LOW-RANK REPRESENTATION AND EMBEDDING OF SEQUENTIAL DATA

A Dissertation
Presented to
The Academic Faculty

By

Jonathan Yuyang Zhou

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in the
College of Computing

Georgia Institute of Technology

May 2024

LOW-RANK REPRESENTATION AND EMBEDDING OF SEQUENTIAL DATA

Thesis committee:

Yao Xie, Ph.D.
Coca-Cola Foundation Chair Professor
School of Industrial & Systems Engineering
Georgia Institute of Technology

Chao Zhang, Ph.D.
Assistant Professor
School of Computational Science & Engineering
Georgia Institute of Technology

Date approved: May 1, 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ARI	Adjusted Rand Index
CP	Clustering Purity
DMHP	Dirichlet Mixture model of Hawkes Processes
DTW	Dynamic Time Warping
EDAT	Empirical Distribution of Inter-Arrival Times
EKG	Electrocardiogram
EM	Expectation Maximization
HMM	Hidden Markov Model
i.i.d.	Independent and Identically Distributed
LDA	Latent Dirichlet Allocation
LS	Least Squares
MD	Mirror Descent
MLE	Maximum Likelihood Estimation
MPPD	Marked Point Process Distance
NMI	Normalized Mutual Information
PGM	Probabilistic Graphical Model
SGD	Stochastic Gradient Descent
SOCP	Second-Order Cone Program
SSEM	Sequence Embedding by Spectral Kernel Decomposition
STPP	Spatio-Temporal Point Processes
SVD	Singular Value Decomposition
SVT	Singular Value Thresholding

TPP Temporal Point Processes

VI Variational Inequality

STANDARD NOTATIONS

Specific Sets

$\mathbb{R}$	Real Numbers.
$\mathbb{R}^n$	Real n-vectors ($n \times 1$ matrices).
$\mathbb{R}^{m\times n}$	Real $m \times n$ matrices.
$\mathbb{R}_{\geq 0}, \mathbb{R}_{>0}$	Nonnegative, positive real numbers.
$\mathbb{Z}$	Integers.
$\mathbb{Z}_{\geq 0}, \mathbb{Z}_{>0}$	Nonnegative, positive integers.

Vectors and Matricies

$(x_1, \ldots, x_n)$	Tuple of size n.
$\mathbf{X}$	Matrix.
$\mathbf{X}^*$	Transpose (Hermitian conjugate) of Matrix.
$\mathbf{x}$	Vector.
x	Scalar quantity.
$\mathcal{A}$	Linear Operator.
$\mathcal{A}^*$	Adjoint of Linear Operator.
$\operatorname{diag} \mathbf{A}$	Diagonal (vector) of matrix $\mathbf{A}$.
$\operatorname{vec} \mathbf{A}$	Matrix $\mathbf{A}$ as a vector stacked column-wise.
$\operatorname{rank} \mathbf{A}$	Rank of matrix $\mathbf{A}$.

Probability and Measure Theory

$\mathbb{E}[X]$	Expected value of random vector X.
$\mathbb{P}[S]$	Probability of event S.
$\mathbb{N}(\mathcal{S})$	Counting Measure on $\mathcal{S}$.
$\mathcal{P}(\mathcal{X})$	Power set of set $\mathcal{X}$.

Norms and Distances

$\lvert \cdot \rvert$	Absolute value.
$\lVert \cdot \rVert$	A norm.
$\lVert \mathbf{x} \rVert_2$	Euclidean (or ℓ_2)- norm of vector **x**.
$\lVert \mathbf{X} \rVert_*$	Nuclear norm of matrix $\mathbf{M}$.
$\lVert \mathbf{X} \rVert_F$	Frobenius norm of $\mathbf{M}$.
$\lVert \mathbf{X} \rVert_2$	Spectral norm of $\mathbf{M}$.
$B(c, r)$	Ball with center c and radius r.
$\mathrm{dist}(A, B)$	Distance between sets (or points) of A and B.
$\langle \cdot, \cdot \rangle$	An inner product.
$\langle \cdot, \cdot \rangle_2$	Euclidean inner product.
$\mathbf{A} \otimes \mathbf{B}$	Kronecker product between $\mathbf{A}$ and $\mathbf{B}$.

Functions and Derivatives

$f : A \rightarrow B$	A function on the set $\mathrm{dom}\, f \subseteq A$ into the set B
$\nabla_x f$	Gradient of f with respect to x.
$\mathcal{D} f$	(Derivative) Jacobian matrix of f.
$\mathbf{1}(\cdot)$	Indicator function on Boolean condition.
$\mathrm{Prox}_x(\cdot)$	Proximal Mapping with respect to centroid x.

SUMMARY

In the world around us we often encounter collections of discrete event sequences on timeframes, i.e., data expressed in the form $X = \{\mathbf{x}_n\}_{n=1}^{N}$, $\mathbf{x}_n = \{(t_i, x_i)\}_{i=1}^{T_n}$. The intervals on which time-points t_i occur may be continuous or discrete, and may be shared or different between the sequences. Illustrative examples include earthquake aftershocks, electrocardiograms and other multichannel signals, emergency call for service records, and electronic health records. Of recent interest has also been natural language and other forms of *symbolic sequential data*. In this work, I outline approaches to learning representations for individual sequences in the context of (1) a *common problem domain*, and (2) *individual representations* of each sequence in the collection. Each sequence will be described by a point in a parameter space, and the geometry of the parameter space corresponds to the problem domain. The points which describe the sequences each encode a *kernel* that describes the dynamics of the observed sequence data. We can use these dynamics to generate sequences similar to the observed time-series.

I discuss how to learn both the common domain and a representation for each of the sequences at the same time. The representation of the individual sequences shall be faithful to their original observations, but also leverage information about the domain in totality. The key to the representation learning lies in the common domain assumption, meaning that the sequences are in some sense similar to each other. We leverage this information by a *low rank* assumption on the space of parameters by which each sequence is described. I discuss how these points may be treated as an embedding and used for downstream tasks such as clustering, classification, and anomaly detection.

Following a common introduction given in Chapter 1, this work divided into two parts corresponding to different types of event sequences. Chapter 2 describes low-rank time-series embedding and parameter recovery for *autoregressive sequences*, where the events are sampled at fixed time-points occurring at a fixed frequency. I describe how to recover the individual sequence dynamics via a Monotone Variational Inequality (VI) formulation. Chapter 3 gives a treatment of the low rank recovery of *temporal point processes*, where each individual sequence corresponds to the realization of a point processes. We parameterize each point processes by the kernel function of their conditional intensity, and propose a framework to learn an embedding for event sequences in conjunction with a series of basis functions (parameterized by neural networks), which captures generation dynamics in the data's problem domain. We characterize the influence kernel of each observed trajectory by a point within the space of plausible basis functions and recover parameters via Maximum Likelihood Estimation (MLE). The formulation can capture a wide range of effects including non-stationary as well as both excitation and inhibition effects. In both cases, I detail the formulations, and provide illustrative numerical examples.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 The Ubiquity of Event Data

In the natural and human world, we often encounter collections of sequential data observations. Notable examples include:

1. A patient's records in a hospital
2. Video-on-demand watching behavior
3. The pricing of a financial security
4. The words people say in a conservation
5. Electrocardiogram (EKG) measurements from patients in a hospital

In addition the sequential character of these sequences, we see also that each trajectory is associated with a individual entity (e.g., person, a business, or a computer). In this work, we take the perspective that each entity is a stochastic source which generates a sequence, for which we have a single observation given to us in the data $X := \{\mathbf{x}_n\}_{n=1}^{N}$. Each of the $\mathbf{x}_n := \{(t_i, x_i)\}_{i=1}^{T_n}$ corresponds to an observations of the sequence generated by the nth entity. For a given *domain*, the sequences will share some commonality (e.g., the behavior of all EKG signals are all similar), but the sequences individually arises from a distinct generating processes. Our objective is, given our observations X, learn the common dynamics shared among the sequences, and also construct models for each sequence tailored to their observations. These dynamics should feasibility correspond to their original generating dynamics.

1.1.1 The Forms of Sequential Data

We consider a sets of event sequences (trajectories) $\mathcal{C} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ of form $\mathcal{T}_j = \{(t_1^j, \mathbf{x}_1^j), \ldots, (t_i^j, \mathbf{x}_i^j), \ldots, (t_n^j, \mathbf{x}_n^j)\}$. Each individual event $(t_i^j, \mathbf{x}_i^j)$ from trajectory j occurs at time-point $t_i^j \in [0, T)$ on a continuous finite time horizon, which is unique in the trajectory. Each x_i^j is likewise from in a mark/feature space $\mathcal{X} \subseteq \mathbb{R}^{d_\mathcal{X}}$ [1, 2]. In the most general setting, the length of individual trajectories $|\mathcal{T}_j|$ may be variable.

Specification: The following are interesting cases of the above

- When the time points come from a finite interval $T \in \sigma(\mathbb{Z}_{\geq 0})$ we have a *time series*. This is the subject of Chapter 2:
  - When $\mathcal{T}_j \in T \times \mathbb{R}$ we have a real-valued time series.
  - When $\mathcal{T}_j \in T \times \mathbb{R}^n$ we have a real-valued time series.
  - When $\mathcal{T}_j \in T \times [\Sigma]$, where $[\Sigma]$ is a discrete finite set of symbols, we have a simple *symbolic sequence*. For example a sequence of genes, or a collection of letters forming a document.
  - When $\mathcal{T}_j \in T \times \mathbb{Z}_+$ we have a count processes.
  - When $\mathcal{T}_j \in T \times \mathcal{P}[\Sigma]$, we have a complex symbolic sequence, for example the evolution of a discrete state vector.
- The time-points may also arise as a finite subset of a real time horizon $T_j \subseteq [0, T)$, $|T_j| \in \mathbb{Z}_{\geq 0}$, we have a *point processes*. This is the subject of Chapter 3.
  - $\mathcal{T} \in T_j \times \emptyset$ is known as a Temporal Point Processes (TPP), examples include passing cars on an expressway throughway.
  - $\mathcal{T} \in T_j \times \mathbb{R}^n$ is known as a Spatio-Temporal Point Processes (STPP). Examples include earthquake aftershocks.
  - $\mathcal{T} \in T_j \times [\Sigma]$ is known as a marked temporal point processes.

1.1.2 The Sequence Representation Problem

Recall that we consider an input collection of *trajectories* $\mathcal{C} = \{\mathcal{T}_i\}_{i=1}^{m}$ such that each trajectory $\mathcal{T}_i = \{(t_i^1, \mathbf{x}_i^1), (t_i^2, \mathbf{x}_i^2), \ldots\}$ represents a finite series of observations from a realization of some point process. Each of these trajectories may represent a collection of observations of a certain character or origin.

At a high level, each sequence may arise from a series of underlying *generative policies* $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_p\}$. Our objective is to find an association between the input trajectories and these policies, without supervision. Here, we suppose a parametric form for each trajectory and perform (by proxy) the identification of data generating processes. For example, suppose there exists an underlying mapping $\mathcal{W} : \mathcal{S} \to \mathcal{D}$ which associates trajectories to data generating processes. We note that such data generating processes can also be associated with certain classes of parameterization of the trajectories, i.e. some subset of a parameter space $\mathcal{K}_{\mathcal{D}_i} \subseteq \Omega_\theta$. The challenge arises in finding an lucid representation for events realized in time. Real world examples of such trajectories include, in transportation, the paths taken by different taxi drivers throughout a day; clinical events of various patients; traces of financial transactions between different agents; or series of robbery and burglary events in a call for service data stream. We note that each individual trajectory may be associated with some given property, such as being a robbery/burglary.

The *sequential data clustering problem* entails the unsupervised identification of structure within the parameters of the estimated conditional intensity function for each trajectory, and (by proxy) the identification of data generating processes. The observations (trajectories) in this case arise from the data generating processes stochastically, and may differ between realizations. Finally, we consider a common domain assumption, in that the records which we observe all come from a similar domain. For example, suppose our observations are those of video watch behavior. We should expect that each person's preferences are different, but the dynamics of all the trajectories should still be in essence similar.

1.2 Relation to Prior Work

In this section, we consider the present objective in the context of previous work. We structure the discussion based on a number of *themes*, where we introduce each theme individually, and then we discuss their relation to the low rank sequential embedding problem.

1.2.1 Applications

The time series and sequence clustering problems have been well explored in literature, and have numerous applications in the real world. Notable examples include:

- Click-streams and click paths when users navigate or browse a web-site [3, 4]
- Clinical and ICU records, where the patient status is of importance. This type of problem is known as *patient embedding* [5]
- Computer logs that list events preceding an incident, such as a hard disk failure or server deadlock [6]
- Protein and genomics modeling [7, 8]
- Lending records and other financial transactions [9]
- Transaction records that describe the order in which a customer adds items to a online shopping cart (from retailers/e-commerce)
- Call-for-service data, including police, fire, and ambulance services

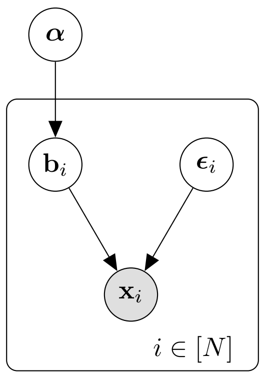


Figure 1.1: A generative model for our observations given in plate notation. $\mathbf{x}_i$ denote observed sequences for every $i \in [N]$. $\mathbf{b}_i$ denote the governing procedures with which we seek to recover parameters for, which in turn is drawn from $\boldsymbol{\alpha}$, a latent distribution in the parameter space. Note that the individuals i are independent of each other, but arise from a common procedure $\boldsymbol{\alpha}$.

1.2.2 Mixture and Generative Modeling

A common way to create more complex models from simpler atoms is to take convex combinations of a number of underlying simple/expert distributions [10]

$$\mathbb{P}[\mathbf{x}|\boldsymbol{B}] = \sum_{k=1}^{K} \alpha_k \mathbb{P}_k[\mathbf{x}] \tag{1.1}$$

such that $\boldsymbol{\alpha} = [\alpha_k]_{k=1}^{K}$ constitute a probability vector. This is the same as to say that the data chosen is given from the generative process

$$\mathbb{P}[z \mid \boldsymbol{B}] := \mathrm{Cat}(z \mid \boldsymbol{\alpha}) \tag{1.2}$$

$$\mathbb{P}[\mathbf{x} \mid z = k, \mathbf{B}] := \mathbb{P}[\mathbf{x} \mid \mathbf{b}_k] \tag{1.3}$$

where $\mathbf{B} = \mathrm{vec}(\mathbf{a}, \{\mathbf{b}_k\}_{k=1}^{K})$ the model parameters. The story of the data is then told by first drawing latent category z, and the data is then drawn according to parameters $\mathbf{b}_k$, the chosen z. Indeed, we can imagine a situation like that illustrated in Figure 1.1, which shows observations $\mathbf{x}_i$ arising from parameters $\mathbf{b}_i$, which is in turn governed by original distribution α. However, in the most general setting, mixture modeling typically leads to a non-convex problem, which admits approximation only by methods such as Expectation Maximization (EM) (expensive and inexact) or by Variational Inference (inexact) [10, 11]. Further, Equation (1.1) although supposes the total distribution of the probability measure is a mixture of sub-distributions, the individual observations may not be mixtures of the simpler distributions (indeed the generative picture says that they arise from an already category z drawn according latent $\boldsymbol{\alpha}$). In our work, we will instead relax the mixture modeling scenario to that subspace learning, where the generating parameters are allowed to come from linear subspace. Note that subspace learning the admits the same Probabilistic Graphical Model (PGM) structure as the mixture modeling case in Figure 1.1, but with different parametric assumption on the nodes and edges. Note the independencies asserted in the PGM of Figure 1.1 states that the sequences are drawn i.i.d. of each other. Here, Nature makes two random choices, first what is the combination underlying latent factors which determining the generating dynamic of $\mathbf{b}_i$, and second, what noise is given when we realize the observable $\mathbf{x}_i$. Note also that it is also immediate that the $\mathbf{x_i}$ and $\mathbf{b}_i$ are independent of each other across observations $i \in [N]$.

