Approximate Inference Applications to Representation Learning and Stochastic Processes Problems

Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
César Ali Ojeda Marin
aus
Lechería, Venezuela
Bonn, 2021

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Promotionskommission:

- Erstgutachter: Prof. Dr. Christian Bauckhage
- Zweitgutachter: Prof. Dr. Stefan Wrobel
- Fachnahes Mitglied: Prof. Dr. Manfred Opper
- Fachfremdes Mitglied: Prof. Dr. Cyrill Stachniss

Tag der Promotion: 19. Januar 2021
Erscheinungsjahr: 2021

Summary

The present dissertation dwells in the development of inference algorithms and methodologies for the study of dynamical datasets. We developed techniques to analyze time-series datasets for point processes, switching dynamical systems, and queues systems dynamics. Furthermore, we developed analysis in the interplay of dynamic population behavior and how semantic structures inform this behavior. Conversely, we studied how dynamics in semantic spaces can be exploited to explain black-box classifiers' decisions. Concretely, we extended the Hawkes process incorporating richer correlations in the excitations via introducing a sigmoid link function over a Gaussian process prior. We incorporate a Polya Gamma data augmentation approach and a sparse Gaussian process approximation into a mean field treatment of the variational lower bound to perform inference with analytic updates, obtaining a fast and scalable algorithm. Second, we introduce a flexible methodology for handling temporal Poisson process intensities based on spline interpolation for fast and scalable unsupervised analysis of point processes. We then propose a similarity measure for time series that is invariant to translations of local patterns. With this similarity measure, we develop a spectral clustering algorithm with a flexible, piecewise kernel evaluation for efficient computation, scaling to a large amount of data. The clustering procedure incorporates an entropy measure to determine how well a certain (intensity) time series is represented by a cluster prototype, allowing for the detection of outliers within a temporal pattern sample. Thirdly, we provide a deep learning solution for service times of queue systems; we exploit the representation learning capabilities of deep neural networks for point processes to infer service time distributions modeled as both, multilayered parametrizations of known distributions or nonparametric models through adversarial neural networks. The adversarial models capture multi-modal and long tail distribution of service times. This approach allows us to characterize the service times' independent dynamics, allowing for exogenous events to be characterized implicitly. As a focus application area we provide the first deep and nonparametric solution for predicting unconfirmed transactions in the Bitcoin Mempool network. As a fourth contribution, different from point processes, we studied switching dynamical systems by exploiting recurrent neural networks (RNNs). This approach allows for an explicit description of non-linear and non-Markovian transition functions for both modes and switching dynamics. Indeed, within our model the modes are learned through independent RNNs whereas, similar to (mixture of) expert systems, the selection of modes is handled via a categorical distribution. As a fifth contribution, we incorporate the knowledge of semantic structures and their influence in dynamical processes. In the context of question answering sites, where knowledge is organized as a series of tags identifying questions experts domains, we propose an algorithm to learn hierarchical taxonomies. This algorithm considers co-occurrences of the tags assigned to the questions. Our algorithm infers hidden hierarchies only from sets of co-occurring tags, i.e. from all the n-tuples a given tag appears. We finally link the taxonomies with the dynamical behavior of the users posting questions. Our extensive empirical evaluation indicates that the tagging process of parent nodes is highly dependent on the tagging process of their descendants and not only of its co-occurring tags. As a final contribution, dynamical processes are studied not on population behavior, but on semantic spaces themselves, for the purpose of explaining classifier decisions. We aim at generating a set of examples that highlight differences in the decision of a black-box model. We use interpolations in latent space to generate a set of examples in feature space connecting the misclassified and the correctly classified points. We then condition the resulting feature-space paths on the black-box classifier's decisions via a user-defined functional. Optimizing the latter over the space of paths allows us to find paths that highlight classification differences. We introduce and

formalize the notion of *stochastic semantic paths*: stochastic processes on feature space created by latent code interpolations. Expected changes of a data point are characterized in terms of stochastic functionals along the path, which leads to the notion of a semantic Lagrangian. To train, say, a Variational Auto-Encoder, one must thus define a new training cost by solving the variational problem, which minimizes the functional along the paths.

Zusammenfassung

Die vorliegende Dissertation befasst sich mit der Entwicklung von Inferenzalgorithmen und Methoden zur Untersuchung dynamischer Datensätze. Wir haben Techniken entwickelt, um Zeitreihendatensätze für Punktprozesse, das Umschalten dynamischer Systeme und die Dynamik von Warteschlangen zu analysieren. Darüber hinaus entwickelten wir eine Analyse des dynamischen Populationsverhaltens und wie semantische Strukturen dieses Verhalten beeinflussen. Umgekehrt haben wir untersucht, wie die Dynamik in semantischen Räumen genutzt werden kann, um die Entscheidungen von Black-Box-Klassifikatoren zu erklären.

