the significant performance improvements, deep architectures can only process structured data, i.e., fixed sampling intervals and discretized spatial locations. Therefore, additional pre-processing and post-processing stages are added to process data [8], [17], [16].

Another line of studies focus on applying the point processes and statistical time-series models to the same setup [8], [9], [17]. These approaches aim to model data directly, and they are specifically designed for the application domain [18], such as the Hawkes process with an intensity that is a function of past event times. Despite their success in modeling events, standard point process intensities are formulated only for the temporal domain [18], [19]. Certain studies introduce spatio-temporal point processes by extending the formulation for the spatial domain [8]. Thus, allowing the estimation of the location of an event along with its occurrence time. Moreover, certain approaches present deep learning-based formulations for the intensity function [8], [9].

Here, we introduce a novel spatio-temporal prediction algorithm. We model the data as a non-stationary sequence in both time and space, as is the case in many real-life applications [17], [20]. We do not assume any prior information about the sparsity of the data, which makes our approach applicable to different domains. Our method is based on point-processes, where we extend the formulation for the spatio-temporal sequences. We adaptively partition the space into

subregions and model sequences in each subregion with interacting processes. Although we focus on the intensity function of the Hawkes process, our formulation can be readily extended to different intensity formulations. We show that our algorithm can model event sequences in real-life data such as crime and earthquake datasets. Finally, we provide a gradient-based optimization procedure for likelihood-maximization. Through this optimization procedure, we jointly optimize the partition boundaries and interacting point processes.

1.2 Prior Art and Comparisons

Deep neural networks have been applied to many real-life prediction problems due to their abilities to model complex patterns [11], [16], [21]. Particularly, RNNs demonstrate exceptional performance in modeling temporal patterns due to their inherent memory [11]. Although deep models can model nonlinear patterns, their performance depends on the availability of massive amounts of labeled data. Moreover, their time-invariant formulation limits their capability under non-stationary sequences [8], [22]. The samples in spatio-temporal data can be unevenly spaced in time and continuously distributed in space, whereas deep models process structured data [10], [8]. Additional pre-processing and post-processing stages may be designed for such cases [16], [23]. However, these stages change the problem description as the continuous nature of the event times and locations are changed. Furthermore, model estimations will also be generated for the fixed spatio-temporal bins [23]. Unlike this approach, we directly model the continuous data without applying any ad-hoc pre-processing stage.

Point processes are also applied to the same problem [17], [22], [24]. However, standard point processes are developed for modeling event occurrence times only, which prevents their direct application on spatio-temporal sequences [25], [26]. To mitigate this problem, marked point processes (MPP) were developed. MPPs estimate additional tags such as the location or type of the event along with its time [27]. Although this method has shown outstanding performance in modeling real-life trajectory sequences, it independently estimates the temporal and spatial

locations. There are also studies that combine deep models with point processes, as in [17], [8]. These methods can capture nonlinear temporal patterns due to the capabilities of deep recurrent models. However, they incorporate auxiliary information including the location of events to the intensity function additively, which limits the generalization potential of the formulation.

Here, we address these issues by formulating a point-process intensity that is a function of both time and space. Instead of incorporating location and time information additively, we use a temporal and a spatial kernel mechanism along with an interaction mechanism to model dependencies in adaptive spatial subregions. Finally, we give a gradient-based likelihood-maximization procedure that jointly optimizes all model parameters.

1.3 Contributions

Our main contributions are:

1. As the first time in the literature, we present a novel algorithm that adaptively partitions the spatial region into subregions and model the interaction between these subregions jointly.

2. Although our formulation focuses on the self-exciting Hawkes process, our approach is generic so that any other point process can be used depending on the application as provided remarks in the thesis.

3. We provide a gradient-based optimization procedure for parameter inference. We use a log-likelihood-based objective function, which can be optimized sequentially in both online and batch setups.

4. Through an extensive set of experiments on both simulated and real-life data, we show that our model can represent a spatio-temporal data such as earthquake and crime data, which are highly non-stationary. We compare

our approach with the standard well-known methods where we demonstrate significant performance improvements.

1.4 Thesis Outline

In the following section, we introduce our problem description. In section III-A, we briefly describe the probabilistic models that we use. In section III-B, we describe the adaptive partitioning of the spatial region and the spatial kernel mechanism used with the standard Hawkes process. In section III-C, we explain the procedure for estimating the times and locations of the samples using our model. In section III-D, we introduce the training algorithm for optimizing the model parameters and the objective function of our algorithm. In section IV, we present the experiment results on both simulated and real-life data. Finally, in section V, we give the concluding remarks.

Chapter 2

Problem Description

In this chapter, we give the mathematical formulation of our problem setup along with the preliminary methods that will provide the necessary background for subsequent chapters. We first describe our notation and the mathematical setup of our problem description. We also review the related methods that are applied on the same problem to support our design, which we will explain the later chapters.

2.1 Problem Formulation

We denote the matrices with boldface and uppercase letters, e.g. $\mathbf{X}$. $\mathbf{X}_{i,j}$ refers to the element of the matrix at the ith row and the jth column. $\mathbf{X}_{i,:}$ is the ith row and $\mathbf{X}_{:,j}$ is the jth column of the matrix $\mathbf{X}$. We denote the vectors with boldface lowercase letters, e.g. $\mathbf{x} = [x_0, x_1, ..., x_N]$ is a vector with length N. The notation $\mathbf{x}^T$ refers to the ordinary transpose of a vector and ℓ^2 norm of the vector $\mathbf{x}$ is $||\mathbf{x}||^2 = \langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{x}^T\mathbf{x}$. $\odot$ operator is the element-wise multiplication and $\mathbb{1}_N$ is a column vector of ones with length N. All vectors are column vectors and real vectors.

A spatio-temporal sequence consists of samples that are distributed along three

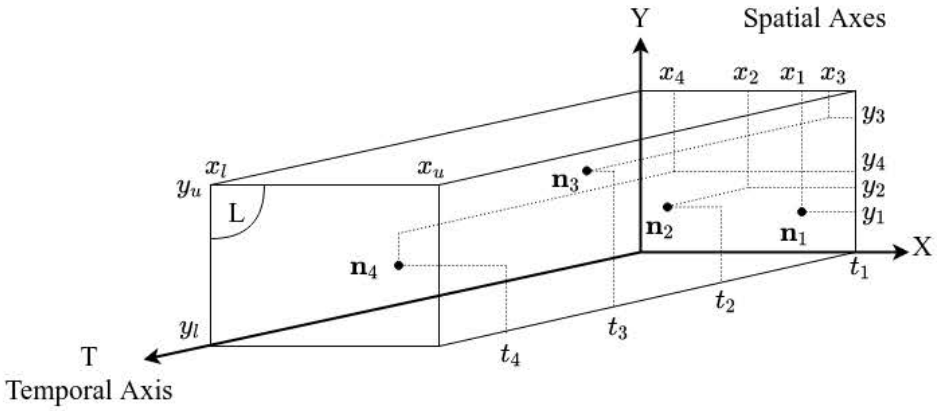


Figure 2.1: Illustration of a spatio-temporal sequence with 4 samples, $\{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4\}$. Each sample has a location stamp $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$ and a timestamp $\{t_1, t_2, t_3, t_4\}$.

axes: a temporal axis T and two spatial axes X and Y. An example sequence of samples $\mathbf{N} = \{\mathbf{n}_i\}_i$ are shown in Fig. 2.1. Each sample $\mathbf{n}_i = [t_i, \mathbf{l}_i]$ corresponds to an event with its respective occurrence time t_i and its location $\mathbf{l}_i = (x_i, y_i)$, which is generally recorded as a latitude and longitude pair. All samples are observed in the spatial region L with boundaries $L = [[x_\mathrm{l}, x_\mathrm{u}], [y_\mathrm{l}, y_\mathrm{u}]]$ as shown in Fig. 2.1, i.e. $x_i \in [x_\mathrm{l}, x_\mathrm{u}]$, $y_i \in [y_\mathrm{l}, y_\mathrm{u}]$ $\forall \mathbf{n}_i$. We model the data as a continuous sequence in both time and space, therefore $t_i \in \mathbb{R}$ and $x, y \in \mathbb{R}$. Moreover, samples are ordered according to their occurrence times, i.e., $t_i > t_j$ for $i > j$.

We form the set $\{t_i\}_i$, which consists of the sample observation times. Using the observation times set, we define the history $\Omega(t)$ at time t, which is expressed as $\Omega(t) = \{\mathbf{n}_i | t_i < t\}_i$, i.e. the observed samples until time t. We aim to predict the observation time and location of future samples. We investigate this problem in two setups. In the first setup, we predict the number of events in a future spatio-temporal interval. This setup is preferable when the events are densely populated in small spatio-temporal intervals, such as the criminal activities in a city. We measure the performance for this setup via the squared error between the predicted number of events and the actual number of events in a fixed horizon. In the second setup, we predict the exact time and location of the next event $\mathbf{n}_i = [t_i, \mathbf{l}_i]$ using the history $\Omega(t_i)$. This setup is used for event sequences where occurrence times are distantly spaced in time, such as the earthquake events in

an area. We assess the performance of our estimation $\hat{\mathbf{n}}_i$ for the sample $\mathbf{n}_i$ with the loss function

$$l(\hat{\mathbf{n}}_i, \mathbf{n}_i) = [(\hat{t}_i - t_i)^2, (\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2]. \tag{2.1}$$

We model the sample observation times and locations with a point process model, therefore the number of events in an interval is computed by integrating the intensity function. The next event time and location are the expected time and location of the sample $\mathbf{n}_i$, i.e. $(\hat{t}_i,\ \hat{\mathbf{l}}_i) = \mathbb{E}[\tilde{t}_i,\ \tilde{\mathbf{l}}_i|\Omega(t_{i-1})]$. Variables $\tilde{t}_i$ and $\tilde{\mathbf{l}}_i$ are random variables, which are the observation time and spatial location of the sample $\mathbf{n}_i$. Therefore, we give brief information about the point processes in the next subsection.