1.2.3 Low Rank Matrix Recovery

We turn our discussion to the recovery of the parameter matrix $\mathbf{B}$. With the requirement to ensure $\mathbf{B}$ be rank constrained, we may interpret our problem as one of *low rank matrix recovery* [12].

The Matrix Sensing Problem A basic problem in signal processing is the recovery of recovering a signal $\mathbf{b} \in \mathbb{R}^n$ given noisy observations

$$\mathbf{y} = \mathbf{A}\mathbf{b} + \boldsymbol{\epsilon} \tag{1.4}$$

Where the matrix $\mathbf{A}$ is known as the sensing matrix through which we receive observations $\mathbf{y}$ with noise $\boldsymbol{\epsilon}$. In most applications, such as recommendation systems [13], computed tomography, and communications, the signal vectors as well as the sensing matrices are of large size. Furthermore, we are allowed only few noisy observations into the matrix. A burgeoning field of research in the past two decades has been, under sparsity and/or low rank assumptions on the signal vector, to exploit the structure for efficient signal recovery [14].

In our case, instead of observing a $\mathbf{x}$ in vector form, we instead aim to observe matrix $\mathbf{B}$, which we claim to be approximately low rank

$$\mathbf{B} \approx \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

i.e., that $\mathbf{B}$ is approximately expressible with a truncated Singular Value Decomposition (SVD) with singular values $\{\sigma_r\}_{r=1}^{R}$ and orthonormal left and right singular vectors $\{\mathbf{u}_r\}_{r=1}^{R}$, $\{\mathbf{v}_r\}_{r=1}^{R}$.

In low rank matrix *recovery*, instead of observing $\mathbf{B}$ directly, we instead observe $\mathbf{y} = \mathcal{A}(\mathbf{B}) + \boldsymbol{\eta}$, where η represents noise and $\mathcal{A} : \mathbb{R}^{M \times N} \to \mathbb{R}^{L}$ is a linear *measurement operator* that takes in matrix $\mathbf{B}$ and takes inner products against L predefined $M \times N$ matrices $\{\mathbf{A}_i\}_{i=1}^{L}$

$$y_i = \langle \mathbf{B}, \mathbf{A}_i \rangle + \eta_i = \mathrm{tr}(\mathbf{A}_i^T \mathbf{B}) + \eta_i \tag{1.5}$$

and for which has adjoint

$$\mathcal{A}^*(\mathbf{w}) = \sum_{i=1}^{L} w_i \mathbf{A}_i \tag{1.6}$$

Low-Rank Recovery and Nuclear Norm Minimization Consider noisy, indirect observations $\mathbf{y} = \mathcal{A}(\mathbf{B}) + \boldsymbol{\eta}$. In our setting, these observations correspond to sequence observations, which inform the entries of $\mathbf{B}$, the parameter matrix. With the indirect observations, we consider the program

$$\min_{\mathbf{B}} ||\mathcal{A}(\mathbf{B}) - \mathbf{y}||_2^2 \qquad \mathrm{rank}(\mathbf{B}) = R \tag{1.7}$$

Unfortunately, solving Equation 1.7 is in general NP Hard. However, by contrast, its convex relaxation

$$\min_{\mathbf{B}} ||\mathcal{A}(\mathbf{B}) - \mathbf{y}||_2^2 + \lambda ||\mathbf{B}||_* \tag{1.8}$$

is an unconstrained convex optimization program which is systematically solvable using proximal algorithms [15, 16, 12]. Namely, for all $\gamma > 0$, the solutions to Equation 1.8 obey the fixed point condition

$$\mathbf{B}_* = \operatorname{Prox}_\gamma(\mathbf{B}_* - \gamma \mathcal{A}^*(\mathcal{A}(\mathbf{B}_*) - \mathbf{y})) \tag{1.9}$$

where the *proximal operator* is defined as

$$\operatorname{Prox}_\gamma(\mathbf{B}) = \arg\min_{\mathbf{X}} ||\mathbf{X} - \mathbf{B}||_F^2 + \gamma\lambda ||\mathbf{X}||_* \tag{1.10}$$

Equation Equation 1.11 gives a method for solving Program Equation 1.8 based on the iteration

$$\mathbf{B}_{k+1} = \operatorname{Prox} D_{\gamma_k}(\mathbf{B}_k - \gamma_k \operatorname{Prox} A^*(\operatorname{Prox} A(\mathbf{B}_k) - \mathbf{y})) \tag{1.11}$$

for an appropriately chosen sequence $\{\gamma_k\}$

In the particular case of the nuclear norm, the proximal operator is given by Equation 1.10 can be found by the *Singular Value Thresholding*. Consider the Singular Value Decomposition (SVD) of $\mathbf{B}$ as

$$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^* \tag{1.12}$$

A key result is that the singular value thresholding operator is the proximity operator associated with the nuclear norm [17], i.e. that

$$\operatorname{Prox}_\tau(\mathbf{B}) := \mathbf{U}\mathcal{D}_\tau(\mathbf{\Sigma})\mathbf{V}^*, \quad \operatorname{Prox}_\tau(\mathbf{\Sigma}) = \operatorname{diag}(\{\max(0, \sigma_i - \tau)\}) \tag{1.13}$$

$$= \arg\min_{\mathbf{X}} \frac{1}{2}||\mathbf{X} - \mathbf{B}||_F^2 + \tau||\mathbf{X}||_* \tag{1.14}$$

It remains to pick $\tau = \frac{\gamma\lambda}{2}$, to give a solution to Equation 1.10. The iteration given in Equation 1.11 is tractable for small/medium sized systems. More sophisticated methods rely on more or less the same idea, see [12] for an overview. We will see in Chapter 2 how the formulation of our sequence representation via the correct choice of operator $\mathcal{A}$ (which in our case maps the observed data to our parameters) enables the application of recent advances from matrix sensing, as well as the related areas of convex optimization [14] and compressive sensing [18, 19] to our problem.

1.2.4 Sequence Modeling

Previous work on time series and sequence modeling has primarily followed three broad approaches [20].

Feature Based Classification Here, we transform a sequence into feature vector, and then apply classification methods that apply work on ordinary vector spaces [10]. For example, in the case of symbolic sequences, we can consider the n-grams, use count/presence of n-grams as features, which motivates classical methods in *topic modeling* such as Latent Dirichlet Allocation (LDA) [21]. Other methods for discrete spaces involve patterns (short sequences of similar character). In continuous valued time series, another type of extractable feature are particular patterns (time series shapelets) which can be isolated by via wavelet decomposition [22]. The key disadvantage of this type of approach is that a domain expert is required to identify and design features, and so the identification becomes highly domain sensitive. A natural extension to this idea is that of vector embedding, where we continually ingest the sequence to a recurrent and use the hidden state as features. This approach is popular in natural language settings with embedding models such as BERT and ELmo [23, 24, 25]. The drawback here lies in the context sensitivity and the method does not extend readily to general time-series setting, where the number of training data samples may be limited and highly domain dependent. For example the context sensitivity is known to cause difficulties in dealing with financial data.

Distance based Classification In this class of methods, we define a distance between two observed trajectories. For example, in the case of discrete sequences, classical methods include edit distance or longest common sequence [26] which are popular in computational biology. The extension of the edit distance idea to real valued timeseries is that of Dynamic Time Warping (DTW), which measures the measuring similarity between two temporal sequences. The observed sequences may vary in their frequency.

Another approach is to transform the sequence into a feature space. For example consider an alphabet $\mathcal{A}$, sequence x is transformed into the feature vector via transform

$$\Phi_k(x) = (\phi_a(x))_{a \in \mathcal{A}^k}$$

where $\phi_a(x)$ is the frequency of a in x then consider kernel function

$$K(x, y) = \Phi_k(x) \cdot \Phi_k(y)$$

upon which we can apply a maximum margin classifier [27].

Model based Classification The third approach is to build a model of the sequence dynamics, and then perform the clustering or classification in the space of model parameters. The simplest approach for symbolic sequences is to build to Hidden Markov Model (HMM) for the sequences [28, 29]. The methods which we propose in the subsequent Chapters are essential in the similar line, however our approach allows for the modeling of a class of dynamics much richer than the discrete setting of HMM for a large variety of sequence types. Furthermore, we do not require any explicit assumptions on the learned latent space beyond low rankness, such as to explicitly pick a number of clusters beforehand.

CHAPTER 2
PARAMETER RECOVERY IN GENERALIZED AUTOREGRESSIVE PROCESSES

This chapter discusses the recovery of dynamics for fixed interval time series. The chapter is organized as follows: we first discuss problem formulation, and then provide some illustrative numerical examples. We introduce nuclear norm constrained Least Squares (LS) estimation in the simplest case of linear autoregressive time-series in Section 2.1. We discuss the problem in the context of *low rank matrix recovery*. We then formulate the problem in the nonlinear case in Section 2.2 via monotone Variational Inequality (VI). We discuss how to generalize the previous models to Section 2.2.2 and categorical (symbolic) Section 2.2.3. We illustrate our method via simulation in Section 2.3.

Problem Formulation Suppose we observe N real-valued time series of length T in the form of $\{x_{i,t}\}$, where $t \in [T]$ and $i \in [N]$. The sequences are independent from each other across i, but have temporal dependence across t. We refer to the history of events for sequence i up to time-point t as $\mathcal{H}_{i,t} := \{x_{i,s} \mid s < t\}$. We expect the behavior at event $x_{i,t}$ to be a function of past observations

$$\mathbb{E}[x_{i,t} \mid \mathcal{H}_{i,t}] = f(\mathbf{b}_i, \mathcal{H}_{i,t}) \tag{2.1}$$

Here, $f(\mathbf{b}_i, \cdot)$ is a class of model that takes in the past history of observations and returns a prediction of the focal timepoint according to parameters $\mathbf{b}_i$ unique to the i^{th} sequence. Let the matrix $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_N \end{bmatrix} \in \mathbb{R}^{d \times N}$ be the parameters across all the sequences. Our goal is to recover a good choice of the matrix $\mathbf{B}$ without supervision. Firstly, we desire that each of the $\mathbf{b}_i$ to be as faithful to the generating dynamics of the observed data as possible. At the same time, we hope to encode a *common domain assumption* about the sequences, and leverage *side information* from the other sequences to inform the prediction of the focal sequence. We express these by a *low rank assumption* on $\mathbf{B}$. To this end, consider the rank r Singular Value Decomposition (SVD) of $\mathbf{B}$

$$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* \tag{2.2}$$

where $\mathbf{\Sigma} = \mathrm{diag}\{\sigma_k\}_{k=1}^r$ corresponds to the singular values, columns of $\mathbf{U} = [\mathbf{u}_k]_{k=1}^r \in \mathbb{R}^{d\times r}$ form an orthobasis in $\mathbb{R}^d$, and columns of $\mathbf{V}^* = [\mathbf{v}_k]_{k=1}^r \in \mathbb{R}^{r\times N}$ form an orthobasis in $\mathbb{R}^N$. The recovered columns $\mathbf{C} := \mathbf{\Sigma}\mathbf{V}^* = [\mathbf{c}_i]_{i=1}^N \in \mathbb{R}^{r\times N}$ give an r-dimensional representation for each of the N sequences. Likewise, the geometry of the subspace $\mathrm{sp}\{\mathbf{u}_k\}_{k=1}^r$ describes the *common domain* from which the generating processes of the sequences arise. Therefore, the full parameter representation for the i^{th} sequence $\mathbf{b}_i = \mathbf{U}\mathbf{c}_i$ encodes the learned dynamics $f(\mathbf{U}\mathbf{c}_i, \cdot)$.

It remains to select a class of parametric model. In this work, we suppose that the each event in the sequence has temporal dependence on up to the past d events. In this autoregressive context, we build up models of increasing sophistication according to the general framework of "generalized" generalized linear models [30]. Our approach to processing the resulting estimation problems is based on convex optimization, which leads to computationally efficient procedures, but without sacrificing of a rich class of dynamics. Indeed, specifying the form of the model via the choice of a monotone link function allows us to model a wide variety of dynamics including

1. Single channel and multichannel autoregressive time-series
2. Symbolic Sequences (language and genetics)
3. Count processes (event intensities)
4. Bernoulli processes (zero or one processes)

The main tools we will formulate our problem are VI with monotone operators [31, 32].

Because rank constrained optimization is in general an NP-Hard problem [33], to enforce the *low rank* requirement on **B**, we instead constrain our setup to a *nuclear norm ball*. The nuclear norm is given by $\|\mathbf{X}\|_* = \sum_{j=1}^r \sigma_i(\mathbf{X})$ where σ_i is the ith singular value of the matrix. The nuclear norm is the tightest convex relaxation to matrix rank [34] and leads to tractable parameter recovery and allows us to leverage a long line of work from compressive sensing and convex optimization [17, 12, 35] in our parameter recovery.