Als erstes Projekt haben wir den Hawkes-Prozess erweitert, indem wir reichere Korrelationen in die Anregungen einbezogen haben. Dies geschieht durch die Einführung einer Sigmoid-Link-Funktion und eines Gaußschen Prozess Priors. Wir haben einen Polya Gamma-Datenerweiterungsansatz und eine Näherungsmethode zur Verdünnung Gaußscher Prozesse benutzt, um durch eine variationelle Mean Field Methode eine untere Schranke der Likelihood zu berechnen. Dies ermöglicht eine Inferenz mit analytisch berechenbaren Iterationen, welche zu einem schnellen und skalierbaren Algorithmus führen.

Zweitens haben wir eine flexible Methode zur Behandlung der Intensitäten zeitlicher Poisson-Prozesse entwickelt, die auf der Spline-Interpolation basiert und eine schnelle und skalierbare unüberwachte Analyse von Punktprozessen ermöglicht. Wir schlagen dann ein Ähnlichkeitsmaß für Zeitreihen vor, welches invariant gegen eine Verschiebung lokaler Muster ist. Mit diesem Ähnlichkeitsmaß entwickeln wir einen spektralen Clustering-Algorithmus der durch eine flexible, stückweise Kern-Auswertung eine effiziente Berechnung erlaubt, welche auf große Datenmengen skaliert. Das Clustering-Verfahren beinhaltet ein Entropiemaß, welches es ermöglicht zu bestimmen, wie gut eine bestimmte (Intensitäts-) Zeitreihe durch einen Cluster-Prototyp dargestellt wird. Hierdurch können Ausreißer innerhalb einer zeitlichen Stichprobe der Muster erkannt werden.

Drittens bieten wir eine Deep-Learning-Lösung für Servicezeiten von Warteschlangensystemen an. Wir wenden die Fähigkeit tiefer neuronaler Netze, Repräsentationen zu lernen auf Punktprozesse an, um auf die Verteilungen von Servicezeiten zu schließen. Diese werden sowohl als mehrschichtige Parametrisierung bekannter Verteilungen als auch als durch nichtparametrische Modelle mit Generative Adversarial Networks (GAN) modelliert. Die GAN Modelle erfassen die multimodale und Langzeitverteilung der Servicezeiten. Dieser Ansatz ermöglicht es uns, die unabhängige Dynamik der Servicezeiten zu charakterisieren. Hierzurch können exogene Ereignisse implizit charakterisiert werden. Als Schwerpunktanwendung entwickeln wir die erste umfassende und nichtparametrische Lösung der Vorhersage unbestätigter Transaktionen im Bitcoin Mempool-Netzwerk.

Im vierten Beitrag untersuchten wir, im Gegensatz zu Punktprozessen, das Umschalten dynamischer Systeme zwischen verschieden Moden durch die Nutzung rekurrenter neuronaler Netze (RNNs). Dieser Ansatz ermöglicht eine explizite Beschreibung nichtlinearer und nicht-markovscher Übergangsfunktionen sowohl für die Moden als auch für die Schaltdynamik. In unserem Modell werden die Moden durch unabhängige RNNs gelernt, während die Auswahl der Moden ähnlich wie bei einem Mixture of Experts Modell durch eine kategoriale Verteilung erfolgt.

Im fünften Beitrag beziehen wir Wissen über semantische Strukturen und deren Einfluss in dynamische Prozesse ein. Im Kontext von Fragen-und-Antworten-Websites, auf denen Wissen durch eine Reihe von Markierungen organisiert ist, welche die Domänen von Anfrageexperten identifizieren, schlagen wir einen Algorithmus zum Erlernen hierarchischer Taxonomien vor. Dieser Algorithmus berücksichtigt das gleichzeitige Auftreten der den Fragen zugewiesenen Markierungen. Unser Algorithmus leitet verborgene Hierarchien nur aus Sätzen von gleichzeitig auftretenden

Markierungen ab, d.h. aus allen n -Tupeln, in denen ein bestimmte Markierung auftritt. Wir verknüpfen schließlich die Taxonomien mit dem dynamischen Verhalten der Benutzer, die die Fragen stellen. Unsere umfassende empirische Auswertung zeigt, dass der Markierungsprozess der übergeordneten Knoten in hohem Maße vom Markierungsprozess ihrer Nachkommen und nicht nur von den gleichzeitig auftretenden Markierungen abhängt.
Im letzten Beitrag werden dynamische Prozesse nicht auf das Verhalten von Populationen, sondern auf semantische Räume selbst angewandt. Hierdurch wollen wir Klassifikatorentscheidungen erklären. Unser Ziel ist es, eine Reihe von Beispielen zu generieren, die Unterschiede in der Entscheidung eines Black-Box-Modells aufzeigen. Wir verwenden Interpolationen im latenten Raum, um eine Reihe von Beispielen im Merkmalsraum zu generieren, die die falsch klassifizierten und die korrekt klassifizierten Punkte verbinden. Anschließend konditionieren wir die resultierenden Merkmalsraum-Pfade mittels einer benutzerdefinierten Funktion auf die Entscheidungen des Black-Box-Klassifikators. Durch die Optimierung des Funktionals über den Raum der Pfade können Pfade gefunden werden, die Klassifizierungsunterschiede hervorheben. Wir führen den Begriff textit stochastische semantische Pfade ein und formalisieren diesen als stochastische Prozesse im Merkmalsraum, die durch latente Code-Interpolationen erzeugt werden. Erwartete Änderungen eines Datenpunkts werden durch stochastische Funktionale entlang des Pfades charakterisiert, was zum Begriff einer semantischen Lagrange-Funktion führt.
Um beispielsweise einen Variationellen-Auto-Encoder zu trainieren, müssen daher neue Trainingskosten durch Lösen des Variationsproblems definiert werden, wobei die Funktion entlang der Pfade minimiert wird.