2.2 Representation of Sample Observations with Point Processes

We define the function $N(t)$ as the total number of samples up to time t. For point processes, the density function of the observation time of the sample $\mathbf{n}_i$ is given as

$$f_{\tilde{t}_i}(t|\Omega(t)) = \lim_{\Delta t \to 0} \mathrm{P}(N(t) - N(t - \Delta t) = 1|\Omega(t)), \tag{2.2}$$

which is a function of previous samples [19]. We can also express the density function in (2.2) using the conditional intensity function

$$f_{\tilde{t}_i}(t|\Omega(t)) = \lambda(t|\Omega(t)) e^{-\int_{t_{i-1}}^{t} \lambda(t'|\Omega(t'))dt'}, \tag{2.3}$$

by approximating it with Bernoulli trials as $\Delta t \to 0$. Point generation mechanisms and the form of the probability density functions of a point process is

controlled by the choice of the intensity function $\lambda(t|\Omega(t))$ [19]. Definition of the intensity function is given as

$$\lambda(t|\Omega(t)) = \lim_{\Delta t \to 0} \frac{\mathrm{P}(N(t+\Delta t) - N(t) = 1|\Omega(t))}{\Delta t}, \tag{2.4}$$

which corresponds to the expected number of observations in an infinitesimal time interval around t. Thus, the expected number of events that will occur in a temporal interval is computed by integrating the intensity function over it.

Different choices of the conditional intensity function will result in different sample generation mechanisms as the magnitude of the intensity control the rate at which the samples are observed. Depending on the formulation, this rate can be constant, time-varying or space-varying [19]. We formulate our algorithm using the Hawkes process intensity, which is a function of the history as

$$\lambda(t|\Omega(t)) = \mu + \sum_{t_j < t} e^{-\gamma(t-t_j)}, \tag{2.5}$$

where $\{t_j\}_j$ are the times of the sample observations before t, μ is the background intensity and γ is called the decay rate.

Remark 1. *Although we give our formulations for the Hawkes process intensity, our approach can be used with other point process intensity formulations. Thus, depending on the application, instead of using the Hawkes process, we could use any other intensity formulation such as the Poisson intensity*

$$\lambda(t|\Omega(t)) = \lambda^*, \tag{2.6}$$

where $\lambda^* \in \mathbb{R}^+$ [18]. We could also use the intensity of the self-correcting process

$$\lambda(t|\Omega(t)) = \mu t - \sum_{t_j < t} \alpha, \tag{2.7}$$

$\mu, \alpha \in \mathbb{R}^+$ [28].

We choose the Hawkes process intensity formulation due to its certain properties. First of all, the dependency on the past samples yields a non-stationary process in time due to the temporal kernel $g_{\mathrm{t}}(t - t_j) = e^{-\gamma(t-t_j)}$. Moreover, the effect of the past samples in the intensity is additive and increases with closer sample times. Thus, it is a self-exciting process. This property is useful for modeling certain real-life applications [24], [23]. Nevertheless, spatial location information of the past samples are not incorporated into the formulation. We aim to predict the times and locations of the samples using the observation times and locations of the past samples. To this end, we introduce a conditional intensity function by incorporating the spatial interactions with a spatial kernel mechanism. In the following section, we give details about this spatial kernel.

In the following chapter, we will discuss a novel spatio-temporal sequence modeling approach based on point processes. The formulation of our approach is based on the spatio-temporal point process formulation, however, our method is able to model non-stationary dynamics in the sequences.

Chapter 3

Novel Spatio-temporal Point Process Formulation For Modeling Non-stationary Dynamics

In this section, we present a novel point process based probabilistic approach to model spatio-temporal sequences. Although our formulation is similar to the MPPs, we design a joint spatial and kernel mechanisms that transform the time and location of previous events in a way that the non-stationary dynamics of the spatial and temporal regions are efficiently modeled. Our approach is based on partitioning the spatial space in an adaptive manner and modeling interacting processes in each partition. Therefore, we first describe the partitioning formulation and then give the overall intensity formulation. Finally, we explain the gradient-based likelihood maximization procedure to find the model parameters.

3.1 Spatial Partitioning Mechanism With A Tree-Based Structure

Here, we give the formulation for the conditional intensity function that uses a temporal and spatial kernel to model the sample observations with a time and space-varying process. Our spatial kernel is based on decision trees, which forms adaptive spatial partitions and can be jointly optimized with the process parameters.

Hawkes process intensity is a time-varying function and can change over time based on the past sample observations. However, in spatio-temporal data, the intensity could change in space as well. Thus, the intensity can be expressed as a function of both time and space to model non-stationary real-life data. Consequently, we define the intensity in (2.4) as the expected number of sample observations in an infinitesimal spatio-temporal interval around time t and location $\mathbf{l}$, which is given as

$$\lambda(t, \mathbf{l}|\Omega(t)) = \lim_{\Delta t \to 0,\ \Delta l \to 0} \frac{\mathrm{P}(N(t + \Delta t, \bar{\mathbf{l}}) - N(t) = 1|\Omega(t))}{\Delta t \Delta l} \tag{3.1}$$

where $\bar{\mathbf{l}}$ is the infinitesimal circular region centered at the location $\mathbf{l}$ with radius Δl, i.e. $\bar{\mathbf{l}} = \{(x,\ y) \in \mathbb{R}^2|\ ||\mathbf{l} - (x,\ y)||^2 \leq \Delta l\}$. Consequently, we define $N(t, \bar{\mathbf{l}})$ as the number of events up to time t and the number of events around the spatial location $\mathbf{l}$ at time t.

Since we model the data with a time and space-varying process, we partition the spatial region L into fixed number of subregions, $\{L_k\}_{k=1}^K$. These subregions form a partition of the whole space, i.e. $\bigcup_{k=1}^K L_k = L$ and $L_i \cap L_j = \emptyset$. We group the samples into subsets depending on their spatial locations, i.e. $\mathbf{N}_k = \{\mathbf{n}_i|\mathbf{l}_i \in L_k\}_i$, which also partition the samples. We also define the vector $\boldsymbol{\rho}(\mathbf{l}) = [\rho_k(\mathbf{l})]_{k=1}^K$ $\forall \mathbf{l} \in L$, which is the subregion vector where

$$\rho_k(\mathbf{l}) = \begin{cases} 1, & \text{if } \mathbf{l} \in L_k \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

One-hot vector $\boldsymbol{\rho}(\mathbf{l})$ indicates the subregion containing the location $\mathbf{l}$. For each subregion, we represent the sample observations with an individual point process model, hence with an individual conditional intensity. For the subregion L_k, the conditional intensity is $\lambda_k(t, \Omega(t))$. As a result, we express the conditional intensity for any arbitrary location as $\lambda(t, \mathbf{l}|\Omega(t))$, which is the conditional intensity of the subregion L_k. It is given as,

$$\lambda(t, \mathbf{l}|\Omega(t)) = \begin{cases} \lambda_1(t|\Omega(t)), & \text{if } \mathbf{l} \in L_1 \\ \lambda_2(t|\Omega(t)), & \text{if } \mathbf{l} \in L_2 \\ ... & \\ \lambda_K(t|\Omega(t)), & \text{if } \mathbf{l} \in L_K. \end{cases} \tag{3.3}$$

We can also express the conditional intensity as

$$\lambda(t, \mathbf{l}|\Omega(t)) = \boldsymbol{\rho}(\mathbf{l})^T \boldsymbol{\lambda}(t, \mathbf{l}|\Omega(t)), \tag{3.4}$$

where $\boldsymbol{\lambda}(t, \mathbf{l}|\Omega(t)) = [\lambda_k(t, \mathbf{l}|\Omega(t))]_{k=1}^K$.

In order to incorporate the location information of the past samples, we also add a spatial kernel to the intensity $\lambda_k(t|\Omega(t))$ as

$$\lambda_k(t|\Omega(t)) = \mu_k + \sum_{t_j < t} g_{\mathrm{t},k}(t - t_j) g_{\mathrm{l},k}(\mathbf{l}_j), \tag{3.5}$$

where the temporal kernel $g_{\mathrm{t},k}(t - t_j)$ is selected as in (2.5) due to the self-exciting property. We formulate the spatial kernel $g_{\mathrm{l},k}(\mathbf{l}_j)$ in (3.5) as

$$g_{\mathrm{l},k}(\mathbf{l}_j) = \boldsymbol{\Gamma}_{:,k}^T \boldsymbol{\rho}(\mathbf{l}_j), \tag{3.6}$$

where $\mathbf{\Gamma}$ is a $K \times K$ matrix modeling the interaction among the samples in all subregions, which is referred to as the interaction matrix. The element $\mathbf{\Gamma}_{k,l}$ corresponds to the effect of a sample $\mathbf{n} \in \mathbf{N}_k$ in the subregion L_k to the intensity of the lth subregion.

Note that the summation in (3.5) accumulates the effects of all the past samples. In online setups, this summation would grow indefinitely, however, the exponential kernel in the temporal kernel allows us to truncate the summation by including only the ν most recent elements. To this end, we ignore the rest of the samples and define the set $\mathbf{N}_\Omega(t, \nu)$, which contains the most recent samples in the interval $[t - \nu, t]$.

We form the spatial subregions using decision trees that are adaptively organized in time. Thus, as the intensity functions are organized with new samples, boundaries of the spatial subregions are also organized. In the following subsection, we give details about the adaptive tree structure.

3.2 Adaptive Partitioning of the Spatial Region with Decision Trees

The decision tree in our algorithm consists of a collection of nodes $D = \{m_r\}_{r=1}^{R}$ that are placed hierarchically as in Fig. 3.1. Each node, except the leaf nodes (located at the bottom of the tree, at level $\ell = 2$), has two children that are linked with branches. The top node, m_1, is referred to as the root node.