2.1 Low Rank Time-series Embedding for Linear Autoregressive Models

Suppose events $x_{i,t} \in \mathbb{R}$ obey a linear autoregressive model $f(\mathbf{b}_i, \mathcal{H}_{i,t}) := \mathbf{b}_i^T \boldsymbol{\xi_{i,t}}$, where $\boldsymbol{\xi_{i,t}} = [x_{i,t-d:t-1}, 1]$ denotes the past d observations together with a constant, such that

$$\mathbb{E}[x_{i,t}|\mathcal{H}_t] = b_{i,d+1} + \sum_{s=1}^{d} b_{i,s} x_{i,t-s} = \mathbf{b}_i^T \boldsymbol{\xi_{i,t}}. \tag{2.3}$$

To recover the parameter matrix **B**, a natural choice is to take least squares loss and write

$$\min_{\hat{\mathbf{B}} \in \mathbb{R}^{d \times N}} \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{T-d} \sum_{t=d+1}^{T} \left(x_{i,t} - \hat{\mathbf{b}}_i^T \boldsymbol{\xi_{i,t}} \right)^2 \right] \qquad \text{s.t.} \qquad \|\mathbf{B}\|_* \leq \lambda. \tag{2.4}$$

Low Rank Recovery and the Nuclear Norm Prox Setup We now discuss Equation (2.4) in the context of low rank matrix recovery [12]. We aim to recover matrix **B**, but instead of observing it directly, we receive indirect samples $\mathbf{y} \approx \mathcal{A}(\mathbf{B})$ through random linear *measurement operator* $\mathcal{A} : \mathbb{R}^{d \times N} \to \mathbb{R}^N$ such that $\mathbb{E}[\mathbf{y}|\mathcal{A}] = \mathcal{A}(\mathbf{y})$. Namely, we realize samples of form $(\mathcal{A}_t, \mathbf{y}_t)$ by selecting a time-point t from the time-series observations. The samples $\mathbf{y}_t = [x_{i,t}]_{i=1}^N \in \mathbb{R}^N$ represent the values from the present time-point across all N sequences. The sampled operator $\mathcal{A}_t$ packages together the preceding length d window of regressors across the N sequences, the dynamics $f(\cdot, \cdot)$, and adds also the corresponding noise $\boldsymbol{\epsilon}$. In particular, we have

$$\mathcal{A}_t(\mathbf{B})_i = \frac{1}{N}\mathbf{e}_i^T\mathbf{B}^T\boldsymbol{\xi}_{\boldsymbol{i,t}} + \epsilon_i \qquad i \in [N] \tag{2.5}$$

where $\mathbf{e}_i$ are the standard bases. We take these the noisy, indirect observations $\mathbf{y} \approx \mathcal{A}(\mathbf{B})$ to be drawn from from $(\mathcal{A}, \mathbf{y}) \sim P$ and consider the expected least squares loss via the stochastic program

$$\min_{\hat{\mathbf{B}}} \ell(\hat{\mathbf{B}}) := \mathbb{E}_{(\mathcal{A},\mathbf{y})\sim P}\|\mathcal{A}(\hat{\mathbf{B}}) - \mathbf{y}\|_2^2 \qquad \text{s.t.} \qquad \|\hat{\mathbf{B}}\|_* \leq \lambda. \tag{2.6}$$

Since the joint distribution P is not readily available and we have only access to the observed temporal slices of size $d+1$ running up to time T, we now build the empirical analogue to Equation (2.6) ,

$$\min_{\hat{\mathbf{B}}} \hat{\ell}(\hat{\mathbf{B}}) := \frac{1}{T-d}\sum_{t=d+1}^{T} \|\mathcal{A}_t(\hat{\mathbf{B}}) - \mathbf{y}_t\|_2^2 \qquad \text{s.t.} \qquad \|\hat{\mathbf{B}}\|_* \leq \lambda, \tag{2.7}$$

which is a Lipschitz smooth convex program on the nuclear ball of radius λ [36]. Notice that Equation (2.7) is the exactly the same as Equation (2.4) except placed in a matrix recovery context. We aim to recover the optimal $\mathbf{B}$ from the samples while accounting for the global structure. When λ is arbitrarily large, there is is no constraint on $\hat{\mathbf{B}}$ and Equation (2.7) corresponds to fitting each sequence individually with no global information. On the other extreme, forcing $\hat{\mathbf{B}}$ to be rank one constrains the model for each sequence to be multiples of each other. The intermediate values of λ correspond to various trade-offs between learning the common global structure and attention to the individual sequences.

Equation (2.7) is readily cast as a Second-Order Cone Program (SOCP) solvable via interior point methods [37]. However, as the size $\mathbf{B}$ may reach into the hundreds of thousands of decision variables, we turn our discussion instead to solutions via efficient first-order proximal algorithms [15, 16]. We will now describe the proximal setup for nuclear norm minimization in the context of time series embedding [35]. To illustrate, consider the Mirror Descent (MD) procedure

$$\mathbf{B}_{k+1} = \mathrm{Prox}_{\mathbf{B}_k}(\gamma_k \nabla_{\mathbf{B}_k}[\ell(\mathbf{B}_k)]) \qquad \mathbf{B}_0 \in \{\mathbf{X} \mid \|\mathbf{X}\|_* \leq \lambda\} \tag{2.8}$$

for an appropriately chosen sequence of steps $\{\gamma_k\}$. The solution at step k is given by the aggregate $\tilde{\mathbf{B}}_k = (\sum_{\tau=1}^{k} \gamma_\tau)^{-1} \sum_{\tau=1}^{k} \gamma_\tau \mathbf{B}_\tau$. We take *prox-mapping*

$$\mathrm{Prox}_{\mathbf{Z}}(\mathbf{X}) = \underset{\|\mathbf{Y}\|_* \leq \lambda}{\arg\min}\, \omega(\mathbf{Y}) + \langle \mathbf{X} - \nabla_{\mathbf{Z}}[\omega(\mathbf{Z})], \mathbf{Y} \rangle$$

and *Bregman divergence* ω to be the ones associated with the nuclear ball. In particular, we can compute the nuclear-norm prox mapping by Singular Value Thresholding (SVT), successively eliminating the small singular values [17]. Namely, with $m := \min(d, N)$; $q := \frac{1}{2\ln(2m)}$; $\alpha := \frac{4\sqrt{e}\log(2m)}{2^q(1+q)}$; $\mathbf{M} = \mathbf{U}\,\mathrm{diag}\{\sigma_i\}_{i=1}^{m}\mathbf{V}^*$ we have Bregman divergence and its subgradient as

$$\omega(\mathbf{M}) = \alpha \sum_{i=1}^{m} \sigma_i^{1+q} \implies \nabla_{\mathbf{M}}[\omega(\mathbf{M})] = \sum_{i=1}^{m} [c(1+q)\sigma_i^q]\mathbf{u}_i \mathbf{v}_i^*.$$

To compute the prox-mapping, consider the SVD of $\mathbf{X} - \nabla_{\mathbf{Z}}[\omega(\mathbf{Z})] = \mathbf{P}\,\mathrm{diag}\{\sigma_k\}_{k=1}^{r}\mathbf{Q}^*$. The optimal value of the linear program

$$\mathbf{t} = \min_{t \in \mathbb{R}^n} \{\sum_{i=1}^{m} \frac{1}{2} t_j^2 - \sigma_i t_i \mid \mathbf{t} \geq 0, \sum_{j=1}^{m} t_j \leq \lambda\} \tag{2.9}$$

gives $\mathrm{Prox}_{\mathbf{X}}(\mathbf{B}) = \mathbf{U}\,\mathrm{diag}\{-\mathbf{t}\}_{t=1}^{m}\mathbf{V}^*$ as the prox-mapping associated with the nuclear ball [35]. Notice that Linear Equation (2.9) can be solved in time $\mathcal{O}(m)$. The time for each iteration of Mirror Equation (2.8) is dominated by the cost of the SVD and the cost to compute the gradients.

2.2 Convex Nonlinear Autoregression

We extend our discussion now to the nonlinear case. Suppose the events $x_{i,t} \in \mathbb{R}$ obey the model $f(\mathbf{b}_i, \mathcal{H}_{i,t}) := \eta(\mathbf{b}_i^T \boldsymbol{\xi_{i,t}})$, where $\boldsymbol{\xi_{i,t}} = [x_{i,t-d:t-1}, 1]$ encodes the past d observations with a constant term and $\eta : \mathbb{R} \to \mathbb{R}$ is a fixed *monotone link function*

$$\mathbb{E}[x_{i,t}|\mathcal{H}_t] = \eta(b_{i,d+1} + \sum_{s=1}^{d} b_{i,s} x_{i,t-s}) = \eta(\mathbf{b}_i^T \boldsymbol{\xi_{i,t}}). \tag{2.10}$$

Our goal is to form a rank constrained stochastic estimate to $\mathbf{B}$. However, with arbitrary monotone link function, the Least Squares approach outlined in Equation (2.4) Equation (2.7) loses convexity and computational tractability in general. Likewise, Maximum Likelihood Estimation (MLE) based approaches also becomes computationally difficult. By contrast we shall cast the parameter recovery problem into a monotone VI formulation, the most general type of convex program for which there are known methods to efficiently find solutions of high accuracy [30, 31, 32].

Preliminaries on Monotone VI A *monotone vector field* on $\mathbb{R}^m$ with modulus of convexity β is a vector field $G : \mathbb{R}^m \to \mathbb{R}^m$ such that

$$\langle G(\mathbf{x}) - G(\mathbf{x}'), \mathbf{x} - \mathbf{x}' \rangle \geq \beta \|\mathbf{x} - \mathbf{x}'\| \qquad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^m$$

when $\beta > 0$, G is *strongly monotone*. For some convex compact set $\mathcal{X} \subseteq \mathbb{R}^m$, a point $\mathbf{x}^*$ is a *weak solution* to the VI associated with $(G, \mathcal{X})$ if for all $\mathbf{x} \in \mathcal{X}$ we have $\langle G(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle \geq 0$. If F is strongly monotone and a weak solution exists, then the solution is unique. When $\langle G(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle \geq 0$ for all $\mathbf{x} \in \mathcal{X}$, we term $\mathbf{x}^*$ a *strong solution* to the VI. When G is continuous on $\mathcal{X}$ all strong solutions are weak solutions and vice versa.

2.2.1 Monotone VI for Nonlinear Parameter Recovery

We now turn our attention to the construction of a Monotone VI which has as its root optimal parameters corresponding to Equation (2.10). We will use the same operator $\mathcal{A}$ from the linear case from Equation (2.5) together with its associated adjoint $\mathcal{A}^*$. We consider the accompanying noisy observations $\mathbf{y}$ such that in expectation $\mathbb{E}[\mathbf{y}|\mathcal{A}] = \eta(\mathcal{A}(\mathbf{B}))$ and the link function η acts entry-wise. Consider now the vector field on the space of matrices

$$\Psi(\hat{\mathbf{B}}) = \mathbb{E}_{(\mathcal{A},\mathbf{y})\sim P}[\mathcal{A}^*(\eta(\mathcal{A}(\hat{\mathbf{B}})) - \mathbf{y})] : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N} \tag{2.11}$$

and notice that the matrix $\mathbf{B}$ of true generating parameters is a zero of Ψ,

$$\begin{aligned}\Psi(\mathbf{B}) &= \mathbb{E}_{(\mathcal{A},\mathbf{y})\sim P}[\mathcal{A}^*(\eta(\mathcal{A}(\mathbf{B})) - \mathbf{y})] = \mathbb{E}_{(\mathcal{A},\mathbf{y})\sim P}[\mathcal{A}^*(\eta(\mathcal{A}(\mathbf{B}))) - \mathcal{A}^*(\mathbf{y})] \\ &= \mathbb{E}_{(\mathcal{A},\mathbf{y})\sim P}[\mathcal{A}^*(\eta(\mathcal{A}(\mathbf{B}))) - \mathcal{A}^*(\mathbb{E}[\mathbf{y} \mid \mathcal{A}])] \\ &= \mathbb{E}_{(\mathcal{A},\mathbf{y})\sim P}[\mathcal{A}^*(\eta(\mathcal{A}(\mathbf{B}))) - \mathcal{A}^*(\eta(\mathcal{A}(\mathbf{B})))] = 0.\end{aligned}$$

Since we do not know beforehand what is the joint distribution P, we take solutions to the empirical version of the VI that takes slices from the time-series

$$\hat{\Psi}(\hat{\mathbf{B}}) = \frac{1}{T-d}\sum_{t=d}^{T}[\mathcal{A}_t^*(\eta(\mathcal{A}_t(\hat{\mathbf{B}}))) - \mathcal{A}_k^*(y_k)] = \frac{1}{T-d}\sum_{t=d+1}^{T}\mathcal{A}_t^*[\eta(\mathcal{A}_t(\hat{\mathbf{B}})) - \mathbf{y}_t]. \tag{2.12}$$

At each time window t, $\mathbf{y}_t$ represents the sequence observations. $\mathcal{A}_t$ then takes in as input the matrix B parameter vector of size $\mathbb{R}^{dN}$ and uses the data from the previous d time-points to output a prediction for $\mathbf{y}_t$. Analogous to Equation (2.7), the VI assosiated with Equation (2.12) likewise admits solutions by MD. To illustrate, consider the recurrence:

$$\mathbf{B}_{k+1} = \mathrm{Prox}_{\mathbf{B}_k}(\gamma_k F(\hat{\mathbf{B}}_k)) \qquad \mathbf{B}_0 \in \{\mathbf{X} \mid \|\mathbf{X}\|_* \leq \lambda\} \tag{2.13}$$

with step sizes $\{\gamma_k\}$. The aggregate solution at step k is given by $\tilde{\mathbf{B}}_k = (\sum_{\tau=1}^{k} \gamma_\tau)^{-1} \sum_{\tau=1}^{t} \gamma_\tau \mathbf{B}_\tau$ [38]. Note if $\eta := \mathbf{Id}$, the identity function, then the VI corresponds exactly to the *gradient field* of Equation (2.7). In which case $F(\mathbf{X}) = \nabla_{\mathbf{X}}[\ell(\mathbf{X})]$ and the MD procedure for VI and LS are the same.

2.2.2 Monotone VI: Vector Case

We now suppose the observed sequence constitutes an order d nonlinear vector time-series with C channels containing vector-valued samples $\mathbf{x}_{i,t} \in \mathbb{R}^C$ that obey the following

$$\mathbb{E}[\mathbf{x}_{i,t} \mid \mathcal{H}_t] = \eta(\mathbf{R}_i \boldsymbol{\xi_{i,t}}) \tag{2.14}$$

where $\boldsymbol{\xi_{i,t}} = \mathrm{vec}(1, \{\mathbf{x}_{i,t-s}\}_{s=1}^{d}) \in \mathbb{R}^{dc+1}$ represents the values from the d preceding observations from $\mathbf{x}_{i,t}$ along with a constant placed into one vector. The matrix $\mathbf{R}_i \in \mathbb{R}^{C \times Cd+1}$ then produces a prediction for each channel of the current observation $\mathbf{x}_{i,t}$. We allow $\mathbf{b}_i = \mathrm{vec}(\mathbf{R}_i) \in \mathbb{R}^{C^2d+C}$ to be the parameters corresponding to the ith sequence arranged as a vector, and allow $\mathbf{B} \in \mathbb{R}^{C^2d+C \times N}$ to be the $\mathbf{b}_i$ arranged column wise. The situation here is essentially that of parameterizing N perceptrons with a monotone activation function. Each perceptron takes in the multichannel data of the past d observations and produces an estimate for the current value [39]. We take $\mathbf{y}_t := \mathrm{vec}([\mathbf{x}_{i,t}]_{i=1}^{N})$, and the measurement operator $\mathcal{A}_t$ to be

$$\mathcal{A}_t(\hat{\mathbf{B}}) := \mathrm{vec}([\mathbf{R}_i \boldsymbol{\xi_{i,t}}]_{i=1}^{N}) : \mathbb{R}^{C^2d+C \times N} \to \mathbb{R}^{CN} \tag{2.15}$$

which together with adjoint $\mathcal{A}_t^*$ and montone link function $\mathbb{R}^{cN} \to \mathbb{R}^{cN}$ admits the same formulation and solution methods as the single channel case in Equation (2.12).