Acknowledgements

First and foremost, I would like to dedicate this work to my family, Mom, Dad, Mami, Mildred and Abuela, which always remain positive among the hardships. To my brother Augusto for his true support, queer eye, and maturity. This work would never have been possible without the advice and words of my supervisor Prof. Christian Bauckhage, thanks for believing in science and creativity. To Prof. Manfred for his wisdom and good stories. All the people in Fraunhofer, but especially to Kostadin, Bogdan, and Jannis. And of course, to Ramses, because one can do research while having a beer.

Finally, to Marie, thank you for the love.

Berlin, 26rd of June 2020

Contents

Chapter 1

General Introduction

The present dissertation's underlying common theme is the development of inference algorithms and methodologies for the study of dynamical datasets. The study of dynamical phenomena has a rich history that underlies the development of modern science. If one is to identify Newton's publication of *Philosophiae Naturalis Principia Mathematica* as the cornerstone of the scientific revolution, it is at is core an acknowledgment of the impact of the study of dynamical systems in science in general. The success of Newton's Principia is a success of our ability to forecast natural phenomena. The introduction of the *Laws of Motion* and the formalism of classical mechanics provide a recipe for how to construct mathematical models that can describe the evolution of physical systems. The modern formulations of such laws, through Hamiltonian and Lagrangian mechanics, dictate that one must introduce prior knowledge in the form of Hamiltonians or Lagrangians to model the evolution of such systems. If one is to provide a genuinely agnostic take on the dynamics of complex systems, one must make away with such prior knowledge.

After humble origins in Statistics, as Information Theory and Machine Learning entered stage halfway through the past century, a change in science's philosophy slowly started to take ground. Instead of imposing human preconceptions such as geometry, conservation laws, and symmetries, the effort shifted to learning. Now, one is possed essentially with the two main tasks. First, developing models that can learn from data and uncover the behavior of the physical systems. Moreover, another of training, where one develops methods to find the right parameters and hyperparameters, which tune the defined model to the data at hand. Once our attention is focused on learning, one is concerned now on providing flexible models that can gain the most of the data at hand. In a way, the evolution of machine learning is the evolution of our capacity to provide models and learning algorithms that are flexible enough to handle data.

There are two main organizing principles behind the current work: data and learning methods—different methodologies applied to different datasets. The original force behind these distinctions is the nature of the datasets. The success of the different models and learning algorithms can be traced back to data as one must pay the price of flexibility with data. The sparser the data sets, the more a priori knowledge one is forced to include in the models. This sparsity can have obscure forms as large data sets can be sparse along different coordinates. In the present work, three different methodologies to endow agnosticism and flexibility in dynamical models are utilized. Bayesian nonparametrics, Deep parametric models and adversarial neural networks.

1.1 Learning Methodologies

1.1.1 Non Parametrics

A more classical approach uses Bayesian nonparametric to allow for flexible, functional families by sampling from gaussian processes. These distributions are trained via approximate inference in a scalable fashion with variational methods and sampling from the exact posterior via Markov chain monte carlo. The approximate variational inference developed in the current work permits fast inference due to analytical forms of the learning procedure's update steps. One could obtain analytical forms by extending the model likelihood with synthetic variables, which renders the model tractable. These auxiliary variables are devoid of physical meaning but create conjugate forms for the likelihood which are amenable to analytical computation. This methodology is introduced for a particular point process model in Chapter 2.

1.1.2 Deep Parametric Models

Latter, we introduce flexibility in the distribution families with the use of general approximation methods for functions, namely deep neural methods which parametrize classical statistical distributions. These approaches are also trained via approximate surrogate distributions with the aid of the variational methodologies. The method introduced here is line with that of variational autoencoders, in which one hidden random variable is trained via approximate inference. However, the deep neural network models' parameters are obtained by maximizing the lower bound, obtaining maximum likelihood estimates of such parameters. Unlike the autoencoder formalism, our goal here was not to create encodings or semantic representations of the data. Our contribution consists of exploiting these existing neural variational methods to create new interpretable dynamical models. This formalism was introduced in Chapter 3.