The sample $\mathbf{n}_i$ is assigned to a spatial subregion using the decision tree. At each node, the sample is either assigned to the left or right node starting from the root node until reaching to a leaf node. A decision function, m_{r_d} with range $\{-1, +1\}$, is computed at each node, which uses the features of the input sample. In certain works, a randomly selected feature from the input sample is used at each node for comparison [29], [30], where the input has features $\mathbf{n}_i = [n_i^{(f)}]_{f=1}^{F}$. The decision function is given as

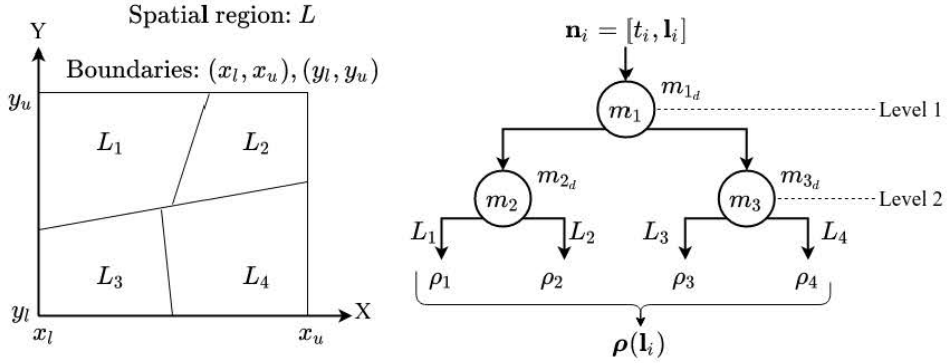


Figure 3.1: Diagram of a 2-level tree, which has three nodes $m_1,\ m_2,\ m_3$. At every node, a decision function is computed, which are shown as $m_{1d},\ m_{2d},\ m_{3d}$. Subregion scores $\boldsymbol{\rho}(\mathbf{l}_i)$ are computed for 4 branch leaves for the input samples $\mathbf{n}_i = [t_i, \mathbf{l}_i]$. Each leaf node is associated with a spatial subregion, which are L_1, L_2, L_3 and L_4 for the 4 branches.

$$m_{r\mathrm{d}}(\mathbf{n}_i) = \mathrm{sign}(n_i^{(m_{r\mathrm{f}})} - m_{r\mathrm{b}}) \lesseqgtr 0, \tag{3.7}$$

where $n_i^{(m_{r\mathrm{f}})}$ is the feature used at the node m_r and $m_{r\mathrm{b}}$ is the threshold for comparison. The sample $\mathbf{n}_i$ is assigned to the left node if (3.7) is positive and to the right node otherwise.

Samples are separated by their features with the decision functions as in (3.7). In our case, features of the samples are only the location vector $\mathbf{l} = (x_i, y_i)$. Therefore, we separate the samples by comparing their spatial components with threshold values. As the depth of the tree grows, the number of spatial subregions will also increase. For a depth-ℓ decision tree, we have $K = 2^\ell$ leaf branches for the subregions $\{L_k\}_{k=1}^K$ as in Fig. 3.1.

Instead of using a single feature for comparison as in (3.7), we use all features $[n_i^{(f)}]_{f=1}^F$ for comparison. Hence, the decisions have the form

$$m_{r\mathrm{d}}(\mathbf{n}_i) = \mathrm{sign}(m_{r\mathbf{w}}^T \mathbf{l}_i - m_{r\mathrm{b}}) \lesseqgtr 0, \tag{3.8}$$

where $m_{r\mathbf{w}}$ is the weight vector combining the location elements $x_i,\ y_i$. With

such decision functions, we partition the space with lines as in Fig. 3.1.

Remark 2. *We use a decision tree to separate the feature space, which results in linear decision boundaries. We could use any nonlinear function of the spatial coordinates, e.g.*

$$\boldsymbol{f}(x_i, y_i) = g(\boldsymbol{w}_x x_i + \boldsymbol{w}_y y_i + b), \tag{3.9}$$

where $\{\boldsymbol{w}_x, \boldsymbol{w}_y\}$ *are* $K \times 1$ *weight vectors and* g *is a nonlinear function. We can generate the vector* $\boldsymbol{\rho}(x_i, y_i)$ *by setting the maximum element of* $\boldsymbol{f}(x_i, y_i)$ *to 1 and the rest to 0, which generates a one-hot vector as in the decision tree partitioning. However, this formulation for the spatial separation could result in a complex partitioning [31].*

We optimize the boundaries of the spatial region by updating the weight parameters in (3.2) via a gradient-based procedure. Instead of using hard functions in the decision functions, we use the sigmoid function ($\sigma(x) = 1/(1 + e^{-x})$) as

$$m_{r\mathrm{d}}(\mathbf{n}_i) = \sigma(m_{r\mathbf{w}}^T \mathbf{n}_i - m_{r\mathrm{b}}), \tag{3.10}$$

which yields a score in $[0, 1]$. Soft functions in decision functions as in (3.10) were previously used in [32] for source coding. In the standard formulation given in (3.8), a sample is either assigned to the left branch or to the right branch. As a result, a sample can only lie in a single subregion, which was expressed with a one-hot vector as in (3.2). However, by using a soft function in the decision function, this hard separation is removed. In this case, the sample $\mathbf{n}_i$ is assigned to both the left branch and to the right branch with scores $m_{r\mathrm{d}}(\mathbf{n}_i)$ and $1 - m_{r\mathrm{d}}(\mathbf{n}_i)$.

In the standard formulation with hard decision functions, we represent the spatial subregion score vector with a one-hot vector. However, in this case, the subregion score vector is $\boldsymbol{\rho}(\mathbf{l}_i) = [\rho_k(\mathbf{l}_i)]_{k=1}^K$ where $\rho_k(\mathbf{l}_i) \in [0, 1]$ $\forall k$. We compute the scores by multiplying the scores obtained at each layer starting from the root node until the leaf branches as

$$\rho_k(\mathbf{n}_i) = \prod_{l=1}^{\ell} \pi(\mathbf{n}_i, \boldsymbol{\nu}_k^{(l)}, \boldsymbol{\tau}_k^{(l)}), \tag{3.11}$$

where

$$\pi(\mathbf{n}_i, \boldsymbol{\nu}_k^{(l)}, \boldsymbol{\tau}_k^{(l)}) = \begin{cases} {\nu_k^{(l)}}_{\mathrm{d}}(\mathbf{n}_i), & \tau_k^{(l)} = \text{left} \\ 1 - {\nu_{k^{(l)}}}_{\mathrm{d}}(\mathbf{n}_i), & \tau_k^{(l)} = \text{right}. \end{cases} \tag{3.12}$$

We define the sequence of branches and sequence of visited nodes to reach the kth leaf as $\boldsymbol{\tau}_k$ and $\boldsymbol{\nu}_k$, e.g. for the depth-2 tree shown in Fig. 3.1, second leaf branch ρ_2 is reached by taking first the left branch and then the right branch. Thus, $\boldsymbol{\tau}_2 = [\text{left, right}]$ and $\boldsymbol{\nu}_2 = [m_1, m_2]$. $\boldsymbol{\nu}_k^{(l)}$ in (3.11) and (3.12) is the visited node at level l to reach the kth leaf. $\boldsymbol{\tau}_k^{(l)}$ is the direction of the branch at level l to reach the kth leaf. The decision function score at level l is expressed as $\pi(\mathbf{n}_i, \boldsymbol{\nu}_k^{(l)}, \boldsymbol{\tau}_k^{(l)})$.

Remark 3. *The following properties are satisfied by the subregion score vector:*

- *The maximum subregion score is obtained for the subregion containing the spatial location, $\boldsymbol{l}_i$ of the sample $\boldsymbol{n}_i$, i.e. $\kappa = \arg\max_{k \in \{0,1,\ldots,K\}}[\rho_k(\boldsymbol{n}_i)] \Leftrightarrow \boldsymbol{l}_i \in L_\kappa$.*
- *Subregion scores sum up to 1 for any location $\boldsymbol{l}_i \in L$, i.e. $\sum_{k=1}^{K} \rho_k(\boldsymbol{l}_i) = 1$.*

The spatial subregion vector of scores $\boldsymbol{\rho}(\mathbf{n}_i)$ are computed by an adaptive decision tree, which is used in the spatial kernel mechanism in (3.6). In the next section, we explain the estimation method for the sample times and locations using the spatio-temporal density function. Moreover, we also explain the procedure for optimizing the model parameters.

3.3 Sample Time and Location Prediction

In order to estimate the time and location of samples, we formulate the joint density function $f_{\tilde{t}_i, \tilde{\mathbf{l}}_i}$, using the intensity (3.1) and the density in (2.3). The joint

density function is

$$
\begin{aligned}
f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t,\mathbf{l}|\Omega(t_{i-1})) &= \lambda(t,\mathbf{l}|\Omega(t_{i-1}))e^{-\Lambda_{t_{i-1}}(t,\mathbf{l})}, \\
\Lambda_{t_{i-1}}(t,\mathbf{l}) &= \int_{t_{i-1}}^{t}\int_{L}\lambda(t',\mathbf{l}'|\Omega(t_{i-1}))d\mathbf{l}'dt'.
\end{aligned}
\tag{3.13}
$$

We use a more compact expression for the conditional intensity as in (3.4), which corresponds to the weighted average of the subregion intensities $\boldsymbol{\lambda}(t,\mathbf{l}|\Omega(t_{i-1}))$. Therefore,

$$
\lambda(t,\mathbf{l}|\Omega(t_{i-1})) = \boldsymbol{\rho}(\mathbf{l})^T\boldsymbol{\lambda}(t,\mathbf{l}|\Omega(t_{i-1})). \tag{3.14}
$$

Note that in (3.14), the computations include an integration. We estimate the integration with the Riemann sum [33]. Thus, uniformly spaced samples $\{\mathbf{n}_j\}_j$ in time and space are sampled for the computation. For example, the exponent $\Lambda_{t_{i-1}}(t,\mathbf{l})$ is calculated as

$$
\Lambda_{t_{i-1}}(t,\mathbf{l}) = \frac{1}{M}\sum_{t_j,\mathbf{l}_j}\lambda(t_j,\mathbf{l}_j|\Omega(t_{i-1}))|T||L|, \tag{3.15}
$$

where M is the number of points used in the integration estimation. In our experiments, we have observed that choosing a sufficiently large number of points does not cause any negative effects on the performance. We sample the points $\mathbf{n}_j$ at times t_j and at locations $\mathbf{l}_j$. Finally, the terms $|T|$ and $|L|$ are the length and area of the temporal and spatial boundaries of the integration respectively. We compute $|T| = t - t_{i-1}$ and $|L| = (x_{\mathrm{u}} - x_{\mathrm{l}}) \times (y_{\mathrm{u}} - y_{\mathrm{l}})$.