2.2.3 Symbolic Sequences

As a special case of Equation (2.14) in Section 2.2.2, consider now that each channel represents the probability of emitting a token taken from syllabary $\{s_c\}_{c=1}^C$ of size C. Then each $\mathbf{x}_{i,t}$ represents a probability vector $\sum_c^C x_{i,t,c} = 1$ and we have $\mathbb{E}[x_{i,t,c}|\mathcal{H}_t] = \mathbb{P}[x_{i,t,c} = s_c]$. We take the sequence-wise monotone softmax activation function

$$\sigma(\mathbf{y}) = \text{vec}([\|\exp(\mathbf{y}^{(i)})\|_1^{-1} \exp(\mathbf{y}^{(i)})]_{i=1}^N),$$

where $\mathbf{y}^{(i)}$ corresponds to values from the i^{th} sequence.

2.3 Numerical Illustration

In this section I detail a numerical illustration of our method using generated autoregressive data. Here, we draw $k = 3$ classes of autoregressive sequences of order $d = 15$ each containing $N_k = 300$ samples, and mix their sequences together. Each sequence contains $T = 250$ time-points. Each type of sequence arises form a linear time-series with preturbed coefficents namely:

1. Linear time series with exponentially decaying parameters plus Gaussian noise.
2. Linear time series with uniformly drawn parameters, perturbed by a random vector (shared across all sequences) times a constant factor
3. Linear time series with exponentially decaying parameters and the $d/3$ most recent values perturbed by Gaussian noise.

We recover the baseline parameters via MD on the associated VI. We denote the true parameters of the sequences $\mathbf{B}$, and their estimates as $\hat{\mathbf{B}}$, and we allow $\mathbf{B}^{(k)}$ (analogously $\hat{\mathbf{B}}^{(k)}$) to be the submatricies corresponding to the parameters for the k^{th} cluster. Let us first examine the relative error in recovering the matrix $\|B - \hat{H}\|_F / \|B\|_F$ for varying levels of λ

in Figure 2.1. We show the values of λ on a logarithmic scale, and show also vertical lines representing the spectrum (singular values) of $\mathbf{B}$ and $\hat{\mathbf{B}}$ (recovered with no rank constraint) respectively. The number of lines preceding (succeeding) a certain point corresponds to the number of eigenvalues which are eliminated (preserved) at this level of constraint on the nuclear norm. We see that eliminating the smallest singular values allows us to leverage side information shared across the sequences to build an improved estimate across all trajectories namely between the small eigenvalues eliminated between the 2^{-3} and 2^{-1}. We achieve optimal reconstruction error close to 2^{-1} which leaves only three principal eigenvalues. Notice that we can eliminate all but the first two eigenvalues and still have reconstruction performance which beats that of estimating all of the sequences independently without eliminating small singular values. The behavior can be seen as well for the sub-matrices $\mathbf{B}^{(k)}$ corresponding to the clusters as well as shown in Figure 2.2. Note all the sub-matrices display similar reconstruction performance, with the leveraging of the joint structure resulting in improved estimation performance. Furthermore, note the similarity even when the spectral distribution is essentially in the case of Gaussian pretrubation in Figure 2.2a (which shows a single large value, and many small ones), and Figure 2.2b and Figure 2.2c, where the original structure in inherently low rank. We note that we can eliminate most of the small singular values in the reconstructed sub-matrices without sacrificing reconstruction performance.

To show how the nuclear constraint eliminates small singular values, we turn our attention to the spectra of the recovered (sub)-matrices themselves. In the combined Figure 2.3, we see indeed with sufficient choice of λ we have the constraints dragging the the majority of singular values to zero, leading to a low rank structure of the reconstruction. The left panel shows the spectra for the original matrix **B**, and the recovery without the nuclear norm $\hat{\mathbf{B}}$, which we see oftentimes captures also the noise. We see a similar story in Figure 2.4, where we the low structure in the original matrices. Without the nuclear constraints we see that we cannot the spectra, but with correct choice of λ, which corresponds to a regime of

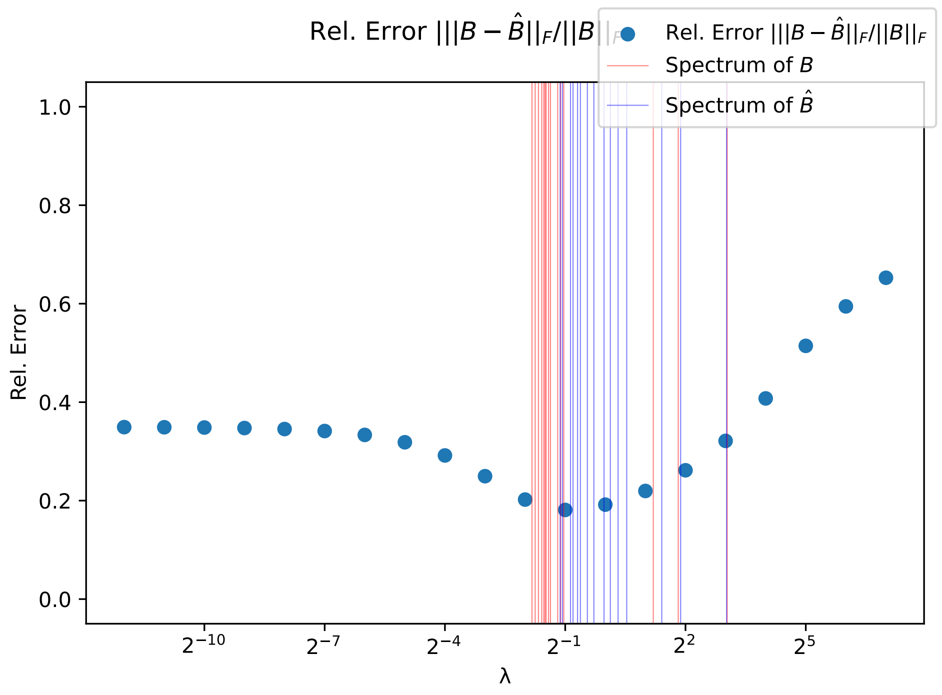
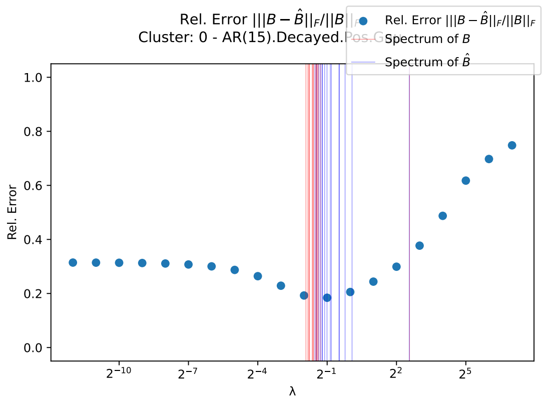


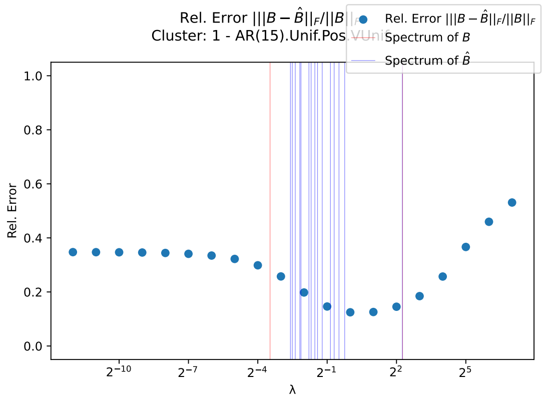
Figure 2.1: $\|\mathbf{B} - \hat{\mathbf{B}}\|_F / \|\mathbf{B}\|_F$ for optimal parameter recovery of three types of autoregressive sequence.

low reconstruction error, we see that we also recover according spectra similar to the true signal.

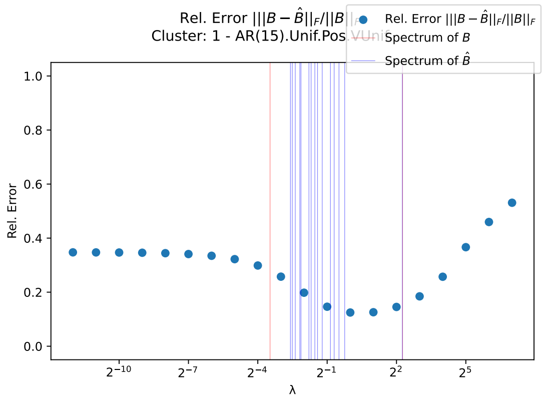
To further demonstrate, Figure 2.5 shows the 15 recovered parameters for 10 of each of the sequences from each of the 3 classes. We notice the increase similarity in the structure as the value of λ is increased. Finally to show the embedding space itself, Figure 2.6 shows the PCA projection of the right singular values space corresponding to the learned low rank representation according to different values of λ. We show each of the groupings in color, and note that the embedding starts out with large variance, and with larger values starts to form compact and separated clusters. Eventually, tending to the extreme case where λ very large, we see the representation collapse to a line.



(a) Exponentially decaying parameters plus Gaussian noise



(b) Uniformly drawn parameters, perturbed by a random vector (shared across all sequences) times a constant factor



(c) Exponentially decaying parameters and the $d/3$ most recent values perturbed by Gaussian noise.

Figure 2.2: $\|\mathbf{B}^{(k)} - \hat{\mathbf{B}}^{(k)}\|_F / \|\mathbf{B}^{(k)}\|_F$ for optimal parameter recovery of three types of autoregressive sequence across sub-matrices.

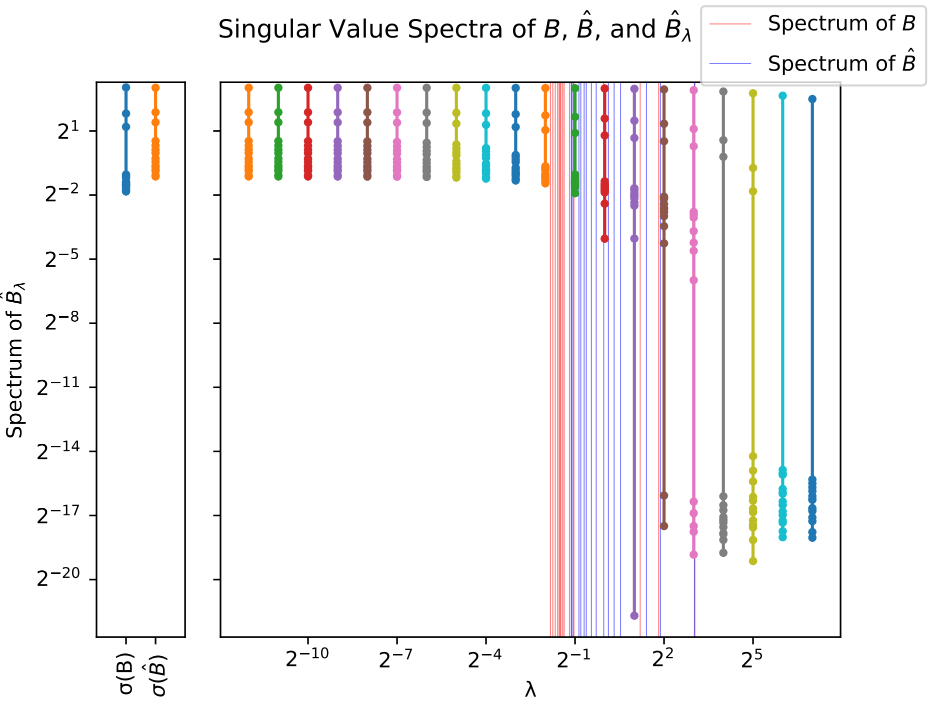
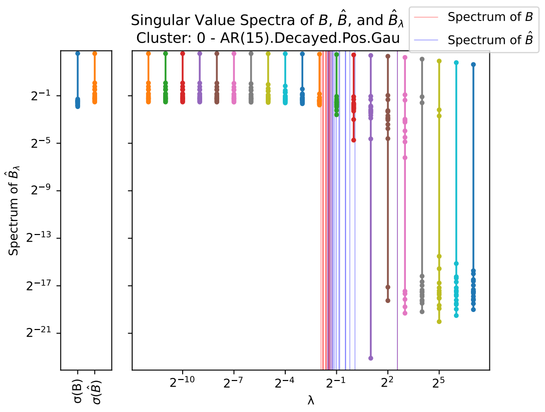
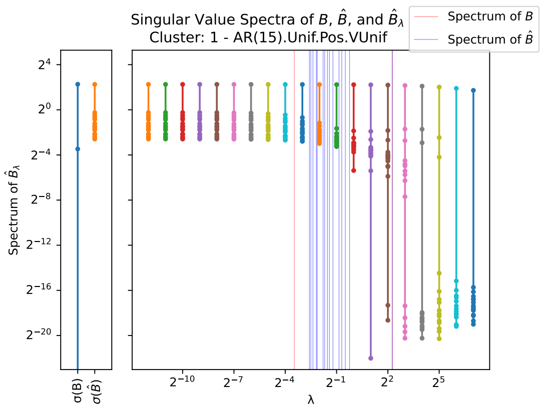


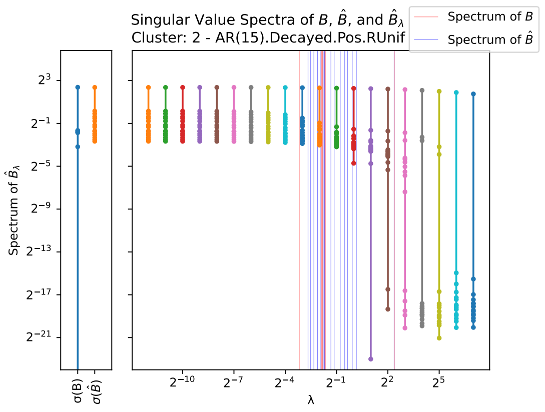
Figure 2.3: Singular Value Spectra of $\mathbf{B}$, $\hat{\mathbf{B}}$, and $\hat{\mathbf{B}}_\lambda$.



(a) Exponentially decaying parameters plus Gaussian noise



(b) Uniformly drawn parameters, perturbed by a random vector (shared across all sequences) times a constant factor.



(c) Exponentially decaying parameters and the $d/3$ most recent values perturbed by Gaussian noise.

Figure 2.4: Singular Value Spectra of $\mathbf{B}^{(k)}$, $\hat{\mathbf{B}}^{(k)}$, and $\hat{\mathbf{B}}_\lambda^{(k)}$ across sub-matrices.