1.1.3 Generative Adversarial Models

Adversarial generative networks can dispense with any distributional form via an implicit methodology. Samples are provided instead of approximating distributions. They provide a novel form of training in which a two-player game is defined where two neural networks compete. One neural network is poised with the task of generating data which resembles the observed data. The other one is endowed with the task of differentiating the real and generated datasets. By training in a two-step matter the two players of the game, an equilibrium point is achieved in which the generator can reproduce the data. We make use of current results in Optimal Transport Theory and provide a solution through Wasserstein's Gans. The dynamic aspect is included with conditional gans, which levers dynamical representations obtained via nonparametric form of transitions functions. Namely, recurrent neural networks. We provided a novel GAN model for the studies of service times in the framework of queueing theory in Chapter 4

1.2 Datasets

1.2.1 Point processes

Point processes account for discrete data points in continuous time. Historically introduced as models for the location of stars, the Poisson process was later introduced in statistics with Erlang's studies of the incoming call in phone call centers. We studied these processes from two angles,

mainly. One in Chapter 2 where we impose a rich nonparametric model in order to exploit prior knowledge as to extract meaningful information from sparse data. Furthermore, a different approach of an unsupervised algorithm, where data is in the aggregate, is rich but sparse on the individual (per time series) level. The first methodology is intended to provide a richer model and leverages the prior forms' Bayesian advantages. In a sense, one can incorporate expert knowledge in the prior that allows for a meaningful treatment of sparse data. In this particular case, we endow the Hawkes process intensities with a correlation structure. In chapter 5, we provide an unsupervised algorithm for the clustering of point processes.

1.2.2 Service Times

Within the family of point process models, in queues systems, one is interested in the interplay between two phenomena: customer arrivals and service times to these costumers. In the application set up, data is abundant, and we, therefore, resourced to a deep and adversarial solution, flexible enough to capture rich forms in the data distribution. The primary heuristic behind this contribution is to leverage the representations provided within the recurrent formalism to encode the customer's process's dynamical information, allowing us to construct a conditional structure to exploit that representation for the sake of the service time distribution. These methodologies are presented in Chapter 4.

1.2.3 Switching Dynamical Systems

Time series provide a challenging ground for machine learning algorithms, as long term dependencies and nonlinearities are usually present in empirical datasets. Methods are required, which highlight the dynamical character of the problem. One classical approach to characterize the time series is to introduce the notion of dynamical regimes. If one is studying hearth beat data, for example, one would like to differentiate arrhythmia patterns from normal behavior in a particular patient hearth. To highlight this natural compositionality, similar to clustering algorithms, one can index the different regions of the time series through a categorical variable. In this dissertation, we provide a deep network solution for such a problem. The use of deep learning parametrization of the transition functions will characterize complex patterns of non-markovianity and nonlinearity, as present on empirical data sets. We present our contribution in Chapter 3.

1.2.4 Semantics Related Process

Instead of analyzing time series directly, we focus on semantic representations. Either those inferred from population behavior, or obtained as latent codes provided by dimensionality reduction technique such as Variational autoencoders. Although a learning algorithm is provided, the focus is on the relationship among knowledge structures provided by a population of its consequent dynamical behavior. Conversely, we obtained semantic structures or representation imposing properties on the dynamical process defined on top of these representations. This themes are developed n chapter 6 and Chapter 7.

1.3 Thesis outline

The present work is intended to be self-contained in that every inference method and the basic theory of point process and switching dynamical process are presented. It divide in three parts, the first for Chapter 2, 3 (concerning Bayesian inference). The adversarial algorithms, as well as the unsupervised algorithms, are explained in chapter 4, 5. Finally, modeling of process related

to semantic structures is presented in part 3, chapter 6. For each part, we provide the theoretical background as required for the inference methodologies, the specific contributions are presented in each chapter, providing the specific theoretical backgrounds of the models in the individual Chapters.

Part I

Bayesian Inference for Stochastic Processes

The main goal of machine learning is to learn from data. In order to do so, one must introduce models which are able to generalize and generate from the data. If one adhere to the statistical framework, variations of the data are though of to arise from the statistical nature of the underlying data distribution. Let $\mathcal{D}$ be the data at hand, one defines a model to train as composed of other hidden variables $\mathbf{Z}$ which are part of some generative process which gives rise to the data, and parameters θ of the model. One can then apply Bayes rule to obtain probabilities over the hidden variables, that is:

$$P_\theta(Z|\mathcal{D}) = \frac{L_\theta(\mathcal{D}|Z)P_\theta(Z)}{P_\theta(\mathcal{D})}, \tag{1}$$

where $L_\theta(Z|\mathcal{D})$ is known as the likelihood, the probability that the current model generated the data. $P(Z)$ is our prior belief for the hidden variables, and $P(\mathcal{D})$ is the evidence [1]. In this formulation, we included separated variables for the model parameters θ which are not part of the generative model. It is important to notice however, that a fully Bayesian model, includes this variables as hidden variables, and posteriors whould be calculated for each one. The obtained result $P_\theta(Z|\mathcal{D})$ is known as the posterior distribution of the hidden variables. Its importance resides in the fact that it allows us to perform predictions. For an unobserved data point X^* one would like to obtain the probability of this data point under the trained model. This is obtained by:

$$P(X^*|\mathcal{D}) = \int L_\theta(X^*|Z)P_\theta(Z|\mathcal{D})dZ, \tag{2}$$

for the practicioner, the initial step consist in defining both the likelihood and the prior. In such a way that one is able to include known characteristics of the data. In order to fully especify the posterior then, one must perform