Remark 4. *Instead of using the Riemann sum for the integration, we could use the Monte Carlo Integration (MCI) method [34], which is preferred over Riemann sum when the function has abrupt changes. To circumvent this issue, we can readily increase the number of sampled points in the Riemann sum given in* (3.15)*. In our experiments, we have observed that choosing a sufficiently large number of samples does not have a negative effect on the performance.*

To estimate the sample time and location, we marginalize the joint density function over time and space to obtain the marginal density functions as

$$\begin{aligned} f_{\tilde{t}_i}(t) &= \int_L f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t,\mathbf{l}|\Omega(t_{i-1}))d\mathbf{l}, \\ f_{\tilde{\mathbf{l}}_i}(\mathbf{l}) &= \int_{t_{i-1}}^{\infty} f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t,\mathbf{l}|\Omega(t_{i-1}))dt. \end{aligned} \tag{3.16}$$

We compute the estimated time and location of the sample $\mathbf{n}_i$ as the conditional mean of the random variables $\tilde{t}_i$ and $\tilde{\mathbf{l}}_i$ as

$$\hat{t}_i = \mathbb{E}[\tilde{t}_i|\Omega(t_{i-1})] = \int_{t_{i-1}}^{\infty} t' f_{\tilde{t}_i}(t'|\Omega(t_{i-1}))dt', \tag{3.17}$$

$$\hat{x}_i = \mathbb{E}[\tilde{x}_i|\Omega(t_{i-1})] = \int_L x' f_{\tilde{\mathbf{l}}_i}(m'|\Omega(t_{i-1}))dl, \tag{3.18}$$

$$\hat{y}_i = \mathbb{E}[\tilde{y}_i|\Omega(t_{i-1})] = \int_L y' f_{\tilde{\mathbf{l}}_i}(n'|\Omega(t_{i-1}))dl, \tag{3.19}$$

where the integration in (3.18) and (3.19) is over the region L.

The integration computations in (3.16) and (3.17) are over an infinitely long time interval. However, the conditional intensity function in (2.4) is, by definition, a non-negative function, and thus (3.13) is a monotonically decreasing function of time. Hence, we compute the Riemann sum of the integration up to a certain time. We experiment with different values for this parameter.

The integration computation in (3.13) is between the time of the last event and any time t. Computing this integration for two arbitrary times $t' > t'' > t_{i-1}$ from scratch would be unnecessary as they share a common interval. Therefore, we sort the sampled points in the Riemann sum estimation in (3.15). Starting from the closest point in time, we accumulate the intensity towards the furthest point and keep the accumulated intensity in memory.

3.4 Model Parameter Optimization via Likelihood Maximization

Here, we describe the objective function that we are using to update the model parameters and the gradient based optimization procedure.

The parameters of the node m_r are the weights $m_{r\mathbf{w}}$ and the threshold $m_{r\mathrm{b}}$. For a decision tree with R nodes, the model parameters are the collection of node parameters $\mathbf{\Theta}_{\text{tree}} = \{(m_{r\mathbf{w}}, m_{r\mathrm{b}})\}_{r=1}^{R}$. We can also group the point process parameters for K subregions. For a single subregion, the intensity in (3.5) has the parameters μ_k and γ_k. Interaction matrix $\mathbf{\Gamma}$ is a common parameter for all subregions. All parameters for all subregions are $\mathbf{\Theta}_{\text{hawkes}} = \{\mathbf{\Gamma},\ \{(\mu_k, \gamma_k)\}_{k=1}^{K}\}$.

Combining the two sets, we form the set of model parameters $\Theta = [\Theta_{\text{tree}}, \Theta_{\text{hawkes}}]$. Full notation for the density function $f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t, \mathbf{l}|\Omega(t_i))$ and $\lambda(t_j, \mathbf{l}_j|\Omega(t_j))$ are $f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t, \mathbf{l}|\Omega(t_i), \Theta)$ and $\lambda(t_j, \mathbf{l}_j|\Omega(t_i), \Theta)$ respectively. We drop the term Θ from the notation for simplicity.

We compute the likelihood of our model using

$$\mathcal{L}(\mathbf{N}) = \prod_{i=1}^{I} f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t_i, \mathbf{l}_i|\Omega(t_{i-1})). \tag{3.20}$$

We find the optimal set of model parameters $\mathbf{\Theta}^*$ by randomly initializing a set of model parameters $\mathbf{\Theta}_0$ and updating the set with the stochastic gradient-ascent algorithm [35]. To remove the exponential terms in (3.20), we compute the gradients with respect to the log-likelihood $\tilde{\mathcal{L}} = \log \mathcal{L}$, which is

$$\tilde{\mathcal{L}}(\mathbf{N}) = \sum_{i=1}^{I} \log f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t_i, \mathbf{l}_i|\Omega(t_{i-1})). \tag{3.21}$$

Using the definition of $f_{\tilde{t}_i,\tilde{\mathbf{l}}_i}(t_i, \mathbf{l}_i|\Omega(t_{i-1}))$ in (3.21) with the formulation in (3.13), we obtain

$$\tilde{\mathcal{L}}(\mathbf{N}) = \sum_{i=1}^{I} \log \lambda(t_i, \mathbf{l}_i | \Omega(t_{i-1})) - \sum_{j=1}^{J} \Lambda_{t_{i-1}}(t_j, \mathbf{l}_j), \tag{3.22}$$

which has two terms, $\mathcal{L}_{\text{positive}}$ and $\mathcal{L}_{\text{negative}}$. The integration in $\mathcal{L}_{\text{negative}}$ can be accumulated in time and expressed as a single integration as

$$\mathcal{L}_{\text{negative}} = \int_{t_0}^{T} \int_{L} \lambda(t', \mathbf{l}' | \Omega(t_{i-1})) d\mathbf{l}' dt', \tag{3.23}$$

where t_0 is the start of the integration simulation and T is the end time. We sample uniformly spaced points in both time and space for the integration in (3.23) as in [17]. $\mathcal{L}_{\text{positive}}$ accumulates the log-intensities of the sample observations and $\mathcal{L}_{\text{negative}}$ penalizes the intensity function for the sampled points, which represent the rest of the spatio-temporal interval. We sum the two terms with a weight term as

$$\tilde{\mathcal{L}} = \mathcal{L}_{\text{positive}} + \alpha \mathcal{L}_{\text{negative}}, \tag{3.24}$$

to control the effect of the negative and the positive terms on the overall $\tilde{\mathcal{L}}$.

We define the optimal parameters as the set of model parameters $\mathbf{\Theta}^*$ that exceeds a certain tolerance level for the log-likelihood $\mathcal{L}_{\text{tol}}$, i.e.

$$\mathcal{L}(\mathbf{N} | \Theta^*) \geq \mathcal{L}_{\text{tol}} \tag{3.25}$$

To compute the optimal set of parameters, we split the given set of samples $\mathbf{N} = \{\mathbf{n}_i\}_{i=1}^{I}$ into three subgroups; training set ($\mathbf{N}_{\text{train}} = \{\mathbf{n}_i\}_{i=1}^{I_{\text{train}}}$), validation set ($\mathbf{N}_{\text{val}} = \{\mathbf{n}_i\}_{i=I_{\text{train}}}^{I_{\text{val}}}$) and test set ($\mathbf{N}_{\text{test}} = \{\mathbf{n}_i\}_{i=I_{\text{val}}}^{I}$). We use the training set for the parameter optimization.

We update the set of parameters starting from the initial values $\mathbf{\Theta}_0$ with the gradient updates. For a model parameter $\boldsymbol{\theta} \in \mathbf{\Theta}$ we have

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \eta \nabla_{\boldsymbol{\theta}} \tilde{\mathcal{L}}(\mathbf{N}_B), \tag{3.26}$$

where $\mathbf{N}_B$ is a mini-batch consisting of randomly picked samples from the training set $\mathbf{N}_{\text{train}}$. The learning rate η scales the magnitudes of parameter updates and $\nabla_{\boldsymbol{\theta}}\tilde{\mathcal{L}}(\mathbf{N}_b)$ is the gradient vector of $\tilde{\mathcal{L}}(\mathbf{N}_b)$ with respect to the parameter $\boldsymbol{\theta}$. Using (3.22) and (3.24), we have

$$\frac{\partial\tilde{\mathcal{L}}(\mathbf{N}_b)}{\partial\boldsymbol{\theta}} = \frac{1}{B}\sum_{b=1}^{B}\left(\left[\frac{1}{\lambda_k(t_b,\mathbf{l}_b)}\right]_{k=1}^{K} + \alpha\mathbb{1}_K\right)\odot\frac{\partial\boldsymbol{\lambda}(t_b,\mathbf{l}_b)}{\partial\boldsymbol{\theta}}. \tag{3.27}$$

Partial derivatives of $\boldsymbol{\lambda}$ with respect to the Hawkes parameters are

$$\frac{\partial\lambda_k(t_b,\mathbf{l}_b)}{\partial\gamma_k} = \boldsymbol{\Gamma}_{:,k}^T\sum_{\mathbf{n}\in\mathbf{N}_\Omega(t_b,\nu)}\rho_k(\mathbf{l})(t-t_b)e^{\gamma_k(t-t_b)}, \tag{3.28}$$

$$\frac{\partial\lambda_k(t_b,\mathbf{l}_b)}{\partial\boldsymbol{\Gamma}_{i,k}} = \sum_{\mathbf{n}\in\mathbf{N}(t_b,\nu)}\rho_i(\mathbf{l})e^{\gamma_k(t-t_b)} \tag{3.29}$$

and

$$\frac{\partial\lambda_k(t_b,\mathbf{l}_b)}{\partial\mu_k} = \rho_k(\mathbf{l}_b). \tag{3.30}$$

Similarly, the derivatives of the tree parameters are

$$\frac{\partial\boldsymbol{\lambda}_k(t_b,\mathbf{l}_b)}{\partial\boldsymbol{\rho}} = \boldsymbol{\lambda}(t_b,\mathbf{l}_b)\odot\left(\mathbb{1}_K + \mathbf{J}^T\mathbf{p}(\mathbf{l}_b)\right), \tag{3.31}$$

where $\mathbf{J}$ is the jacobian matrix. The element $\mathbf{J}_{i,j}$ corresponds to $\partial\lambda_i/\partial\rho_j$. Derivative of the weight vector $m_{r\mathbf{w}}$ is

$$\frac{\partial\boldsymbol{\rho}(\mathbf{l}_b)}{\partial m_{r\mathbf{w}}} = \sum_{k=1}^{K}\prod_{\substack{l=1,\\ l\neq l'}}^{L}\pi(\mathbf{n}_b,\nu_k^{(l)},\tau_k^{(l)})\sigma(1-\sigma)\mathbf{l}_b. \tag{3.32}$$

where l' is the level of the node m_r. Similarly,

$$\frac{\partial \boldsymbol{\rho}(\mathbf{l}_b)}{\partial m_{r\mathbf{b}}} = \sum_{k=1}^{K} \prod_{\substack{l=1,\\ l\neq l'}}^{L} \pi(\mathbf{n}_b, \nu_k^{(l)}, \tau_k^{(l)}) \sigma(1-\sigma), \tag{3.33}$$

where $\sigma = \sigma(m_{r\mathbf{w}}^T \mathbf{l}_b + m_{r\mathrm{b}})$.