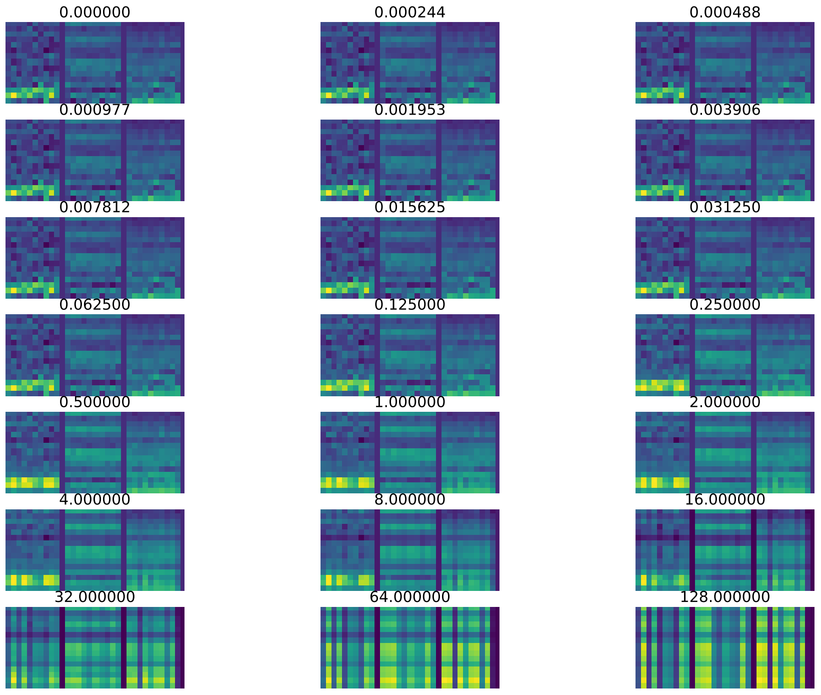


Figure 2.5: Heat-maps showing the recovered parameters for the three classes of data with different levels of regularization.

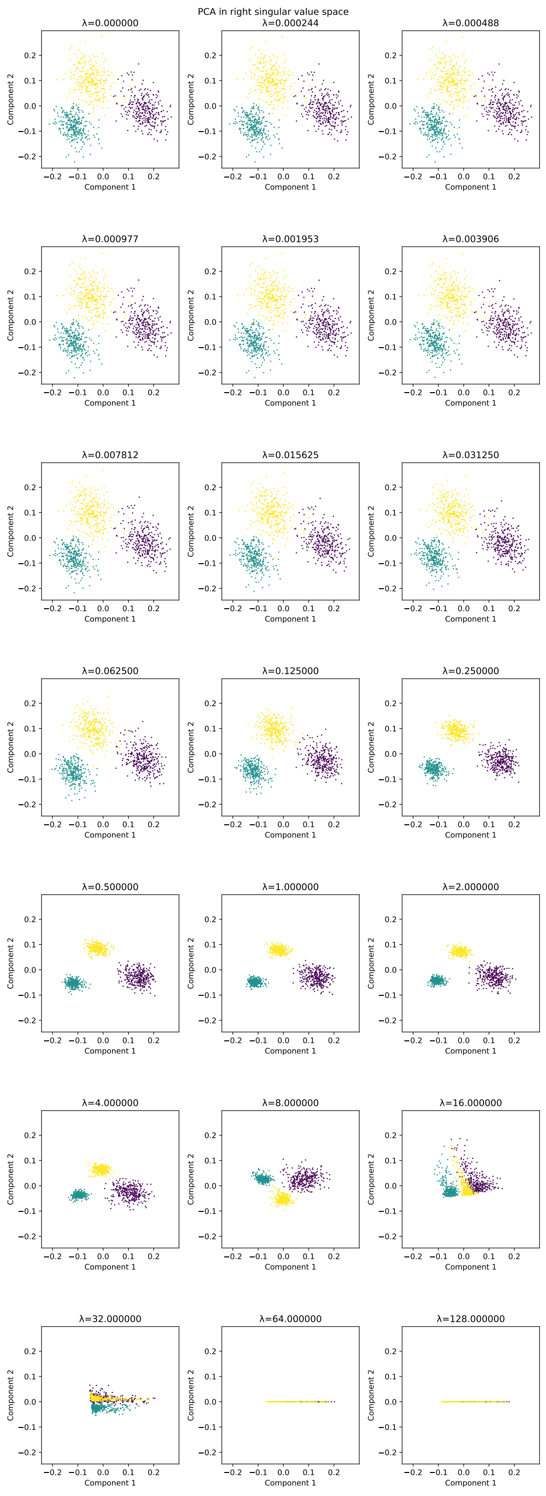


Figure 2.6: PCA projection of right singular value space corresponding to embedding with different choices of λ. The colors show the original class to which the sequences belonged.

CHAPTER 3
RECOVERY OF TEMPORAL POINT PROCESSES

3.1 Problem Setting

This chapter discusses the recovery of dynamics for sequences of discrete events on continuous time frames. These sequences are well modeled by Temporal Point Processes (TPP) that capture the underlying dynamics behind the event sequences.

Examples of data of this form include neuron action potentials in the brain [40], social behavior [41, 42], and a wide array of healthcare and epidemiological settings [43]. The underlying processes which generate event sequences are well modeled by TPP and the spatial extension in Spatio-Temporal Point Processes (STPP) [44]. These models not only capture temporal information, but may additionally incorporate spatial and categorical distributions of the data (marks) to effectively represent excitation and inhibition effects.

At the same time, a lingering challenge in working with event sequence data in a variety of downstream tasks is the intrinsic high dimensionality and nonuniformity of event sequence data, which precludes the use of popular clustering and regression models that suppose the the data lie in a dense low dimensional embedding/latent space [45].

Although there is a burgeoning point processes literature [46, 1], there exists a notable gap concerning the creation of high-quality embeddings from point processes models. Such embeddings would allow for the meaningful application of well established methods of clustering, classification, and anomaly detection to point processes data.

For instance, the most well studied downstream task, often motivated by sociological contexts, is clustering different event sequences into meaningful groupings (*sequence clustering*) [47]. Real-world examples include identifying similar users based on video-watching behavior [48], grouping together similar types of patents based on clinical events

[49], and tracing financial transaction behavior between different agents [9]. In practical terms, identification of different *clusters of behavior* enables more effective and tailored service for different customers, communities, and patients.

To solve the embedding problem, we treat each event sequence in a series of sequences as a realization of a point processes, and propose an effective model-based approach that jointly learns

1. Sequence embeddings for each observed trajectory

2. A finite set of basis functions within which the embeddings exist and which capture the generational dynamics within a given problem domain.

For each observed sequence, we express the influence kernel — which underlies the conditional intensity of the process generating the event sequence — by a particular combination within the span of the learned set of basis functions. The function space given by the span of the learned basis functions is taken as the embedding space of event sequences. We motivate this representation from the truncated Singular Value Decomposition (SVD) for kernel functions [50], which addresses the desire for a low-rank representation for event sequences. This selection of basis functions and coefficients for each sequence is likewise related the problem of dictionary learning [51].

The chosen formulation addresses the fact that many data-driven learning procedures for point processes from previous literature either make the assumption that

1. the observed sequences arise from the *same* underlying stochastic processes, or

2. the observed sequences arise from a mixture of expert point processes (for which we wish to obtain both the mixture coefficients and parameter estimates for each expert)

. However, in real data, it is not realistic to expect that even similar sequences have the identical kernels. Likewise, the setting of item 2 is limited by the fact there is an explicit assumption of a mixture of point processes from a number of expert functions. Furthermore, learning parameters is often only tractable by Expectation Maximization (EM), which greatly increases the computational cost of estimation. By contrast, the model fitting using our representation is tractable via direct optimization of the likelihood function.

3.2 Related work

Embedding for fixed-length vectors is a well studied problem, wherein the key idea lies in compressing originally sparse high dimensional data into low dimension space via feature projection, for example via via auto-encoding [52]. Embedding for sequential data like gene sequences or language is also well studied. Techniques include reconstruction [53], learning an underlying model like a Markov network [54], or learning a hidden space via recurrent or transformer methods [23], which may be general or adapted to a specific problem domain [5]. For fixed-interval time series, transformer and basis function based-methods are also popular [55], but can also struggle with learning global features in the time-series.

However, these methods are not tractable for event sequence data because of the non-uniform temporal dependency, heterogeneous sequence lengths, and non-stationarity. That said, there is an extensive concurrent body of work representing event processes data, starting from the seminal work of [56]. A key modeling challenge in this area is to find a representation capable of capturing a full range of underlying generative behaviors while maintaining model efficiency and identifiability. To tackle this challenge, we adopt a promising line of work that combines statistical models with neural network representations, allowing us to harness both interpretability in the former and expressive power in the latter [57].

In terms of the deep representation of point processes themselves, we adopt the approach in the line with the first of two main streams:

1. Parameterization of the influence kernel (rather than the intensity function) by neural networks [58, 59]

2. Recurrent and transformer based approaches wherein events are continuously in-

gested to produce a hidden state [60, 61, 62]. This hidden state may be interpreted as a an embedding, but this space is be highly nonlinear, and faces difficulty in capturing non-stationary behavior, and, due to the locality property of the hidden state, be biased towards the later elements in the sequence.

With regards to the *sequence clustering* problem as a specific downstream task of interest, there exist numerous mixture models, which suppose that the data arises out of a mixture of experts/topics [63, 47]. However, model-fitting in these cases is frequently resource intensive, relying on Expectation Maximization (EM). A final related line of work seeks to define seeks to define point process distances [64, 65], but the distances do not yield an underlying space, and do not consider the underlying generative processes. Furthermore, computing a pairwise distance between all of each processes requires a number of evaluations in the square of the number of sequences.

3.3 Organization

The remaining sections are organized as follows: Section 3.4 discusses a framework of jointly learning

1. Representations (embeddings) for each observed trajectory
2. A finite set of basis functions which form the space in which the representations lie

which characterize the generational dynamics within a given domain given a collection of event sequences. We give procedures to efficiently learn and recover the embedding by Maximum Likelihood Estimation (MLE) at no additional cost compared to without relaxing the assumption that all sequences arise from the same influence kernel. Section 3.5 shows via experiments on synthetic and real-world data, improved or comparable performance over recurrent methods, but with greater computational efficiency and interpretability.

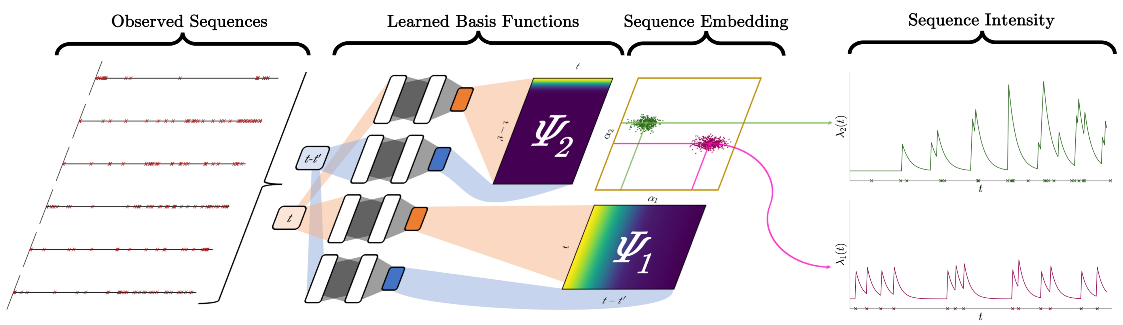


Figure 3.1: Graphical illustration of our joint basis function and embedding learning method (SSEM) from a collection of observed sequences $\mathbf{X}$ as described in Equation 3.9. We fix the rank of temporal factorization at $R = 2$, and learn a two sets of basis functions $\Psi_1 = \psi_1 \otimes \phi_1$ and $\Psi_2 = \psi_2 \otimes \phi_2$ in event times t and displacements t'. The gold box depicts learned sequence embeddings for $n = 500$ sequences showing two distinct clusters. Sample realizations assigned close to the centroids of the clusters are given on the far right.

3.4 Methods

3.4.1 Background

Temporal Point Processes (TPP) TPPs are popular models to sequences of random events which occur in continuous time [44, 2, 1]. A given TPP is characterized completely by its *conditional intensity* of an TPP, which is dependent on its in a realization $\mathcal{H}_t = \{t_j | t_j < t\}$ for collection of events $\{t_j\}$ such that

$$\lambda(t|\mathcal{H}_t) = \lim_{\Delta t \to 0} \frac{\mathbb{E}[\mathbb{N}([t, t+\Delta t))|\mathcal{H}_t]}{|\Delta t|} \tag{3.1}$$

Where $\mathbb{N}(S) = |\mathcal{H}_T \cap S|$ is a right-continuous counting measure on the number of events in $[t, t + \Delta t]$.

Parameterization of TPPs In terms of capturing the conditional intensity function, the classical temporal Hawkes process [56] supposes that past influence is linearly additive, and gives a conditional intensity of the form

$$\lambda_k(t', t) = \mu + \sum_{x' \in \mathcal{H}_t} k(t', t) \tag{3.2}$$

where $\mu > 0$ is some background intensity and $k : [0, T) \times [0, T) \to \mathbb{R}$ is a fixed influence kernel function that captures the excitation effect of past events on the likelihood of event occurrence at the current time-point. In the Hawkes processes case, it is common to assume that the kernel function k is *stationary*, i.e. only dependent on $t - t'$.

The key idea of expressing the influence by a kernel function is extended to the more general TPP case, where we parameterize the conditional intensity of t via kernel function k as follows:

$$\lambda_k(t) = \int_{t' \in \mathcal{X}_t} k(t', t) \mathrm{d}\mathbb{N}(t') \tag{3.3}$$

where the integral over the counting measure $\mathbb{N}$ of a function $f : \mathcal{X} \to \mathbb{R}$ is defined as:

$$\int_S f(t) \mathrm{d}\mathbb{N}(t) = \sum_{t_i \in \mathcal{H}_T \cap S} f(t_i) \tag{3.4}$$

And for which we have the log-likelihood of observing $\mathcal{H}$ from a given processes conditioned on the history $\mathcal{H}$

$$\log \ell(\mathcal{H}) = \sum_{i=1}^{n} \log \lambda_k(t_i) - \int_0^T \lambda_k(t) dt \tag{3.5}$$

Neural Point Processes parameterize the intensity function by leveraging the representational power of neural networks. The key idea is that input observations $\mathbf{x}$ which contain the temporal and spatial/mark information are continuously fed into a recurrent network f in order to produce hidden state vectors $\mathbf{h}_{i+1} = f(\mathbf{h}_i, \mathbf{x}_i)$ [60]. The conditional intensity between time-points t_i and t_{i+1} is then characterized by the relationship of the current hidden state and the time $\lambda(t) = g(t, \mathbf{h}_i)$. Extensions to this general idea include the use of LSTM instead of recurrent units [62], as well as attention based models like Transformer

Hawkes [61].