$$P_\theta(Z) = \int L_\theta(\mathcal{D}|Z)P(Z)dZ = \int P(X,Z)dZ. \tag{3}$$

Except for very specific cases, one is not able to perform such integral. In this case, the practiticioner must resource to *approximation methods*, which are able to get testimates of the posterior. Sampling methods, shuch as markov chain monte carlo, allow us to obtain samples $\{Z^i\}_{i=0}^M$ from the posterior distribution and approximate the predictive distribution as:

$$P(X^*|\mathcal{D}) \sim \frac{1}{M}\sum_{i=0}^{M} L(X^*|Z^i), \tag{4}$$

another class of approximating method, the variational kind, resource to a rather different strategy. One must define an approximate distribution $Q(Z)$ which is tractable. This distribution must approximate the posterior. This is enforced by minimizing the Kullback Leibler divergence $\mathrm{KL}(Q(Z)||P(Z|\mathcal{D}))$. Since this quantity is not explicitly available as the posterior is unknown. Variational approximate inference requiere us to maximize the following bound, in order to obtain the approximating distribution.

$$\mathcal{L} = \mathbb{E}_{Q(Z)}\left[L_\theta(\mathcal{D}|Z)\right] - \mathrm{KL}(Q(Z)||P(Z)) \tag{5}$$

for the first term, one must perform averages of the likelihood over the approximate distribution. The second term, enforces the posterior distribution over the likelihood and plays the role of a rgularizer alowwing us to avoid overfitting and hence permiting generalization. We now present a general overview as to how to obtain flexible models.

[1] in physics this is known as the partition function

Non Parametrics

For several application areas, in particular in the study of dynamical systems and temporal dynamics, one must obtain generative models that are able to capture the behavior of functions with support on the real line. Gaussian Processes arise as one of the main tools within the Machine learning community, which allows the practitioner to define priors over a family of functions. Due to the functional nature, one can think of these models as estimating the value of an infinite set of variables, the unknown function's values. The infinite nature of these priors and posteriors are the reason as to why these methodologies are known as nonparametric. Under the Gaussian process formalism, one assumes that for every finite set of points $\{(t_i, f_i)\}$ where t_i is defined over the function support and f_i is the value of the function attained at this point. The values are sampled from a multivariate gaussian distribution.

$$(f_1, ..., f_N) \sim \mathcal{N}(\mu, K), \tag{6}$$

where K is defined with the aid of a kernel $k(\cdot, \cdot)$ and $K_{ij} = k(t_i, t_j)$, the mean is given by a continuos function evaluated at the support poitns $\mu(t_1), \mu(t_2), ..., \mu(t_N)$. The appeal of the gaussian process

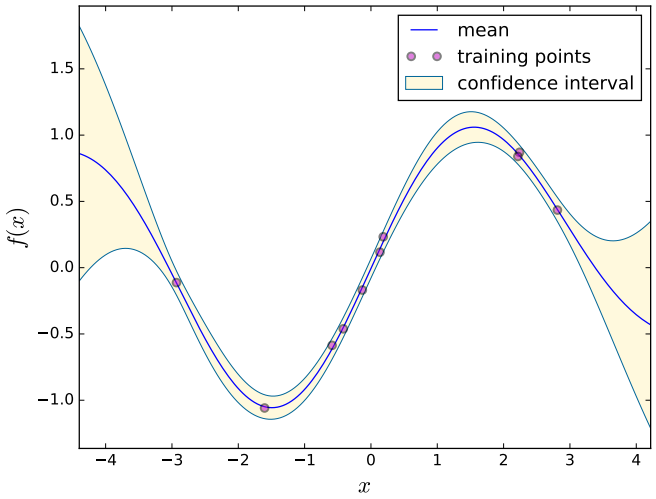


Figure 1 – Sample from a Gaussian Process and the value of its mean and variance

formalism, lies the close form of expression which one is able to obtain for the model posteriors and marginal likelihoods.

$$p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}\right) = \int p\left(\mathbf{y} \mid \mathbf{f}\right) p\left(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\theta}\right) d\mathbf{f} = \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_N + \sigma^2 \mathbf{I}_N\right)$$

Predictive distribution of a new point $\mathbf{x}$

$$p\left(y \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta}\right) = \mathcal{N}\left(y \mid \mu\left(\mathbf{x}\right), \sigma\left(\mathbf{x}\right)\right)$$

Mean $\mu\left(\mathbf{x}\right) = \mathbf{k_x}^T \left(\mathbf{K}_N + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}$ and Covariance $\sigma\left(\mathbf{x}\right) = K_{\mathbf{xx}} - \mathbf{k_x}^T \left(\mathbf{K}_N + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{k_x} + \sigma^2$

Deep Parametric Models

In recent years, deep neural networks have arisen as successful models that are able to provide great flexibility and leverage the information content of modern massive data sets. Aided by power GPUs computer architectures, deep neural networks have shown enormous success in areas as Image Classification, Speech Recognition, Natural Language Processing such as translation and sentence

classification, etc. One of the modern attempts to exploit the flexible deep models within the Bayesian framework was obtained by (Kingma and Welling 2013a). In its seminal work, a generative data model is defined in which the likelihood is given by a Multivariate Gaussian distribution parametrized via deep neural networks. For this generative model, one hidden variable Z is defined as the latent code representing a semantic representation of the data. Inference is performed by maximizing the variational lower bound, and deep parametrizations of the multivariate gaussian also define the posterior. This is a Bayesian version of the traditional autoencoder, but where the code is handled through distributions. The inference is bayesian respect to the hidden code, but the neural network parameters are obtained via maximum likelihood.