By the definition in (3.1), the conditional intensity should be a non-negative function. Optimizing the model parameters with gradient based updates without imposing any constraint could result in a negative intensity. Thus, we use the softplus function

$$\tilde{\lambda}(t, \mathbf{l}|\Omega(t)) = \log(1 + e^{\lambda(t,\mathbf{l}|\Omega(t))}) \tag{3.34}$$

to make the intensity non-negative through the optimization steps.

Algorithm 1 Likelihood based first-order gradient training procedure.

Require: Hyperparameters, Samples $\mathbf{N}$
  $i = 0$
  Randomly initialize parameters, $\boldsymbol{\theta}_{\text{tree}}$, $\boldsymbol{\theta}_{\text{hawkes}}$.
  $\boldsymbol{\Theta}_i = [\boldsymbol{\theta}_{tree}, \boldsymbol{\theta}_{\text{hawkes}}]$
  Sample J negative points $\{[t_j, x_j, y_j]\}_{j=1}^{J}$
  $\tilde{\mathcal{L}}_{\text{positive}} \leftarrow \sum_{i=1}^{I} \log \lambda(t_i, \mathbf{l}_i|\Omega(t_{i-1}))$
  $\tilde{\mathcal{L}}_{\text{negative}} \leftarrow \sum_{j=1}^{J} \Lambda_{t_j}(t_j, \mathbf{l}_j)$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}}_{\text{positive}} + \alpha \tilde{\mathcal{L}}_{\text{negative}}$
  while $\tilde{\mathcal{L}} < \log \mathcal{L}_{\text{tol}}$ **do**
    Sample J negative points $\mathbf{J} \leftarrow \{[t_j, x_j, y_j]\}_{j=1}^{J}$
    $\tilde{\mathcal{L}}_{\text{positive}} \leftarrow \sum_{i=1}^{I} \log \lambda(t_i, \mathbf{l}_i|\Omega(t_{i-1}))$
    $\tilde{\mathcal{L}}_{\text{negative}} \leftarrow \sum_{j=1}^{J} \Lambda_{t_j}(t_j, \mathbf{l}_j)$
    $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}}_{\text{positive}} + \alpha \tilde{\mathcal{L}}_{\text{negative}}$
    Compute updates, $\Delta\boldsymbol{\theta}_i \leftarrow \nabla_{\boldsymbol{\theta}} \tilde{\mathcal{L}}(\mathbf{N}, \mathbf{J})$
    $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \eta * \Delta\boldsymbol{\theta}_i$
    $i \leftarrow i + 1$
  end while

Our optimization algorithm is presented in Algorithm 1, which is an iterative optimization procedure. We first randomly initialize a decision tree with depth ℓ and point processes corresponding to all subregions. At the ith iteration, we first compare the current log-likelihood with the tolerance level $\mathcal{L}_{\text{tol}}$. If

it is not exceeded, we compute the positive log-likelihood of the model with the current parameters. We also sample points in the spatio-temporal interval and compute the negative term in the log-likelihood using these points. Using the log-likelihood, we compute the parameter updates, $\Delta\boldsymbol{\theta}_i$ for each model parameter. If the tolerance level is achieved at a step, we terminate the training process.

To find the best set of hyperparameters, we perform multiple training runs with a different set of hyperparameters. At the end of each run, we compute the validation performance on the validation set. We pick the set that yields the highest log-likelihood. After the best set of hyperparameters are found, we compute the test performance with the same metric.

In our training procedures, we use the stochastic gradient-ascent method, which uses the first order derivatives to update the model parameters with respect to an objective function. To increase the convergence speed of our training procedures, we also use the Adaptive Moment Estimation (ADAM) optimizer [36], which uses the first order derivatives. Thus, it does not introduce any computational cost for the update equations.

3.5 Online Model Optimization

In many real-life applications, data flows in a streaming manner. Prediction model needs to be updated using the newly observed samples. However, starting the training procedure described in the previous section would consume excessive time and memory as the data would grow indefinitely. To ameliorate this issue, we present a fully-online procedure that is suited better to the real-time applications.

Note that the objective function in (3.22) require all past sample observations and uniformly sampled points from time t_0 to T. Hence, it is not suitable for online setup since the sequence of past samples will expand as new samples are observed. To this end, we modify the objective function for online setup. Suppose that the sample $\mathbf{x}_i = [t_i,\ \mathbf{l}_i]$ is observed. We compute the term $\mathcal{L}_p$ as

$$\mathcal{L}_{p_{\text{online}}} = \log \lambda(t_i, \mathbf{l}_i | \Omega(t_{i-1})). \tag{3.35}$$

Similarly, we modify the term $\mathcal{L}_n$ as

$$\mathcal{L}_{n_{\text{online}}} = -\sum_{j=1}^{J} \lambda(t_j, \mathbf{l}_j | \Omega(t_{i-1})), \tag{3.36}$$

in which we sample random points only in the interval $[t_{i-1},\ t_{i-1} + \delta_{i-1}]$ where δ_{i-1} is the length of the temporal interval, i.e. $t_j \in [t_{i-1},\ t_{i-1} + \delta_{i-1}]$. We set δ_{i-1} to $20\Delta t$ as we limit the integration in (3.16) in the offline setup. As a result, we train our model in online setup as the updates will only depend on the most recent sample observations. This procedure is given in Algorithm 2 as a pseudocode.

Algorithm 2 Online training procedure.

Require: Hyperparameters: ρ
  Randomly initialize depth-L tree, θ_{tree}.
  Randomly initialize Hawkes parameters θ_{hawkes}.
  Initialize $X = \{\}$
  $\boldsymbol{\Theta} \leftarrow [\boldsymbol{\theta}_{tree}, \boldsymbol{\theta}_{hawkes}]$
  while $\mathbf{x}$ arrives **do**
    $X \leftarrow \{X_{\phi-1}, \mathbf{x}\}$
    Compute $\mathcal{L}_{p_{\text{online}}}$, $\mathcal{L}_{n_{\text{online}}}$ in (3.35), (3.36)
    Compute updates, $\Delta\Theta$
    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} + \eta * \Delta\boldsymbol{\Theta}$
  end while

In Algorithm 2, we append the newly arrived sample, $\mathbf{x}$ to a set X. However, we only keep the ϕ most recent samples in that set as the rest of the samples do not effect the intensity computations. Therefore, with Algorithm 2, we present a setup that can infer the model parameters under non-stationary source as it updates the model parameters with newly observed samples.

Chapter 4

Experiments and Simulations

In this section, we present the experiment setup and the result comparisons on different datasets and for different models. We first describe the real-life datasets by giving brief information about their content and how they are processed. Then, we explain the algorithms that we use in our comparison. Finally, we investigate the effect of choosing different hyperparameters in our model by giving empirical results.

As described earlier, we assess the performance of our model in two setups. For event count estimation in a window, we use a real-life criminal events dataset and an earthquake events dataset. For the precise prediction of event times and locations, we use only the earthquake dataset. Details for each setup are given in the following sections.

4.1 Real-life Datasets For Spatio-Temporal Prediction

Here, we explain the datasets used in the comparison of our algorithm with the existing approaches. The first dataset we use is the Chicago Crime Dataset [37].

The dataset contains criminal event records that occurred in Chicago City between 2001 and 2021. Each record is tagged with its date, latitude, and longitude. Also, each event has a description of the event details and an event class. We focus our attention on the recent entries between 2012 and 2017. Moreover, we only consider the time and location-stamps of the events for predicting future events.

Since many criminal activities take place even in small spatial and temporal subintervals, rather than predicting the individual event time and locations, we estimate the total number of events that will occur in a specified temporal horizon and spatial area. The details of this evaluation are given in the following subsection.

The second dataset is the Significant Earthquakes dataset [38]. The dataset is recorded by the National Earthquake Information Center (NEIC). The dataset contains entries of earthquake event records with their latitude, longitude, depth, magnitude and date. It covers all the earthquake events from 1900 to 2000 that have a magnitude higher than 5.5. Location and time prediction of the earthquake events using the past data fits our problem description since the events are spatio-temporal samples.

Since the dataset only contains the earthquake event records with magnitudes higher than 5.5, it excludes the aftershock records. Aftershocks of earthquakes could have been effectively modeled by the Hawkes process formulation [20]. To this end, we choose a particular spatial region L between the latitudes 31.92° and 72.05° and between the longitudes 110.2° and 180.1° from the whole data, which contains a large number of sequential earthquakes. This region corresponds to the east of China and Russia, and all of Japan and Korea.

We convert the earthquake records to spatio-temporal sequence $\mathbf{N} = \{\mathbf{n}_i\}_{i=1}^{I}$. In particular, we generate 20 sequences by splitting the data into equal-length parts in time, i.e. 5 years. We further process the data by converting the dates of the events to time differences with respect to the first event time in seconds and scale the differences as $\{t_i/(3600 \times 24 \times 30)\}_{i=1}^{I}$. Moreover, the spatial locations,

i.e. latitude and longitude of the events are scaled as $L = \{(x,\ y) \in \mathbb{R}^2 | -10 < x < 10,\ -10 < y < 10\}$.

Since the earthquake events are separated distantly in time, we estimate the time and location of specific events directly. Thus, we assess the performance of our algorithm directly on these estimates, which we explain in the following subsection.

Finally, we also measure the performance of our algorithm on a simulated dataset. This dataset is artificially generated via the Thinning algorithm [39] using the same model structure.

4.2 Benchmark Models For Performance Comparison

In this section, we describe the algorithms that we use in our benchmark for performance comparison. Since we investigate the performance of our algorithm on estimating both the total number of events and the event time and locations, we use two different set of algorithms.

4.2.1 Event Count Estimation Models

We estimate the total number of event in the horizon window by accumulating the intensity function. Although our model can estimate the number of events in any arbitrary interval, for evaluation purposes, we discretize the spatial region into grids with a selected resolution. Consequently, we obtain a 2-D array containing the number of events in each grid. Hence, we generate a 2-D frame containing the expected number of events in the prediction horizon for the spatial region.

Finally, since the real-life data consists of exact event dates, latitudes and longitudes, we convert these values to relative position stamps with selected temporal

and spatial resolutions. Moreover, we assess the performance of the predictions using the Root Mean-Squared-Error (RMSE) metric for each grid.

We use a CNN model, which processes the discrete data structure and predicts the event amount in a specified horizon. For every time step, we make a prediction using the most recent frames. The model applies several convolutional kernels to the most recent frames and generates an array containing the counts.