Concurrent to the recurrent approach is modeling the influence kernel function directly [66], e.g. by representation via spectral decomposition [67]. For example, unified point process models using low-rank deep non-stationary kernels are proposed in [59, 58]. The key idea is to take a finite rank approximation the influence kernel k as the composition of several basis functions. Namely, we can parameterize the kernel input as $(t', t-t')$ and write the kernel function as

$$k(t', t-t') = \sum_{l=1}^{L} \alpha_l \psi_l(t') \phi_l(t-t') \tag{3.6}$$

In this case, $\forall l \in [L], \psi_l, \phi_l : [0,T) \to \mathbb{R}$ represents a series of pairs of temporal basis functions that characterize the temporal influence at time-point t' and the effect across $t-t'$. The motivation for such a decomposition arises from the rank constrained spectral decomposition of a general kernel functions [68]. In this work, we extend the deep kernel learning procedure to the following problem setting.

3.4.2 Sequence Embedding

We consider a set of event sequences $\mathbf{X} = \{\mathbf{x}\}_{i=1}^{n}$ of form $\mathbf{x} = \{t_{i,j}\}_{j=1}^{|\mathbf{x}|}$. Each individual event $t_{i,j}$ from the i-th event sequence occurs on $[0,T)$, a continuous time horizon. Note also that the length of individual trajectories $|\mathcal{X}_i|$ may be variable. We aim to learn an embedding function $E : \mathcal{T} \to \mathcal{A} \subset \mathbb{R}^m$ where $\mathcal{T}$ is the space of plausible event sequences for the problem domain where we draw realizations $\mathbf{X} \subset \mathcal{T}$, and $\mathcal{A}$ is embedding space.

3.4.3 Joint Representation of Basis Functions and Embedding for Multiple Sequences

We extend the parameterization given in Equation 3.6 to the problem setting as given in Section 3.4.2. For each sequence $\mathbf{X} = \{\mathbf{x}\}_{i=1}^{n}$, we suppose the sequence has an underlying TPP having corresponding conditional intensities $\{\lambda_i\}_{i=1}^{n}$ that induce kernel functions

$\{k_i\}_{i=1}^n$ and baseline intensities $\mathbf{M} = \{\mu_i\}_{i=1}^n$, since we expect the sequences to arise from heterogeneous sources.

Our goal is to jointly learn a representation of the observed event sequences $\mathbf{X}$

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_n \end{bmatrix} \in \mathbb{R}^{n \times (L+1)},$$

where each of the $\mathbf{b}_i$ is in correspondence with sequence $\mathbf{X}_i$, along with a set of R temporal basis functions $\mathbf{\Psi} = [\psi_i]_{r=1}^R$, R temporal displacements basis function $\mathbf{\Phi} = [\phi_r]_{r=1}^R$, and baseline intensities $\mathbf{M}$. Each of the basis functions are individually represented by fully connected feed-forward deep neural networks. We express each kernel function as

$$k_i(t', t - t') = \sum_{l=1}^{L} \alpha_{il} \psi_l(t') \phi_l(t - t') \tag{3.7}$$

$$= (\text{diag}(\mathbf{b}_i) \mathbf{\Psi} \otimes \mathbf{\Phi})(t', t - t') \tag{3.8}$$

Where $\otimes$ is the exterior product, and where we regularize the program with factor $\beta \in \mathbb{R}_{\geq 0}$:

$$\underset{\mathbf{M},\mathbf{B},\mathbf{\Phi},\mathbf{\Psi}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{X}_i, \lambda_i) + \beta ||\mathbf{b}_i||_2^2 \tag{3.9}$$

$$\forall i \in [n], t \in T, \lambda_i(t) \geq 0 \tag{3.10}$$

We take the non-negative conditional intensity in terms of the kernel functions to be

$$\lambda_i(t|\mathcal{H}_t) = \mu_i + \sum_{(t') \in \mathcal{H}_t} k_i(t', t) \tag{3.11}$$

and take the loss $\ell(\mathbf{x}, \lambda_i)$ between each of the sequences and their representation as the negative log-likelihood:

$$\ell(\mathbf{x}, \lambda_i) = -\sum_{j=1}^{|\mathbf{X}_i|} \log(\lambda_i(t_{i,j})) + \int_{t \in [0,T)} \lambda_i(t) \mathrm{d}t \tag{3.12}$$

A high level illustration of our method when considering $L = 2$ is given in Figure 3.1. We note that the parameterization given in Equation 3.9, frames our question as a Dictionary Learning problem, where we learn the representation of the input data as a linear combination of the basic elements (the bases), in conjunction with the basis functions themselves. In this sense, the embedding of the input data $\mathbf{X}_i$ is given by the kernel representation prescribed by the columns $\mathbf{b}_i$, which corresponds to a plausible generating processes for $\mathbf{X}_i$. To this end, we also introduce the regularization term with hyper-parameter β to ensure the identifiability of a solution and bound the magnitude of **B** in the Frobenius sense.

3.4.4 Neural Kernel and Basis Function Learning

To enable the expression of broadest range of possible dynamics, we use a feed-forward neural network to represent each of the basis functions, harnessing their universal approximation power [69]. We use a separate neural network for each basis in $\mathbf{\Psi} \otimes \mathbf{\Phi}$, and so for a kernel with temporal rank L we require $2L$ separate neural networks.

For our experiments Section section 3.5 in particular, we represent each basis function using a feed-forward network having two hidden layers for each basis function each containing 64 hidden nodes with `ReLU` activations [70].

We do not place restrictions on the form of the basis functions themselves, and they can have positive or negative contributions corresponding to inhibiting and inducing effects. The only restriction placed on the optimization is that conditional intensity λ be non-negative across the temporal/spatial horizon. There are a number of ways to address this constraint, for example via a penalty function (log-barrier) during training [59], or use a nonlinear positive activation function on the output of the basis functions [60, 58].

3.4.5 Log-Liklihood Computation and Optimization

Since the log-liklihood in Equation 3.12 are not directly tractable, we formulate an approximate solution to the maximum likelihood via discrete approximation [71].

$$\ell(\mathbf{X}_i, \boldsymbol{\Theta}) = -\mathrm{LS}(\mathbf{X}_i, \boldsymbol{\Theta}) + \mathrm{I}(\mathbf{X}_i, \boldsymbol{\Theta}) \tag{3.13}$$

where the LS represents the log-sum term and I represents the integral term in Equation 3.10. $\boldsymbol{\Theta}$ represents the parameters of the basis functions in conjunction with the coefficients for event sequence $\mathbf{b}_i$ and the particular baseline intensity μ_i. Learning the bases in conjunction with coefficients, comes at additional no asymptotic cost versus fitting the same coefficients for all trajectories.

3.4.6 Log-Summation Evaluation

We now describe a strategy for the efficient computation of $\mathrm{LS}(\mathbf{X}_i, \boldsymbol{\Theta})$ in line with [59]. First expand the LS term as

$$\mathrm{LS}(\mathbf{x}, \boldsymbol{\Theta}) = \sum_{j=1}^{|\mathbf{X}_i|} \log(\mu_i + \sum_{t' \in \mathcal{H}_t} k_i(t', t) \tag{3.14}$$

$$\approx \sum_{j=1}^{|\mathbf{X}_i|} \log(\mu_i + \sum_{t_k < t_j} \hat{k}_i(t_j - t_k, t_j)) \tag{3.15}$$

To compute $\hat{k}_i$, recall that $k_i(t', t)$ consists of a weighted combination of the evaluations of functions in $\boldsymbol{\Psi}, \boldsymbol{\Phi}$ of according to $\mathbf{b}_i$. We need only evaluate the components of $\boldsymbol{\Psi}$ and $\mathbf{U}$ at the time-points $\{t_j\}_{j=1}^{|\mathbf{X}_i|}$. To approximate the displacement terms $t_j - t_k$, we can first evaluate each of the basis functions in $\boldsymbol{\Phi}$ with a sufficiently dense uniform grid over the time horizon $[0, T)$. We approximate the value $\hat{\varphi}(t_j - t_k)$ by linearly interpolating using the two nearest grid-points $t_j - t_k$, thus avoiding the quadratic cost to consider to consider pairs. We reuse these calculations for all the sequences (with different weightings).

Algorithm 1 Joint Learning of Kernel Basis Functions and Sequence Representations

```
**Input:** Sequences $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$. Temporal Ranks $L$. Batch Size $B$. Training Epochs $E$.
Pretraining Epochs $P$. Regularization Factor $\beta$.
**Output:** $\mathbf{\Theta} = (\mathbf{M}, \mathbf{B}, \mathbf{\Phi}, \mathbf{\Psi})$. $\mathbf{M}$ baseline intensities, $\mathbf{B}$ coefficient representations, $\mathbf{\Phi}, \mathbf{\Psi}$
temporal basis functions.
Initialize $\mathbf{A} \leftarrow \mathbf{0}, \mathbf{M} \leftarrow \mathbf{0}$.
Randomly initialize networks in $\mathbf{\Phi}, \mathbf{\Psi}$.
for $i \in \{1, 2, \dots, P + E\}$ training epochs do
  for $j \in \{1, 2, \dots, \lceil n/B \rceil\}$ batches do
    Sample batch $(\mathbf{X}_j, \mathbf{B}_j)$ from $(\mathbf{X}, \mathbf{B})$ sequences (representations).
    Evaluate $\mathbf{\Psi}$ at $\{t_j\}_{j=1}^{|\mathbf{X}_i|}$. Evaluate $\mathbf{\Phi}$ on dense uniform grids.
    Evaluate $\hat{\ell}(\mathbf{x}_l, \mathbf{\Theta})$ for each $\mathbf{x}_l$ in $\mathbf{X}_j$.
    Compute $\hat{\mathcal{L}} = \frac{1}{B}\sum_{l=1}^{B} \hat{\ell}(\mathbf{x}_l, \mathbf{\Theta}) + \beta||\mathbf{b}_l||_2^2$
    if $i \leq P$ then
      Update $\mathbf{\Theta}$ with via SGD step on $-\nabla_{\mathbf{\Theta}'}$ where $\mathbf{\Theta}'$ holds all weights in each column
      $\mathbf{b}_i$ constant.
    else
      Update $\mathbf{\Theta}$ with via SGD step on $-\nabla_{\mathbf{\Theta}}\hat{\mathcal{L}}$.
    end if
  end for
end for
```

3.4.7 Computation of the Integral over Conditional Intensity

To approximate $\mathrm{I}(\mathbf{x}_i, \mathbf{\Theta})$ with, we expand as

$$\mathrm{I}(\mathbf{X}_i, \mathbf{\Theta}) = \int_{t \in \mathcal{X}} \lambda_i(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_0^T \lambda_i(t) \mathrm{d}t \tag{3.16}$$

$$= \mu T + \sum_{j=1}^{|\mathbf{x}|} \int_0^T \mathbf{1}[t_j < t] k_i(t_j, t) \mathrm{d}t \tag{3.17}$$

$$= \mu T + \sum_{i=1}^{|\mathbf{x}|} \sum_{l=1}^{L} \alpha_{ilr} \psi_l(t_i) \int_0^{T - t_i} \phi_l(t) \mathrm{d}t$$

We can compute the integral over ϕ_l using the pre-computed $\mathbf{\Phi}$ grid. We take a linear spline $V_{\phi_l}(t)$ across the grid, and evaluate $\int_0^{t_e} \phi_l(t)\mathrm{d}t \approx \int_0^{t_e} V_{\phi_l}(t)\mathrm{d}t$ [59]. We approximate the integral by averaging the grid evaluations within the integration domain.

3.4.8 Model Training

We train our model via the procedure described in the Algorithm 1 using Stochastic Gradient Descent (SGD). To prevent poor initialization, we start by take all of the weights in each column $\mathbf{b}_i$ to be the same for the first few epochs. We compute the regularization term at the batch level. We apply gradient descent on the likelihood function directly to optimize the log-likelihood, in contrast to work using mixture models that require EM [47].

We consider the per-epoch cost of Algorithm 1 a given sequence of n trajectories having maximum sequence length m, number of basis functions L, number of temporal grid-points P_t, number of spatial grid-points P_s, and neural-network evaluation time τ. We require m evaluations of constituents of $\mathbf{\Psi}$ requiring time of $\mathcal{O}((L+R)(nm+m\tau))$. We use grid-points for $\mathbf{\Phi}$ requiring time $\mathcal{O}(Lnm+LP_t\tau)$. The remaining work is (e.g. calculating the regularization) is bounded by $\mathcal{O}(nm)$. Since in practice $P_t \ll \tau, m$. We have the per epoch time as bounded by $\mathcal{O}(L(mn+\tau))$. The computational efficiency of our method enables the us to harnesses longer sequences as well as greater data volume.

3.5 Numerical Illustration

In this section, we detail a numerical illustration to demonstrate our model. A brief overview of the type of synthetic datasets we consider is given in Table 3.1. We generate a number of different permutations of parameters for each of the types of sequences. We evaluate three types of sequence data: Temporal only sequences (including both non-stationary and stationary processes), Temporal sequences with marks, and spatio-temporal sequences. We also discuss the choice of hyper-parameter r (the number of basis functions) and β (the choice of the regularization factor) in the context of clustering. A detailed description of the synthetic datasets is given in Appendix A.

Table 3.1: Brief Summary of Synthetic Evaluation Datasets

GROUPING	TYPE
NONSTATIONARY	EXP.+COSINE
	INHIBITION
	INHOMOG. POISSON

3.5.1 Baselines

We compare our method SSEM against the following baselines:

1. Empirical Distribution of Inter-Arrival Times (EDAT)

2. Marked Point Process Distance (MPPD)

3. Dirichlet Mixture model of Hawkes Processes (DMHP)

Briefly Item 1 and Item 2 attempt to define a pairwise distance between event sequences. Finally, Item 3 gives each sequence as a mixture of underlying point processes based on generative processes (as a mixture of Hawkes processes).

For each evaluated method, we provide a short description, tested variants, and hyperparameters. In general, the tested methods fall under three broad categories: (1) *Embedding*, which produces, for given event sequence $\mathbf{X}$, a vector $\mathbf{e_X} \in \mathbb{R}^d$, where d is the dimension of the latent space; (2) pairwise-*Distance*, which outputs a pair-wise distance $D(\mathbf{X}, \mathbf{Y})$ between given sequences $\mathbf{X}$ and $\mathbf{Y}$; and (3) *Assignment* which outputs an assignment of a sequence to some mixture of categories. We consider event sequences of form $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{|X|}$, where $\mathbf{x}_i = (t_i, \mathbf{m}_i) \in \mathbb{R}_{\geq 0} \times \mathbb{R}^m$ is the ith event in the pattern, and m is the dimension of the spatial/mark space. Furthermore, let $\mathcal{Z} = \{\mathbf{X}_i\}_{i=1}^{|\mathcal{Z}|}$ be the sequence of event trajectories under consideration.