From the application perspective, the main caveat to optimize the bound 5 in the variational autoencoder formalism, lies in the average required for the hidden variables Z posterior. The representation of variational posteriors as individual variational parameters becomes a big burden to optimization, to overcome this, they are now defined as a transformation over the data. This allows us to reduce the number of parameters for optimization (which no longer grows linearly with the size of data) and to perform fast inference at test time. We introduce a differentiable transformation $\tilde{z} = g_\phi(\epsilon, x)$. And sample an auxiliary noise $\epsilon \sim p(\epsilon)$. The standard approach uses a deep neural networks in order to parametrize multivariate gaussian distributions, i.e. $Q(Z) = \mathcal{N}(\mu, \sigma)$. In this case, one is lead to:

$$z = \mu + \sigma \cdot \epsilon, \tag{7}$$

where $\epsilon = \mathcal{N}(0, 1)$. A multivariate gaussian distribution.

Chapter 2

Self Exciting Point Processes

Sequences of self-exciting temporal events are frequent footmarks of natural phenomena: Earthquakes are known to be temporally clustered as aftershocks are commonly triggered after the occurrence of the main event (Ogata 1988); in social networks, the propagation of news can be modeled in terms of information cascades over the edges of a graph (Zhao et al. 2015a); and neural activity is frequently studied as a set of spike trains modeled via a network of synaptic connectivity (Linderman and Adams 2015). The Hawkes process (Hawkes and Oakes 1974), a type of clustered Cox process, provides a tractable model that can incorporate both *exogenous* events, as well as a causal relationship through self-excitation (*endogenous*) events. It is defined as a double stochastic Poisson process, in which each new event (or *arrival*) contributes to the intensity function via a memory kernel which encodes how the excitation contribution changes over time.

This simple model is usually extended by incorporating excitations variables that describe how each arrival distinctively affects future events and imposing a notion of causality in multivariate versions of the process. Popular models for earthquake occurrence incorporate spatial correlations through the excitations. Recently temporal correlations were introduced via a stochastic process in the excitations functions (Lee et al. 2016).

In this chapter, we incorporate richer correlations in the excitations via the introduction of a sigmoid link function over a Gaussian process prior. Gaussian processes have been shown to provide flexible and successful models to model intensity function in point process, also known as the sigmoid Gaussian process (Samo and Roberts 2015a; Lloyd et al. 2015). As well as providing a flexible framework for inference methods in stochastic processes (Archambeau et al. 2007). The Gaussian process formalism has the added advantage of allowing the practitioner to incorporate expert knowledge through selecting kernels, which models specific functional form (Rasmussen and Williams 2006).

The introduction of the nonlinear sigmoid function, required to ensure a positive form of the intensity function, renders the inference problem cumbersome as this leads to non tractable lower bounds for approximate inference. Recent work on data augmentation for logistic regression resolve this difficulty expressing the sigmoid function as a marginal over Pólya-Gamma random variables (Polson et al. 2013; Linderman et al. 2015). We incorporate this data augmentation approach and a sparse Gaussian process approximation into a mean field treatment of the variational lower bound to perform inference with analytic updates, obtaining a fast and scalable algorithm that is amenable for big data sets.

In the following section, we describe our model and introduce the algorithm for its simulation. Next, we introduce the augmented variable formalism that extends the likelihood and present the variational approach required for inference. Finally, we describe our synthetic and empirical data results in neural activity from calcium fluorescence recordings.

2.1 Background

We start by providing the basic theory of point processes. Later we introduce our model as a modification of the known Hawkes process. This section will be of interest in chapter 4 concerning the analysis of queues systems and chapter 5 where we develop an unsupervised methodology for poisson process clustering.