During the training, we minimize the mean squared error between the generated array and the ground truth array over the whole training session. We update the model parameters using the ADAM optimizer with respect to the mean squared error between the generated frames and the ground-truth frames.

4.2.2 Event Time and Location Prediction Models

The first model in this comparison is the linear regression (we refer to as Linear), which predicts the displacement in time and space using the past displacement vectors with a linear model. Predictions are computed as

$$\begin{aligned}
\Delta t_i &= \mathbf{w}_{\mathrm{tt}}^T[\Delta t_{i-k}]_{k=1}^K + \mathbf{w}_{\mathrm{tx}}^T[\Delta x_{i-k}]_{k=1}^K + \mathbf{w}_{\mathrm{ty}}^T[\Delta y_{i-k}]_{k=1}^K, \\
\Delta x_i &= \mathbf{w}_{\mathrm{xt}}^T[\Delta t_{i-k}]_{k=1}^K + \mathbf{w}_{\mathrm{xx}}^T[\Delta x_{i-k}]_{k=1}^K + \mathbf{w}_{\mathrm{xy}}^T[\Delta y_{i-k}]_{k=1}^K, \\
\Delta y_i &= \mathbf{w}_{\mathrm{yt}}^T[\Delta t_{i-k}]_{k=1}^K + \mathbf{w}_{\mathrm{yx}}^T[\Delta x_{i-k}]_{k=1}^K + \mathbf{w}_{\mathrm{yy}}^T[\Delta y_{i-k}]_{k=1}^K,
\end{aligned} \tag{4.1}$$

which are using the K most recent observations.

Another method for comparison is the standard RNN model, which predicts the displacement in time and space using the state vector. The state transition is computed as

$$\mathbf{h}_i = \tanh(\mathbf{W}_{\mathrm{x}}^T[\Delta t_{i-1}, \Delta x_{i-1}, \Delta y_{i-1}] + \mathbf{W}_{\mathrm{h}}^T\mathbf{h}_{i-1}),$$

where $\tanh = (e^x - e^{-x})/(e^x + e^{-x})$. We compute the displacements as

$$\Delta t_i = \mathbf{w}_{\mathrm{to}}^T \mathbf{h}_i,$$
$$\Delta x_i = \mathbf{w}_{\mathrm{xo}}^T \mathbf{h}_i,$$
$$\Delta y_i = \mathbf{w}_{\mathrm{yo}}^T \mathbf{h}_i.$$

We also repeat the same analysis for the LSTM model.

4.2.3 Hybrid Approaches

Finally, we compare our model with the point process based approaches. These approaches are applicable to both evaluation setups. First, we test the check-in time prediction model presented in [9] (RSTPP). Here, an RNN is used to extract temporal features from the historical check-in times of users. Combining with the user trajectory information, the intensity is computed as

$$\lambda(t, \mathbf{l}|\Omega(t)) = \exp(\mathbf{w}_{\mathrm{h}}\mathbf{h}_i + w_{\mathrm{t}}(t - t_{i-1}) + w_{\mathrm{l}}||\mathbf{l} - \mathbf{l}_{i-1}||^2 + b), \tag{4.2}$$

where $\mathbf{h}_i$ is the state vector of the RNN model, which has the state transition

$$\mathbf{h}_i = \tanh(\mathbf{W}_{\mathrm{x}}^T \mathbf{x}_{i-1} + \mathbf{W}_{\mathrm{h}}^T \mathbf{h}_{i-1}). \tag{4.3}$$

Originally, user activity features and past check-in time and locations were included in the vector $\mathbf{x}_i$. However, we only use the past time and location information.

Similar to the check-in time prediction, Recurrent Marked Temporal Point Process (RMTPP) approach also estimates the intensity via an RNN [8]. However, in this case, the intensity is defined only for the temporal axis and the location is predicted with markers. Both the intensity and the location markers are computed from the state vector of the RNN model.

4.3 Evaluation of the Presented Algorithm

In this section, we present the experiment results for assessing the presented algorithm on real-life datasets. We split the data sequence $\{X_t\}_{t=1}^{T}$ into three sets: training, validation, and test sets, which are ordered with time. These sets have split boundary indices T_{train}, $T_{\text{validation}}$, and T, respectively. Thus, the training, validation and test sets are defined as $\mathbf{X}_{\text{train}} = \{X_t\}_{t=1}^{T_{\text{train}}}$, $\mathbf{X}_{\text{validation}} = \{X_t\}_{t=T_{\text{train}}}^{T_{\text{validation}}}$, and $\mathbf{X}_{\text{test}} = \{X_t\}_{t=T_{\text{validation}}}^{T}$.

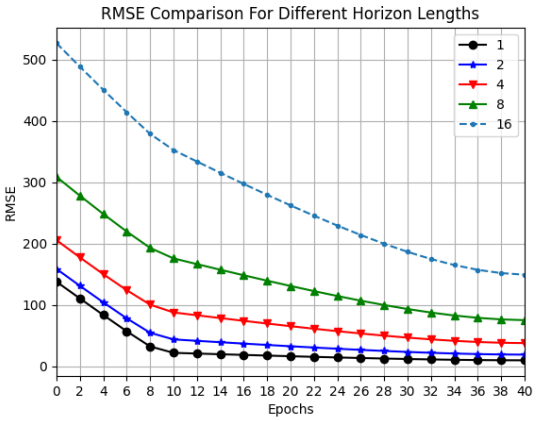
4.3.1 Prediction Horizon Effect

We investigate the performance for different prediction horizon lengths. For horizon length h, the prediction frame $\hat{y}_t$ is generated by accumulating the intensity in the horizon interval. We have selected spatial and temporal resolutions as 3 km and 30 days. Furthermore, we choose the past window length as 30 days due to its performance. From Fig. 4.1(a) and 4.1(c), we can observe that the error increases with the increasing horizon length. Moreover, we can see that the increase in error is not linearly increasing with the horizon length. A unit length of horizon in the figures correspond to 15 days.

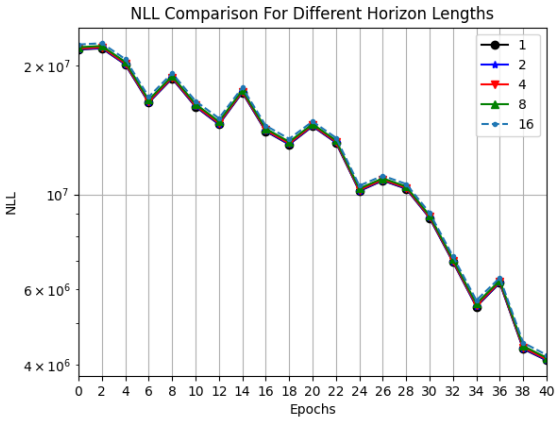
We assess the performance on three datasets: Chicago Crime dataset, Earthquake dataset and the simulated dataset. For the crime and simulated dataset, we compute the RMSE and the NLL in the validation sets through epochs as in Fig. 4.1(a), 4.1(b), 4.1(c), and 4.1(d) respectively.

We also perform the same experiments on the earthquake dataset, which contains less events and has a sparse distribution. Finally, we report the test results for all datasets and all horizon lengths on Table 4.1. Furthermore, we also report the RMSE error when predicting number of events occurring in a unit spatio-temporal cell with the mean number of events per cell for comparison.

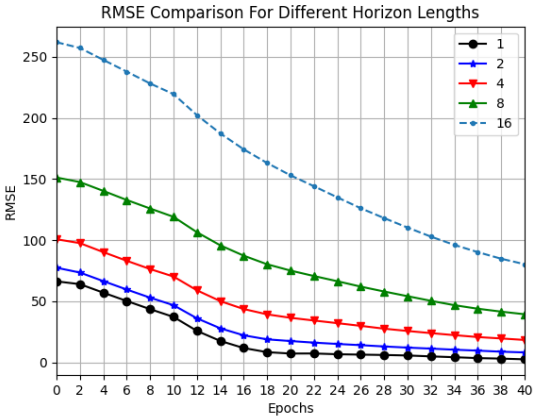
At the end of the training, the algorithm achieved an RMSE of 10.21 on the



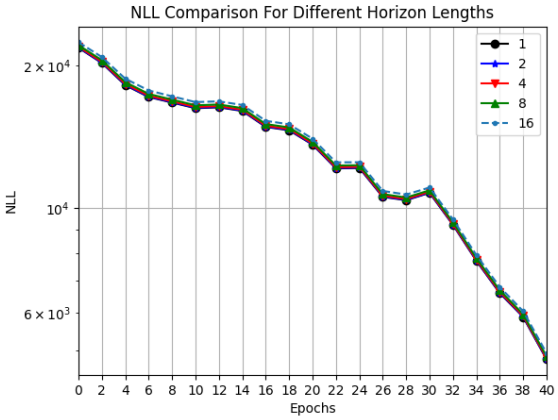
(a) Prediction horizon RMSE comparison for the Chicago Crime Dataset



(b) Prediction horizon NLL comparison for the Chicago Crime Dataset



(c) Prediction horizon RMSE comparison for the Simulated Dataset



(d) Prediction horizon NLL comparison for the Simulated Dataset

Figure 4.1: Prediction horizon performance comparison for our algorithm on two datasets.

crime dataset, which is significantly less than 526.21, the average number of events observed in 15 day frame. We observe that the model achieves small improvements in the prediction performance after the horizon length 16 (240 days). After this point, the RMSE performance is close to the average event numbers. Moreover, we also observe that the performances are better for the experiments on simulated data, as we have generated the simulated data with the same model structure.

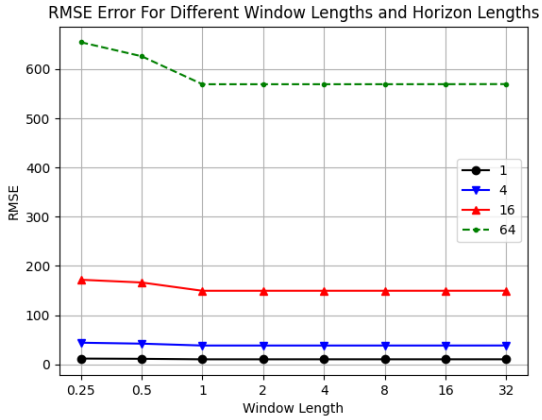
4.3.2 Past Temporal Window Length Effect

We also measure the effect of choosing different temporal window lengths on the RMSE performance. We perform 4 experiments with window lengths 1, 4, 16, and 64. A unit of temporal length corresponds to 15 days as in the previous setup. For 4 window lengths, we measure the RMSE with 4 different horizon lengths on the crime, simulated and earthquake data as in Fig. 4.2(a), 4.2(b), and 4.2(c), respectively.