[Distance] Empirical Distribution of Inter-Arrival Times (EDAT) We consider only the temporal component of the event sequences, i.e. $\mathbf{x}_i = t_i, t_i \in \mathbb{R}_{\geq 0}$. A naive way to quantify the distance between two event sequences is to first compute an empirical distribution of inter-arrival times for some event sequence **X** to yield

$$(\Delta\mathbf{X})(t) = \frac{1}{|X|-1} \sum_{i=1}^{|X|-1} \delta(|\mathbf{x}_{i+1} - \mathbf{x}_i| - t). \tag{3.18}$$

Then we take as the distance between two sequences as

$$D(\mathbf{X},\mathbf{Y}) = \mathcal{W}_1(\Delta\mathbf{X},\Delta\mathbf{Y}) = \inf_{\pi\in\Gamma(\Delta\mathbf{X},\Delta\mathbf{Y})} \int_{\mathbb{R}\times\mathbb{R}} |x-y| \mathrm{d}\pi(x,y) \tag{3.19}$$

where the distance between the two event sequences is taken to the be the first Wasserstein distance between the empirical distributions of inter-arrival times and where $\Gamma(u,v)$ denotes the set of probability distributions on $\mathbb{R}\times\mathbb{R}$ [72].

[Distance] Marked Point Process Distance (MPPD) We compute the pair-wise distances between all event sequences according to the *cumulative distance* as per [64].

Suppose that we have two such sequences **X** and **Y**, which each contain at most N events respectively. We first define a *cumulative count function*

$$F^{\mathbf{s}}_{\mathbf{X}}(\mathbf{v}) = |\{\mathbf{x}\in\mathbf{X} | \forall j, s_j x_j < s_j v_j\}| \tag{3.20}$$

where $\mathbf{s}\in\{-1,1\}^m$ selected on a hypercube determines the direction of counting. Consider $\mathbf{i}\subseteq[m]$ to be a subset of an index set the space, then the restriction F to the subspace defined by **i** is

$$F^{\mathbf{s}}_{\mathbf{X}}(\mathbf{v};\mathbf{i}) = |\{\mathbf{x}\in\mathbf{X} | \forall j\in\mathbf{i}, s_j x_j < s_j v_j\}| \tag{3.21}$$

Then, the *cumulative distance* between marked point processes realizations is defined as

$$D(\mathbf{X}, \mathbf{Y}) = \sum_{\mathbf{i}} \lambda(\mathbf{i}) \sqrt{\sum_{\mathbf{s} \in \{-1,1\}^m} \int \left(F_{\mathbf{X}}^{\mathbf{s}}(\mathbf{v}; \mathbf{i}) F_{\mathbf{Y}}^{\mathbf{s}}(\mathbf{v}; \mathbf{i})\right)^2 \mathrm{d}\mathbf{v}} \tag{3.22}$$

in which $\lambda(\mathbf{i})$ is the weight assigned to subspace $\mathbf{i}$, which are found via neighborhood components analysis [73]. The computation time of the distance metric for two sequences above is $\mathcal{O}(2^m |\mathbf{X}||\mathbf{Y}|)$. We use the implementation as given in the `TAHP` toolkit[1].

[Assignment] Dirichlet Mixture model of Hawkes Processes (DMHP) To discover clustering structure across different sequences, we suppose that the event sequences $\mathcal{Z}$ consists of a mixture of Hawkes processes, and so each of the sequences belonging to the $k \in [K]$-th grouping and for each event of type c we have

$$\lambda_c^k(t) = \mu_c^k + \sum_{t_i < t} \psi_{cc_i}(k)(t - t_i) = \mu_c^k \sum_{t_i < t} \sum_{r=1}^{R} \alpha_{cc_i,r}^k \kappa_r(tt_i) \tag{3.23}$$

then the following the Dirichlet Mixture Model underlines the generation of the sequences

$$\pi \sim \mathrm{Dir}(\alpha/K, \ldots \alpha/K), k|\pi \sim \mathrm{Category}(\pi) \tag{3.24}$$

$$\mu \sim \mathrm{Rayleigh}(\mathbf{B}), \mathbf{A} \sim \mathrm{Exp}(\boldsymbol{\Sigma}), \mathbf{s}|k, \mu, \mathbf{A} \sim \mathtt{Hawkes}(\mu_k, \mathbf{A}_k) \tag{3.25}$$

The intensity of each event is generated via a single Hawkes process, while the likelihood of an event sequence is a mixture of likelihood functions from different Hawkes processes [47]. The parameters are found by Variational Bayesian Inference. We interpret the assignment of each of the sequences $\mathbf{X}_i$ as the embedding. We use the implementation as given in the `TAHP` toolkit.

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

3.5.2 Clustering

Within the temporal sequences we consider $k \in \{2, 3, 5\}$ different types of sequences and for spatio-temporal and marked sequences, we report with number of clusters $k = 3$. During each run of the experiment, we select k distinct point processes (selected from the groupings from Table 3.1 with various permutations of parameters) and generate $n = 500$ sequences from each of the point processes for a total of $n * k$ different sequences. The sequences on average have a length of 259 elements. We then evaluate the performance of each method using these sequences, and repeat the experiment 30 times and for each different grouping of point processes. We choose the combinations to be distinct over each trial. We report the mean performance of each method in Table 3.3. We evaluate the clustering performance of each method based on three metrics

1. Clustering Purity (CP),
2. Adjusted Rand Index (ARI), and
3. Normalized Mutual Information (NMI)

We consider the clustering experiments where the clusters are known (synthetic data setting). For each the methods, if the method type is *Distance*, we apply Spectral Clustering to determine the optimal cluster assignments. If the method outputs an *Assignment*, we take the category assigned the most weight. If the method type is *Embedding* we use Spectral Clustering on the distance matrix of the Euclidean distance between the embeddings.

Synthetic Data (Clusters Known): We aim recover and group together the sequences which are drawn from the same generating processes. To accomplish this task, we each pair/triple/quintet $k \in \{2, 3, 5\}$ from each category of synthetic data. We then generate $m = 500$ trajectories from category and apply each of the methods above.

To evaluate the performance of the clustering we consider the following three metrics:

- **Clustering Purity (CP)** Correct assignments divided by total assignments.

$$\text{CP} = \frac{1}{N}\sum_{k=1}^{K} |\mathcal{W}_k \cap \mathcal{C}_j| \tag{3.26}$$

  where N is number of data, K the number of classes and $\{\mathcal{W}_k\}_{k=1}^K$, $\{\mathcal{C}_k\}_{k=1}^K$ matched partitions over the data representing the assigned cluster or the real cluster for index k respectively.

- **Adjusted Rand Index (ARI)** Similarity of learned and ground truth assignments [74]. For matched clustering partitions

$$\text{RI} = (a+b)/\binom{N}{2}, \qquad \text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{\max(\text{RI}) - \mathbb{E}[RI]]} \tag{3.27}$$

- **Normalized Mutual Information (NMI)** The mutual information between the true class labels and the cluster assignments, normalized by the entropy of the true labels and the cluster assignments [75].

$$\text{NMI} = \frac{2I(X;Y)}{H(X) + H(Y)} \tag{3.28}$$

  where X, Y are the true and assigned labels, $H(X)$ is the entropy of X, and $I(X;Y)$ is the mutual information between X and Y.

To find the clusters, if the method gives a distance or an embedding, we apply a Hierarchical Clustering to the distance matrices. If the method outputs an assignment, we take the the one with the greatest influence as the assignment. We detail results from synthetic temporal data clustering with different numbers of basis functions and given k different types of generated sequence in Table 3.3.

Number of Basis Functions and Regularization Parameter To demonstrate that low rank representation is often sufficient, and determine the optimal number of basis functions, we ran the clustering experiment for $r \in \{2, 3, 4, 5\}$ basis functions and report the performance in Table 3.2 using the the same experimental scheme as before. In the temporal setting, we find that $n = 3$ is sufficient for the tested temporal sequences. Table 3.2 describes the model performance on synthetic temporal data clustering with different numbers of basis functions and given k different types of generated sequence.

Table 3.2: Performance of synthetic temporal data clustering with different numbers of basis functions and given k different types of generated sequence

		Rank			
k	Metric	2	3	4	5
2	Purity	0.915	**0.930**	0.913	0.911
	ARI	0.785	**0.796**	0.774	0.769
	NMI	0.766	**0.770**	0.751	0.744
3	Purity	0.792	**0.801**	0.793	0.767
	ARI	0.593	**0.613**	0.608	0.559
	NMI	0.616	**0.648**	0.644	0.591
5	Purity	**0.654**	0.564	0.584	0.534
	ARI	**0.438**	0.319	0.373	0.287
	NMI	**0.546**	0.449	0.524	0.435

Recovery of Embedding of Different Generating Processes To illustrate the recovery of embeddings, Figure 3.2 provides an illustration of the model training loss across 200 epochs and $R = 3$ basis functions recovering from samples of an inhomogeneous Poisson process as well as a delayed exponential process. The histogram displays the learned separation of the baseline intensities. The remaining six plots display PCA projections of the learned embeddings with and without the baseline intensities and compare the true labels versus those found by hierarchical clustering.

Table 3.3: Average performance for clustering of synthetically generated sequences with k different types of sequence

Type	k	Type	Purity	ARI	NMI
Temporal	2	DMHP	0.538	0.062	0.056
		EDAT	0.805	0.467	0.442
		MPPD	0.754	0.376	0.390
		SSEM	**0.884**	**0.640**	**0.603**
Temporal	3	DMHP	0.230	0.020	0.057
		EDAT	**0.572**	**0.408**	**0.518**
		MPPD	0.420	0.168	0.272
		SSEM	0.549	0.297	0.373
Temporal	5	DMHP	0.364	0.042	0.049
		EDAT	0.666	0.410	0.483
		MPPD	0.607	0.283	0.326
		SSEM	**0.750**	**0.530**	**0.565**

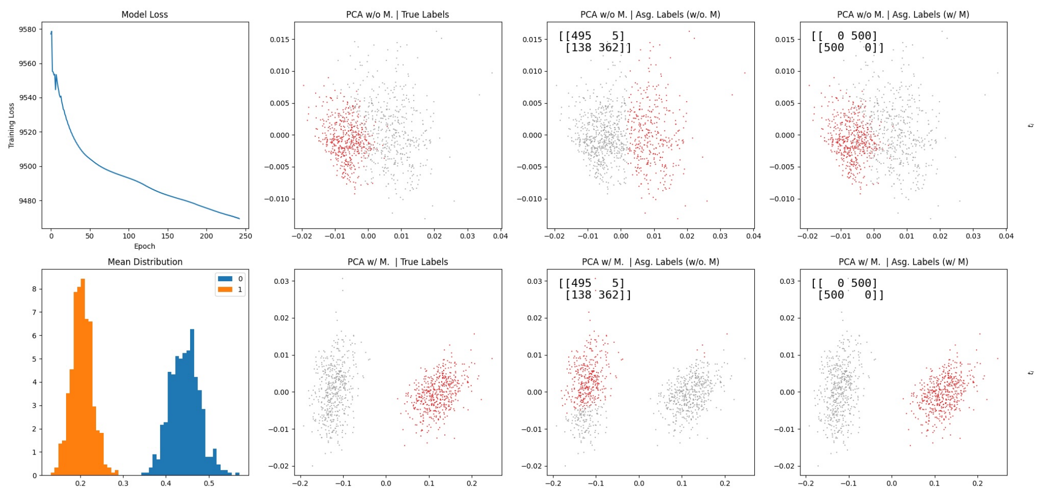


Figure 3.2: Illustration of Model Training and Embedding Recovery from Synthetic Data. The leftmost two panels represent the model training loss (log-likelihood) across 200 epochs and the learned baseline intensities ($R = 3$ basis functions) recovering from samples of an inhomogeneous Poisson process as well as a delayed exponential process. The histogram displays the learned separation of the baseline intensities. The remaining six plots depict PCA projections of the learned embeddings with and without the baseline intensities, comparing the true labels versus those found by hierarchical clustering.

Appendices

APPENDIX A
POINT PROCESSES DATA GENERATION PROCEDURE

We detail the data generation algorithms used for the synthetic data. In Algorithms Algorithm 3 and Algorithm 2, $\lambda^*(t)$ is generally taken to be a Homogeneous Poisson Process with intensity $\bar{\lambda} = \sup_{t \in [0,T]} \lambda(t)$, and realizations of generated $\lambda^*(t)$ as generated as needed.

Algorithm 2 Ogata Thinning for Non-Stationary Point Processes [76]

```
Input: Desired intensity $\lambda(\cdot)$, stop time $T$, spatial/mark distribution $\mathcal{S}$, known intensity
function $\lambda^*(\cdot)$,
$i := 1; t := 0; x_1 := 0$
repeat
    Sample $u, d \sim \mathrm{Uniform}(0, 1)$
    Sample $s \sim \mathcal{S}$
    $t := t_i - \log u / \lambda^*(t)$
    if $d \leq \lambda(t, s \mid t_1 \dots t_i) / \lambda^*(t)$ then
        $t_i := t$
        $i := i + 1$
    end if
until $t_i \geq T$
return $[t_1, t_2, \dots, t_i]$
```

A.1 Synthetic Data Descriptions

Unless otherwise specified, all non-homogeneous sequences below are generated from their kernel or intensity functions via thinning according to the Ogata method for point processes [76], and the Lewis method for inhomogeneous Poisson processes [77]. We take all realizations on a finite time horizon of $t \in [0, 600]$ and report statistics using 500 generated trajectories in each category. Average sequence length is defined as the number of events in each trajectory. The average sequence variance is defined as the variance. The inter-arrival variance is the variance between the arrival times of the events in the processes.