2.1.1 Point Processes

We start by introducing the concept of a point process. The exposition here is intended to be rather explanatory or phenomenological in character. One needs to remember that a stochastic process is defined as a collection of random variables endowed with an index (Gallager 2012). The aim is to characterize a sequence of event in continuos time. Let $\mathbf{T} = \{t_1, t_2, ..., t_N \mid t_j < t_{j+1} \text{ where } t_j \in \mathbb{R}^+\}$ be the time events also known as arrivals. These represent a particular *realization* of the process. To characterize such set of events one is confronted with different possibilities. We can study the inter event times defined as $\tau_i = t_{i+1} - t_i$. Where τ_i is another random variable. Given τ_i the arrival process is defined by:

$$t_i = \sum_{i=1}^{i} \tau_i, \tag{8}$$

another possibility is to study the counting process defined by the number of events at time t, namely: $N(t)$. If we let $N(t, t+\delta)$ be the number of arrivals in a period δ, then we can further characterize the process in terms of a positive measurable function $\lambda(t)$ also known as the *rate* or *intensity* of the process. Given $\lambda(t)$, the probabilities for $N(t, t+\delta)$ are given by

$$Pr\{N(t, t+\delta) = 1\} = \lambda(t)\,\delta + o(\delta^2) \tag{9}$$

$$Pr\{N(t, t+\delta) = 0\} = 1 - \lambda(t)\,\delta + o(\delta^2) \tag{10}$$

$$Pr\{N(t, t+\delta) \geq 2\} = o(\delta^2) \tag{11}$$

where $o(\delta^2)$ accounts for negligible contributions. Intuitively, $\lambda(t)$ provides us with the probability that a given arrival occurs within an infinitesimal time window δ. The different members of the family of the point process formalism are defined by specifying these different process or random variables $(\tau, N(t), \lambda(t))$. For example, the **poisson process** is given when the inter arrival distribution follow an exponential law:

$$P(\tau = \tau) = \alpha e^{-\alpha\tau}, \tag{12}$$

which is equivalent to a constant rate λ. Furthermore, one can also show, that the counting process for the poisson process is given by the poisson distribution:

$$P(N(t) = N) = \frac{(\alpha t)^N}{N!} e^{-\alpha t}, \tag{13}$$

the poisson process owes its name to this distribution, known as the poisson law, first observed in the studies of populations (Daley and Vere-Jones 2007a). We can further extend the notion of poisson process by allowing $\lambda(t)$ to be a funtion in time. This allows us to define an *inhomogeneous poisson processes*. If one introduces $m(t) = \int_0^t \lambda(x)dx$, the counting random variable follows the law:

$$P(N(t) = N) = \frac{(m(t))^N}{N!} e^{-m(t)}. \tag{14}$$

For the inhomogeneous poisson process, the concept of interarrival time is better understood in terms of the survival function. To facilityte further generalization of the process as requiered below,

we now condition the intentisity on some object $\mathbf{x}$. Now, Eq. 14 allows us to specify the probability of no event ocurring in a given time interval i.e. $N(t) = 0$

$$S(t|\mathbf{x}) = \exp\left(-\int_0^t \lambda(t'|\mathbf{x})dt'\right) = 1 - F(t|\mathbf{x}), \tag{15}$$

where $S(t|\mathbf{x})$ denotes the conditional survival function of the process, and $F(t|\mathbf{x})$ the cumulative conditional density distribution. Under the framework of survival functions, the intensity function is also knwon as the *hazard function* and its related to the other variables via the expresion:

$$\lambda(t|\mathbf{x})dt = \frac{f(t|\mathbf{x})dt}{S(t|\mathbf{x})} = \frac{f(t|\mathbf{x})dt}{1 - F(t|\mathbf{x})}. \tag{16}$$

where $f(t|\mathbf{x})$ is the conditional density function. Finally, this will allow us to write the conditional density function for the next arrival to happen at time t reads:

$$f^*(t|\mathbf{x}) = \lambda^*(t) \exp\left\{\int_{t_i}^t \lambda^*(t')\, dt'\right\}. \tag{17}$$

2.1.2 Cox and Cluster Process

As stated above, one can further generalize the point process by introducing different structures on the intensity. For a doubly stochastic process, also known as the Cox process. The intensity $\lambda(t)$ is given by an stochastic process. Once the notion of a stochastic intensity is introduced. One is in the position to introduce a self exciting property. Interestenly enough, if one is to use the processes themselves as objects upon with to condition. One can speack of cluster process. Here the aim is to specify a process given another point process. So one can consider a series of steps. First, an initial process is laid down and then a secondary pro- cess is defined, with distributions conditional on the realization of the initial process.

2.1.3 Hawkes Processes

In this section, we outline the main characteristics of our model. A Hawkes process or self exciting point process, is a type of Cox process (Kingman 1993) which defines self excitations through a cluster representation (Daley and Vere-Jones 2007b) around *exogenous events*. Let $\mathcal{T}_T = [0, t] \in \mathbb{R}$. We define the counting measure $N(\mathcal{T}_t)$ as the number of arrivals in the sequence $\mathcal{H}_t = \{T_1, ..., T_{N(\mathcal{T}_t)} : T_i \in \mathcal{T}_t \wedge T_{i-1} < T_i\}$ where $\mathcal{H}_t$ defines the history of the process until time t. The counting measure $N(\cdot)$ has an associated intensity defined as

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\mathbb{E}[N(\mathcal{T}_{t+\Delta t}) - N(\mathcal{T}_t)|\mathcal{H}_t]}{\Delta t}. \tag{18}$$