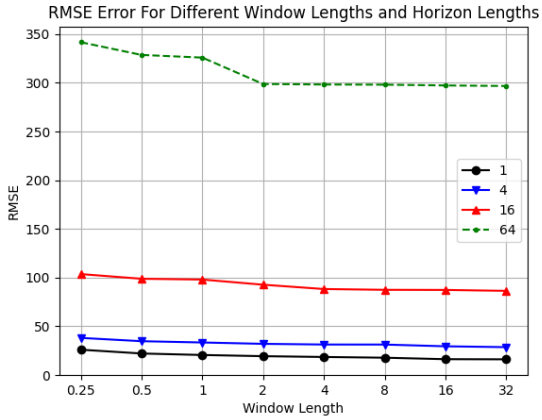
In all of the experiments, we can see that the RMSE converges to a level after a certain window length, which shows the temporal dependency length for each dataset. Specifically, for the crime dataset, we can see that choosing the window length 1 (15 days) is sufficient.

We obtain the results in Fig. 4.2(c) on the earthquake dataset. In this case, we can observe that the improvement is more apparent as the data consists of temporally spaced events. This is due to the collection of earthquakes that have magnitude higher than a certain level. This filters out preshocks and aftershocks, which eventually results in a sparse event sequence. Thus, to capture more past events, the time window should be selected longer, as it is also shown empirically in the figures.

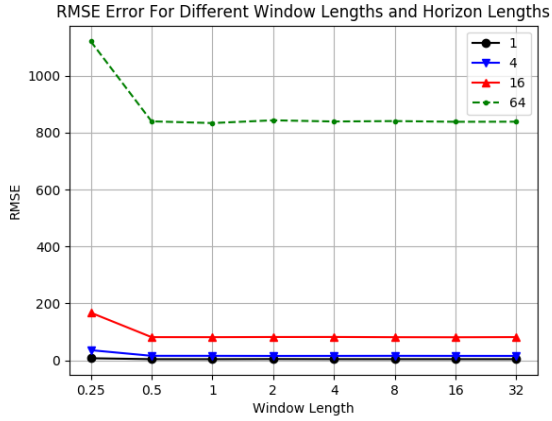
We also report the RMSE performances on Table 4.2 and 4.3



(a) Past temporal window length RMSE comparison for the Chicago Crime Dataset



(b) Past temporal window length RMSE comparison for the Simulated Dataset



(c) Past temporal window length RMSE comparison for the Earthquake Dataset.

Figure 4.2: Past temporal window length comparison for different horizon lengths on two datasets.

4.3.3 Parallel and Separate Training

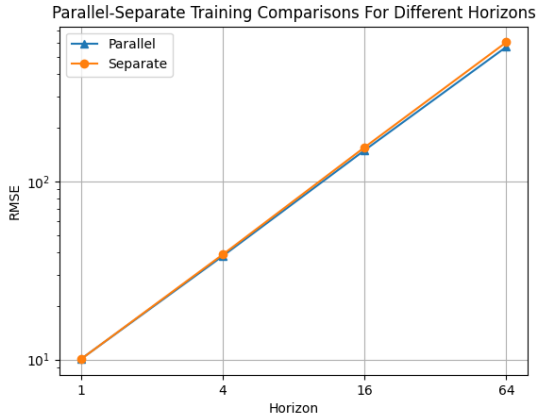
We perform two different training modes for different horizon lengths. In separate training mode, the model is trained with a single negative-log-likelihood loss (NLL) function that is computed for a specific horizon length. In the parallel mode, the model is trained with the NLL loss for all horizons at the same time. This method has positive impact on the model performance due to better generalization capability. Training the model for predicting the events in near and far future results in a better set of parameters that are capable of modeling the actual sequence. We show this with the empirical results in Fig. 4.3(a) and 4.3(b), and 4.3(c) for the crime, earthquake, and simulated data, respectively.

In all of the datasets, the parallel training outperforms the separate training. For this reason, in all experiment sets, we use the parallel training mode. In addition to these results, we also present the performance on the test set in Table , where P-TreeHawkes and S-TreeHawkes correspond to the parallel and separate scenarios respectively.

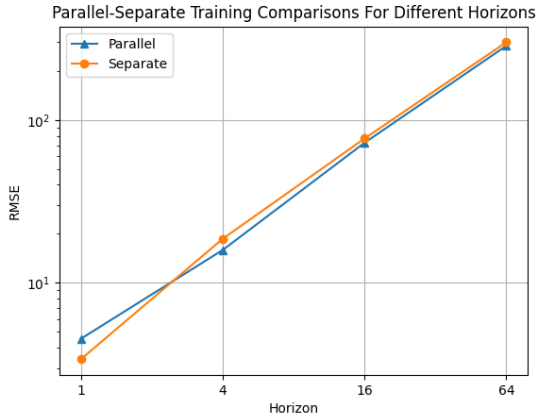
4.3.4 Spatial Subregion Number Effect

Since the presented algorithm consists of a decision tree that is used to partition the spatial region of interest, we experiment with different tree depths. As the depth of the tree increases, the number of spatial subregions increase exponentially. The spatial precision in the predictions will increase with the increasing number of subregions. We demonstrate the effect on the performance for different tree depths as in Fig. 4.4(a) and 4.4(b), respectively.

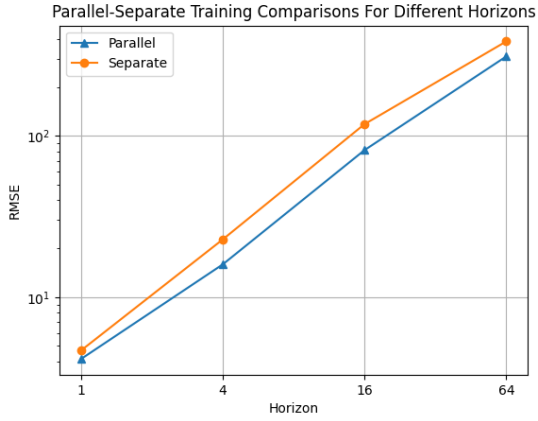
From the results in the figures, as the number of spatial subregions increase, the model performance also increase. We observe that the performance converges to a level at the tree with depth 4 on the crime dataset. Thus, we use a 4-level tree in our experiments for comparison. We also perform the same set of experiments on the earthquake dataset. Although we observe a similar increase



(a) Parallel and Separate training RMSE comparison for the Chicago Crime Dataset

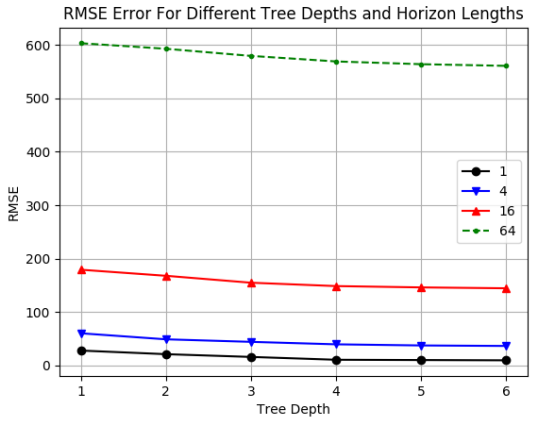


(b) Parallel and Separate training RMSE comparison for the Simulated Dataset

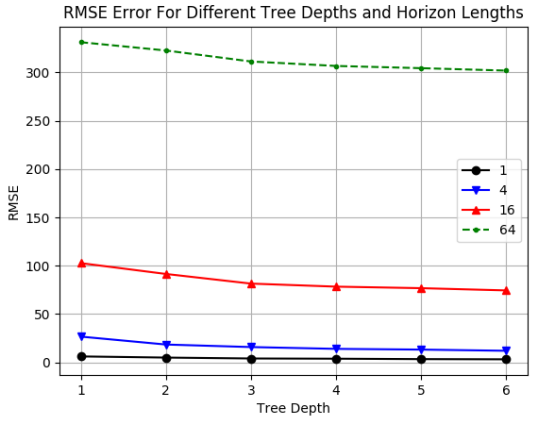


(c) Parallel and Separate training RMSE comparison for the Earthquake Dataset

Figure 4.3: Parallel and Separate training comparison on two different datasets.



(a) RMSE comparison for different tree depths on the Chicago Crime Dataset



(b) RMSE comparison for different tree depths on the Earthquake Dataset

Figure 4.4: Performance comparison for different number of spatial subregions on the crime and earthquake datasets.

in the performance level with the increasing tree depth, 3 is selected for the tree depth on this dataset. Since the tree structure we are using is a binary tree, the effective number of regions for these scenarios are 16 and 8, respectively.

Horizon (15 days)	Chicago RMSE	Chicago Average	Simulated RMSE	Simulated Average	Earthquake RMSE	Earthquake Average
1	10.21	29.56	3.12	5.37	4.12	13.12
2	21.27	31.02	7.61	10.74	7.01	26.24
4	39.15	62.04	17.74	21.48	15.93	52.48
8	75.44	124.08	38.11	42.96	38.20	104.96
16	148.32	248.16	78.12	85.92	81.55	209.92
32	294.12	496.32	158.80	171.84	157.49	419.84
64	569.46	992.64	313.78	343.68	311.25	839.68

Table 4.1: Experiment results on crime and simulated datasets for different horizon lengths.

4.4 Performance Comparison With Benchmark Models

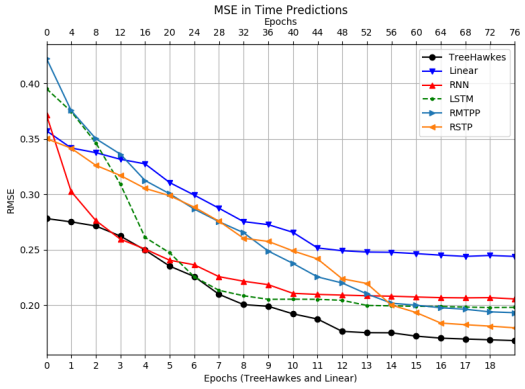
RMSE comparisons for the crime dataset for different models are shown in Table 4.2. In Table 4.2, S-TreeHawkes corresponds to the separate training scenario and P-TreeHawkes corresponds to the parallel training scenario. It is clear that the performance of our algorithm surpasses the rest of the approaches on this dataset. However, the performance margin decreases with the increasing horizon length. This is due to the averaging effect for the event counts in long temporal intervals. As the temporal horizon length is increased, the number of events inside the horizon will converge to the average density of events in time.