Table A.1: Summary of Synthetic Evaluation Datasets

Grouping	Type	Parameters	Avg. Seq. Length	Avg. Seq. Variance	Interarrival Variance
Stationary Temporal	Homog. Poisson	$\mu = 0.7$	418.14	410.95	2.06
		$\mu = 0.55$	329.30	309.61	3.34
		$\mu = 0.4$	239.69	246.22	6.28
	Gaussian	$\mu = 0.2, \eta = 0.6, \sigma = 3.5$	169.70	330.44	18.12
		$\mu = 0.2, \eta = 1.0, \sigma = 3.5$	238.29	975.66	12.21
		$\mu = 0.2, \eta = 0.6, \sigma = 8.0$	171.37	314.47	15.27
		$\mu = 0.2, \eta = 1.0, \sigma = 8.0$	236.85	858.14	9.51
	(Delayed) Exp.	$\mu = 0.2, \alpha = 0.4, \beta = 1.1, d = 3.5$	186.16	436.53	14.44
		$\mu = 0.2, \alpha = 0.6, \beta = 1.4, d = 3.5$	208.39	642.99	12.37
		$\mu = 0.2, \alpha = 0.6, \beta = 1.1, d = 0.0$	264.94	1294.52	14.42
		$\mu = 0.2, \alpha = 0.4, \beta = 1.1, d = 0.0$	189.75	521.02	19.00
		$\mu = 0.2, \alpha = 0.6, \beta = 1.4, d = 0.0$	210.27	600.40	18.07
		$\mu = 0.2, \alpha = 0.6, \beta = 1.1, d = 3.5$	263.63	1353.97	8.83
		$\mu = 0.2, \alpha = 0.4, \beta = 1.4, d = 3.5$	167.77	322.29	16.60
		$\mu = 0.2, \alpha = 0.4, \beta = 1.4, d = 0.0$	168.70	353.16	21.22
Nonstationary Temporal	Inhomog. Poisson	$\mu = 0.15, \alpha = 0.12, \beta = 30$	449.45	447.62	3.79
		$\mu = 0.15, \alpha = 0.12, \beta = 100$	449.49	429.17	4.08
		$\mu = 0.15, \alpha = 0.06, \beta = 100$	449.26	425.52	2.09
		$\mu = 0.15, \alpha = 0.06, \beta = 30$	448.76	455.66	2.10
	Inhibition	$\alpha = 0.6, \beta = 0.0002, \gamma = 1.1$	248.69	1022.44	15.44
		$\alpha = 0.4, \beta = 0.0002, \gamma = 1.5$	161.97	264.65	21.69
		$\alpha = 0.4, \beta = 0.0005, \gamma = 1.1$	174.33	353.82	20.55
		$\alpha = 0.6, \beta = 0.0005, \gamma = 1.1$	225.65	845.57	16.95
		$\alpha = 0.4, \beta = 0.0002, \gamma = 1.1$	184.42	417.84	19.36
		$\alpha = 0.4, \beta = 0.0005, \gamma = 1.5$	156.07	283.49	22.30
		$\alpha = 0.6, \beta = 0.0002, \gamma = 1.5$	191.95	487.32	19.70
		$\alpha = 0.6, \beta = 0.0005, \gamma = 1.5$	182.11	424.75	20.35
	Exp.+Cosine	$\alpha = 0.4, \beta = 1.0, \gamma = 2$	155.45	263.17	21.83
		$\alpha = 0.6, \beta = 1.0, \gamma = 1$	188.39	520.30	19.19
		$\alpha = 0.6, \beta = 1.0, \gamma = 2$	184.78	435.18	19.55
		$\alpha = 0.4, \beta = 0.6, \gamma = 2$	196.45	593.07	17.47
		$\alpha = 0.4, \beta = 1.0, \gamma = 1$	156.99	275.99	22.01
		$\alpha = 0.6, \beta = 0.6, \gamma = 1$	374.83	15155.01	10.73
		$\alpha = 0.4, \beta = 0.6, \gamma = 1$	196.72	584.28	17.32
		$\alpha = 0.6, \beta = 0.6, \gamma = 2$	363.55	14903.75	10.68

Stationary Temporal Processes

We provide statistics for the generated temporal processes in Table Table A.1.

Uniform Poisson $\lambda(t) = \lambda$.

$\lambda \in \{0.4, 0.55, 0.7\}$.

(Delayed) Exponential Kernel $k(t, t') = \alpha \exp(-\beta(t - t' - d))$.

$\mu \in \{0.2\}, \alpha \in \{0.6, 0.4\}, \beta \in \{1.1, 1.4\}, d \in \{0.0, 3.5\}$

Gaussian Kernel $\frac{\eta}{\sigma\sqrt{2\pi}} \exp(-\frac{(t-t')^2}{2\sigma^2})$.

$\mu \in \{0.2\}, \eta \in \{0.6, 1\}, \sigma \in \{3.5, 8\}$

Non-Stationary and Inhibition Temporal Processes

Inhomogenous Poisson $\lambda(t) = \lambda + \alpha \sin \frac{t\pi}{\beta}$.

$\lambda \in \{0.15\}, \alpha \in \{0.06, 0.12\}, \beta \in \{30, 100\}$

Inhibition $k(t', t) = \alpha(1 - \beta t) \exp(-\eta(t - t'))$

$\alpha \in \{0.4, 0.6\}$, $\beta \in \{\ 0.00005, 0.0002\}$, $\gamma \in \{1.1, 1.5\}$

Exponential-Cosine $\alpha \exp(-\beta(t - t'))(\frac{1}{2} + \frac{1}{2}\cos(\gamma t)$

$\alpha \in \{0.4, 0.6\}\ \beta \in \{0.6, 1.0\}\ \gamma \in \{1, 2\}$

Algorithm 3 Lewis Thinning for Inhomogeneous Poisson Processes [77]

```
Input: Desired intensity $\lambda(\cdot)$, stop time $T$, spatial/mark distribution $\mathcal{S}$, known intensity function $\lambda^*(\cdot)$ with realizations $[0, x_2^*, x_3^*, \ldots]$
$i := 1$; $j := 1$; $t_1 := 0$; $x_1 := 0$
repeat
    Sample $u \sim \mathrm{Uniform}(0, 1)$
    Sample $s \sim \mathcal{S}$
    if $u \leq \lambda(x_{j+1}, s)/\lambda^*(x_{j+1})$ then
        $t_{i+1} := x_{j+1}$
        $i := i + 1$
    end if
    $j := j + 1$
until $s_j \geq T$
return $[t_1, t_2, \ldots, t_i]$
```

REFERENCES

[1] D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes. Volume II: General Theory and Structure*. Springer, 2008.

[2] J. Moller and R. P. Waagepetersen, *Statistical inference and simulation for spatial point processes*. CRC press, 2003.

[3] Microsoft Corporation, *Microsoft Sequence Clustering Algorithm*, publisher: Microsoft Corporation, Oct. 2023.

[4] W. Min, W. Liang, H. Yin, Z. Wang, M. Li, and A. Lal, "Explainable deep behavioral sequence clustering for transaction fraud detection," *CoRR*, vol. abs/2101.04285, 2021. arXiv: 2101.04285.

[5] L. Rasmy, Y. Xiang, Z. Xie, C. Tao, and D. Zhi, "Med-bert: Pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction," *npj Digital Medicine*, vol. 4, no. 1, p. 86, May 20, 2021.

[6] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, "System log clustering approaches for cyber security applications: A survey," *Computers & Security*, vol. 92, p. 101 739, 2020.

[7] R. Argelaguet *et al.*, "Multi-omics factor analysis-a framework for unsupervised integration of multi-omics data sets," *Mol Syst Biol*, vol. 14, no. 6, e8124, 2018.

[8] R. Argelaguet *et al.*, "Mofa+: A statistical framework for comprehensive integration of multi-modal single-cell data," *Genome Biology*, vol. 21, no. 1, p. 111, 2020.

[9] D. Babaev *et al.*, "Coles: Contrastive learning for event sequences with self-supervision," in *Proceedings of the 2022 International Conference on Management of Data*, ser. SIGMOD '22, Philadelphia, PA, USA: Association for Computing Machinery, 2022, pp. 1190–1199, ISBN: 9781450392495.

[10] K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[11] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009, ISBN: 0262013193.

[12] M. A. Davenport and J. Romberg, "An overview of low-rank matrix recovery from incomplete observations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 608–622, Jun. 2016.

[13] J. Bennett, C. Elkan, B. Liu, P. Smyth, and D. Tikk, "Kdd cup and workshop 2007," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 51–52, Dec. 2007.

[14] A. Juditsky and A. Nemirovski, *Statistical Inference via Convex Optimization*. Princeton University Press, 2020, ISBN: 9780691197296.

[15] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. New York, NY: Springer New York, 2011, pp. 185–212, ISBN: 978-1-4419-9569-8.

[16] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[17] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010. eprint: https://doi.org/10.1137/080738970.

[18] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[19] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[20] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 40–48, Nov. 2010.

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. null, pp. 993–1022, Mar. 2003.

[22] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, Paris, France: Association for Computing Machinery, 2009, pp. 947–956, ISBN: 9781605584959.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[24] M. E. Peters *et al.*, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237.

[25] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[27] C. Leslie, E. Eskin, and W. S. Noble, “The spectrum kernel: a string kernel for SVM protein classification,” *Pac Symp Biocomput*, pp. 564–575, 2002.

[28] P. Smyth, “Clustering sequences with hidden markov models,” in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, ser. NIPS’96, Denver, Colorado: MIT Press, 1996, pp. 648–654.

[29] S. Helske and J. Helske, “Mixture hidden markov models for sequence data: The seqhmm package in r,” *Journal of Statistical Software*, vol. 88, no. 3, pp. 1–32, 2019.

[30] A. Juditsky, A. Nemirovski, Y. Xie, and C. Xu, *Generalized generalized linear models: Convex estimation and online bounds*, 2023. arXiv: 2304.13793 `[stat.ME]`.

[31] A. B. Juditsky and A. S. Nemirovski, “Signal recovery by stochastic optimization,” *Automation and Remote Control*, vol. 80, no. 10, pp. 1878–1893, Oct. 1, 2019.

[32] A. Juditsky, A. Nemirovski, L. Xie, and Y. Xie, “Convex parameter recovery for interacting marked processes,” *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 3, pp. 799–813, Nov. 2020.

[33] B. K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995. eprint: https://doi.org/10.1137/S0097539792240406.

[34] B. Recht, M. Fazel, and P. A. Parrilo, “Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization,” *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010. eprint: https://doi.org/10.1137/070697835.

[35] Y. Nesterov and A. Nemirovski, "On first-order algorithms for l1/nuclear norm minimization," *Acta Numerica*, vol. 22, pp. 509–575, 2013.

[36] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modeling and Theory, Third Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2021. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976595.

[37] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization* (MOS-SIAM Series on Optimization). Society for Industrial and Applied Mathematics, 2001.

[38] A. Nemirovski, "Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004. eprint: https://doi.org/10.1137/S1052623403425629.

[39] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[40] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *Journal of Neurophysiology*, vol. 93, no. 2, pp. 1074–1089, 2005, PMID: 15356183. eprint: https://doi.org/10.1152/jn.00697.2004.

[41] E. W. Fox, M. B. Short, F. P. Schoenberg, K. D. Coronges, and A. L. Bertozzi, "Modeling e-mail networks and inferring leadership using self-exciting point processes," *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 564–584, 2016. eprint: https://doi.org/10.1080/01621459.2015.1135802.

[42] J. R. Zipkin, F. P. Schoenberg, K. Coronges, and A. L. Bertozzi, "Point-process models of social network interactions: Parameter estimation and missing data recovery," *European Journal of Applied Mathematics*, vol. 27, no. 3, pp. 502–529, 2016.

[43] F. P. Schoenberg, M. Hoffmann, and R. J. Harrigan, "A recursive point process model for infectious diseases," *Annals of the Institute of Statistical Mathematics*, vol. 71, no. 5, pp. 1271–1287, 2019.

[44] A. Reinhart, "A review of self-exciting spatio-temporal point processes and their applications," *Statistical Science*, 2017. eprint: arXiv:1708.02647.

[45] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[46] O. Shchur, A. C. Turkmen, T. Januschowski, and S. Gunnemann, "Neural temporal point processes: A review," in *IJCAI 2021*, 2021.

[47] H. Xu and H. Zha, "A Dirichlet mixture model of Hawkes processes for event sequence clustering," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[48] D. Luo *et al.*, "Multi-task multi-dimensional Hawkes processes for modeling event sequences," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15, Buenos Aires, Argentina: AAAI Press, 2015, pp. 3685–3691, ISBN: 9781577357384.

[49] Z. Zhu, C. Yin, B. Qian, Y. Cheng, J. Wei, and F. Wang, "Measuring patient similarities via a deep architecture with medical concept embedding," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 749–758.

[50] M. Mollenhauer, I. Schuster, S. Klus, and C. Schütte, "Singular value decomposition of operators on reproducing kernel hilbert spaces," in *Advances in Dynamics, Optimization and Computation: A volume dedicated to Michael Dellnitz on the occasion of his 60th birthday*, Springer, 2020, pp. 109–131.

[51] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. Bach, "Supervised dictionary learning," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21, Curran Associates, Inc., 2008.

[52] Y. Lecun, *Modeles connexionnistes de l'apprentissage (connectionist learning models)*. Universite P. et M. Curie (Paris 6), Jun. 1987.

[53] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, Jun. 2014, pp. 1188–1196.

[54] L. Song, B. Boots, S. M. Siddiqi, G. Gordon, and A. Smola, "Hilbert space embeddings of hidden markov models," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 991–998, ISBN: 9781605589077.

[55] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[56] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, Jul. 1971, Full publication date: Apr., 1971.

[57] R. T. Q. Chen, B. Amos, and M. Nickel, "Neural spatio-temporal point processes," in *International Conference on Learning Representations*, 2021.

[58] S. Zhu, H. Wang, Z. Dong, X. Cheng, and Y. Xie, "Neural spectral marked point processes," in *International Conference on Learning Representations*, 2022.

[59] Z. Dong, X. Cheng, and Y. Xie, *Spatio-temporal point processes with deep non-stationary kernels*, 2022.

[60] 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*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1555–1564, ISBN: 9781450342322.

[61] S. Zuo, H. Jiang, Z. Li, T. Zhao, and H. Zha, "Transformer hawkes process," in *International Conference on Machine Learning*, PMLR, 2020, pp. 11 692–11 702.

[62] H. Mei and J. Eisner, "The neural Hawkes process: A neurally self-modulating multivariate point process," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6757–6767, ISBN: 978-1-5108-6096-4.

[63] Y. Zhang, J. Yan, X. Zhang, J. Zhou, and X. Yang, "Learning mixture of neural temporal point processes for multi-dimensional event sequence clustering," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed., Main Track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 3766–3772.

[64] K. Iwayama, Y. Hirata, and K. Aihara, "Definition of distance for nonlinear time series analysis of marked point process data," *Physics Letters A*, vol. 381, no. 4, pp. 257–262, 2017.

[65] M. C. van Rossum, "A novel spike distance," *Neural Comput*, vol. 13, no. 4, pp. 751–763, Apr. 2001.

[66] M. Okawa, T. Iwata, Y. Tanaka, H. Toda, T. Kurashima, and H. Kashima, "Dynamic Hawkes processes for discovering time-evolving communities' states behind diffusion processes," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1276–1286.

[67] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm192761164.

[68] J. Mercer, “Functions of positive and negative type and their commection with the theory of integral equations,” *Philos. Transactions Royal Soc.*, vol. 209, pp. 4–415, 1909.

[69] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993.

[70] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10, Haifa, Israel: Omnipress, 2010, pp. 807–814, ISBN: 9781605589077.

[71] S. Remes, M. Heinonen, and S. Kaski, “Neural non-stationary spectral kernel,” *arXiv preprint arXiv:1811.10978*, 2018.

[72] A. Ramdas, N. G. Trillos, and M. Cuturi, “On wasserstein two-sample testing and related families of nonparametric tests,” *Entropy*, vol. 19, no. 2, 2017.

[73] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, “Neighbourhood components analysis,” in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17, MIT Press, 2004.

[74] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” *Journal of Machine Learning Research*, vol. 11, no. 95, pp. 2837–2854, 2010.

[75] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Is a correction for chance necessary?” In *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML ’09, Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 1073–1080, ISBN: 9781605585161.

[76] Y. Ogata, “On Lewis’ simulation method for point processes,” *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 23–31, Jan. 1981.

[77] P. W. Lewis and G. S. Shedler, “Simulation of nonhomogeneous poisson processes by thinning,” *Naval research logistics quarterly*, vol. 26, no. 3, pp. 403–413, 1979.