For the crime dataset, we can see that the CNN model achieves the second best performance in higher horizon lengths after our algorithm. This is due to the capability of CNNs in capturing spatial patterns. Moreover, the CNN model is used in an auto-regressive manner, which allows it to learn temporal patterns in a fixed window length. The exceptional performance of our model and the CNN is due to the performance in modeling spatial interactions. Unlike the other methods that independently model spatial patterns, or incorporate the location information additively, the kernel mechanism in our model demonstrated an exceptional performance. The performance difference between our model and the CNN is due to the non-stationary modeling ability of our model in the spatial domain. Since the CNN applies the same kernel through the whole space, different spatial dynamics may not be effectively processed under different dynamics. On the other hand, we model each partition with an individual process.

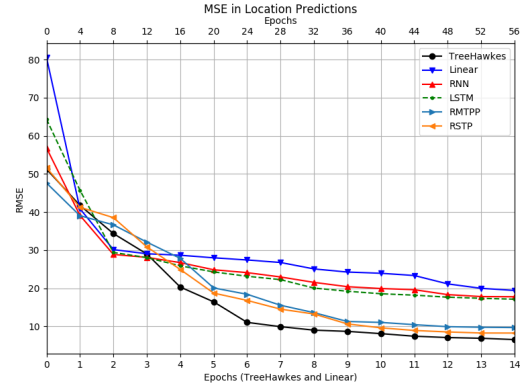
RMSE comparisons for the earthquake dataset are shown in Table 4.2. For this comparison, we can see that the convergence between performances is less apparent. This is due to the nature of the earthquake sequences, which are more sparsely distributed compared to criminal event records.

We also observe that the point process based approaches outperform the CNN model. This results shows that the temporal patterns in certain regions have

more significant effect compared to spatial patterns on the future event arrivals. However, both of the probabilistic approaches yield worse results compared to our model. Unlike the independent temporal and spatial location modeling, our model is capable of jointly modeling both temporal and spatial patterns.



(a) RMSE comparison for different models on the time estimation on Earthquake Dataset



(b) RMSE comparison for different models on the location estimation on Earthquake Dataset

Figure 4.5: Performance comparison in both time and location estimations on the earthquake dataset.

Finally, to complete our performance analysis, we also provide the results achieved for the time and location estimations of earthquake events in Fig. 4.5(a) and 4.5(b) for the validation set. We also give the test results in Table 4.4 for both time (MSE-t) and location (MSE-l) estimations. In both time and location estimations, our model achieves the best performance, which is followed by the point process based approaches. This is due to the capabilities of these methods

for modeling sparse sequences. However, all methods weakly combine temporal and spatial information as they either independently model each domain, or incorporate information with a simple model such as a linear combination.

As it can be seen from the figure, performance of the linear model is limited to a certain level both in time and location predictions. This is due to the simple structure of the model, which is a linear combination of the past sample times and locations. For the RNN and LSTM models, we obtain better results compared to the linear model both in time and location predictions. The reason behind this performance increase is due to the nonlinearity and the inherent state introduced by the RNN and LSTM transition equations. Instead of using a fixed number of past sample times and locations for prediction, this model computes the predictions using the state vector.

Model	1	4	16	64
P-TreeHawkes	10.21	**39.15**	**148.32**	**569.46**
S-TreeHawkes	**9.95**	40.83	154.41	582.21
CNN	11.52	46.64	167.36	604.75
RMTPP	15.43	57.11	182.98	622.34
RSTP	16.65	57.73	180.36	616.47

Table 4.2: RMSE Test results on the Chicago Crime Dataset

Model	1	4	16	64
P-TreeHawkes	**4.12**	**15.93**	**81.55**	**311.25**
S-TreeHawkes	4.55	23.40	119.26	411.63
CNN	15.36	57.31	167.47	468.11
RMTPP	7.38	23.62	126.89	430.27
RSTP	5.97	22.45	130.71	454.84

Table 4.3: RMSE Test results on the Earthquake Dataset

Models	MSE-t	MSE-l
TreeHawkes	0.162	6.47
Linear	0.271	33.28
RNN	0.202	31.86
LSTM	0.186	29.78
CNN	0.264	25.47
RSTPP	0.192	15.6
RMTPP	0.189	19.2

Table 4.4: Test results for the time and location estimations on the earthquake dataset.

Chapter 5

Conclusion

We introduce a novel spatio-temporal prediction model that predicts the times and locations of the samples as well as the expected number of events in a spatio-temporal interval using the past samples. Our formulations are based on the Hawkes process, but can be readily extended to other point process models depending on the application. We also incorporate the spatial and temporal connections between the past samples and the intensity function via kernel mechanisms. Furthermore, we also partition the spatial region into subregions via an adaptive decision tree. Therefore, we optimize our point process parameters jointly with the subregion boundaries using a likelihood based objective function and the stochastic gradient descent method. Thanks to self-exciting and non-stationary intensity formulation of our point process model and the adaptive partitioning mechanism, we are able to represent highly sparse and non-stationary data. Finally, we demonstrate significant performance improvements through an extensive set of experiments where we compare our model with the baseline and standard approaches on a real-life crime and an earthquake dataset.

Bibliography

[1] M. Xu, Y. Yang, M. Han, T. Qiu, and H. Lin, "Spatio-temporal interpolated echo state network for meteorological series prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 6, pp. 1621–1634, 2019.

[2] V. Roberto and C. Chiaruttini, "Seismic signal understanding: a knowledge-based recognition system," *IEEE Transactions on Signal Processing*, vol. 40, no. 7, pp. 1787–1806, 1992.

[3] H. Quan, A. Khosravi, D. Yang, and D. Srinivasan, "A survey of computational intelligence techniques for wind power uncertainty quantification in smart grids," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–18, 2019.

[4] B. Wang, P. Yin, A. L. Bertozzi, P. J. Brantingham, S. J. Osher, and J. Xin, "Deep learning for real-time crime forecasting and its ternarization," *Chinese Annals of Mathematics, Series B*, vol. 40, no. 6, pp. 949–966, 2019.

[5] W. Zhang, X. Lai, and J. Wang, "Social link inference via multiview matching network from spatiotemporal trajectories," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2020.

[6] Q. Zhu, J. Chen, D. Shi, L. Zhu, X. Bai, X. Duan, and Y. Liu, "Learning temporal and spatial correlations jointly: A unified framework for wind speed prediction," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 1, pp. 509–523, 2019.

[7] C. Alippi, M. Roveri, and F. Trovò, "A self-building and cluster-based cognitive fault diagnosis system for sensor networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1021–1032, 2014.

[8] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1555–1564, 2016.

[9] G. Yang, Y. Cai, and C. K. Reddy, "Recurrent spatio-temporal point process for check-in time prediction," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2203–2211, 2018.

[10] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, 2017.

[11] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] L. Duan, T. Hu, E. Cheng, J. Zhu, and C. Gao, "Deep convolutional neural networks for spatiotemporal crime prediction," in *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, pp. 61–67, The Steering Committee of The World Congress in Computer Science, Computer ..., 2017.

[14] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, 2017.

[15] Z. Gao, X. Wang, Y. Yang, C. Mu, Q. Cai, W. Dang, and S. Zuo, "Eeg-based spatio–temporal convolutional neural network for driver fatigue evaluation,"

IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 9, pp. 2755–2763, 2019.

[16] B. Wang, D. Zhang, D. Zhang, P. J. Brantingham, and A. L. Bertozzi, "Deep learning for real time crime forecasting," *arXiv preprint arXiv:1707.03340*, 2017.

[17] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process," in *Advances in Neural Information Processing Systems*, pp. 6754–6764, 2017.

[18] D. R. Cox and V. Isham, *Point processes*, vol. 12. CRC Press, 1980.

[19] D. J. Daley and D. Vere-Jones, *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

[20] H. Kanamori, "Earthquake prediction: An overview," 2003.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[22] Y. Wang, B. Xie, N. Du, and L. Song, "Isotonic hawkes processes," in *International Conference on Machine Learning*, pp. 2226–2234, 2016.

[23] B. Wang, X. Luo, F. Zhang, B. Yuan, A. L. Bertozzi, and P. J. Brantingham, "Graph-based deep modeling and real time forecasting of sparse spatio-temporal data," *arXiv preprint arXiv:1804.00684*, 2018.

[24] G. O. Mohler, M. B. Short, P. J. Brantingham, F. P. Schoenberg, and G. E. Tita, "Self-exciting point process modeling of crime," *Journal of the American Statistical Association*, vol. 106, no. 493, pp. 100–108, 2011.

[25] B. Cseke, A. Zammit-Mangion, T. Heskes, and G. Sanguinetti, "Sparse approximate inference for spatio-temporal point process models," *Journal of the American Statistical Association*, vol. 111, no. 516, pp. 1746–1763, 2016.

[26] M. Adepeju, G. Rosser, and T. Cheng, "Novel evaluation metrics for sparse spatio-temporal point process hotspot predictions-a crime case study," *International Journal of Geographical Information Science*, vol. 30, no. 11, pp. 2133–2154, 2016.

[27] M. Jacobsen, *Point process theory and applications: marked point and piecewise deterministic processes*. Springer Science & Business Media, 2006.

[28] V. Isham and M. Westcott, "A self-correcting point process," *Stochastic Processes And Their Applications*, vol. 8, no. 3, pp. 335–347, 1979.

[29] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278–282, IEEE, 1995.

[30] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[31] B. C. Csáji *et al.*, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, no. 48, p. 7, 2001.

[32] F. M. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, 1995.

[33] B. P. Flannery, W. H. Press, S. A. Teukolsky, and W. Vetterling, "Numerical recipes in c," *Press Syndicate of the University of Cambridge, New York*, vol. 24, p. 78, 1992.

[34] R. E. Caflisch, "Monte carlo and quasi-monte carlo methods," *Acta Numerica*, vol. 7, pp. 1–49, 1998.

[35] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] C. P. Department, "Crimes - 2014: City of chicago: Data portal," Mar 2021.

[38] N. E. I. Center, "Significant earthquakes." https://www.usgs.gov/natural-hazards/earthquake-hazards/earthquakes, 2020.

[39] Y. Ogata, "On lewis' simulation method for point processes," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 23–31, 1981.