Following (Lee et al. 2016; Rasmussen 2013), the intensity is given by

$$\lambda(t) = \lambda_0 + \sum_{T_i : t > T_i} Y(T_i)\mu(t - T_i), \tag{19}$$

where λ_0 is the homogeneous base intensity, $Y(T_i)$ is called the excitation and $\mu(\cdot)$ is the memory kernel defining the change in the excitation value for each arrival. The notion of clustering and branching structure [1] is easily understood in terms of the superposition theorem for Poisson processes (Kingman 1993) as one can interpret each term in Eq. 19 as an independent Poisson process. The term λ_0 will generate a set of arrivals of *exogenous* nature (also known as *immigrants*) which occur independently of others arrivals and define the cluster centers. The other terms in Eq. 19 are of

[1] Clusters refers to a type of Cox Processes which

endogenous nature, as they depend on arrivals T_i in the history $\mathcal{H}_t$. Consequently, we define the *offspring process* Φ_i as the Poisson process defined from arrival T_i with intensity function given by $\lambda_i(t) = Y(T_i)\mu(t - T_i)$. This gives a sequential relationship similar to a branching process, because each arrival can be seen as the parent node of the arrivals generated under its intensity λ_i. We say that T_j has an *ancestor* T_i of order n, if there is a sequence $s_1, s_2, ..., s_n$ where $s_k \in \Phi_{k-1}$ and $s_1 = T_i$ and $s_n = T_j$. Equivalently we say that T_j belongs to the nth generation of T_i. The *total offspring* of event T_i will be the process given by $C_i = \{T_j : T_j \text{ belongs to the } n \text{ generation of } T_i \ , n \in \mathbb{N}_0\}$. In order to guarantee numerical stability, one must establish conditions by which the number of events in a cluster $S = |C_i|$ remain finite. Similar to the Galton-Watson branching process (Watson and Galton 1875), one is required to study the *offspring distribution* (probability for the number of possible offspring's). In the context of the Poisson process Φ_i this will be obtained via $\nu = \mathbb{E}[\int_{T_i}^{\infty} \lambda_i(t)dt]$. This quantity is of interest as it allows us to define the average size and length of the clusters. Letting S be the average size of the cluster, it is given by $\mathbb{E}[S] = 1 + \nu + \nu^2 + ... = \frac{1}{1-\nu}$ where we impose $\nu < 1$ in order for the summation to be finite; this, in turn, requires $Y_i < 1$[2].

2.2 Sigmoid Gaussian Excitations

Traditionally, the excitations $Y_i = Y(T_i)$ are assumed to be constant or to be drawn i.i.d. from a fixed distribution (Møller and Rasmussen 2005) which associates a *mark* with each offspring. Here, however, we incorporate serial correlations of the excitations by requiring the $Y(t)$ to be parametrized as $Y(t) = \sigma(f(t))$, where $\sigma(x) = (1 + e^{-1})^{-1}$ and placing a Gaussian process prior over $f \sim \mathcal{GP}$. Gaussian processes (Rasmussen and Williams 2006) provide a non parametric approach, allowing us to have a distribution over possible functions and thus to obtain more flexible models. Similar to previous work with Gaussian Cox processes (Adams et al. 2009), the function σ guarantees that we meet conditions for the cluster size ($\mathbb{E}[S]$) to converge, as well as a positive intensity function. On the other hand, Gaussian process priors provide a great degree of flexibility in the temporal dependencies of the excitations.

2.3 Likelihood

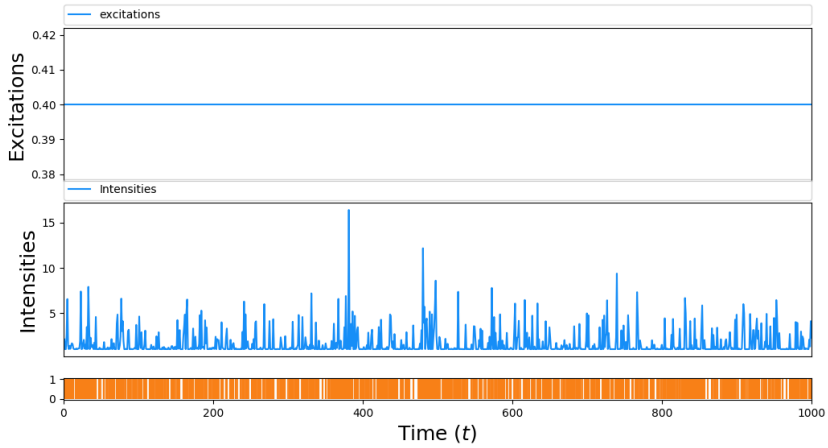
Formally, the likelihood is written down as an inhomogeneous Poisson process between arrivals, conditioned on the history of arrivals $\mathcal{H}_{t_i}$ (Daley and Vere-Jones 2007b). We will now use expression Eq. 14, but since we are interested in the self excitations properties, we subsititute the dependance on $\mathbf{x}$ with the history. For one-dimensional processes the conditional likelihood that the next arrival happens at time, t reads:

$$f^*(t|\mathcal{H}_i) = \lambda^*(t) \exp\left\{\int_{t_i}^{t} \lambda^*(t')\, dt'\right\} , \tag{20}$$

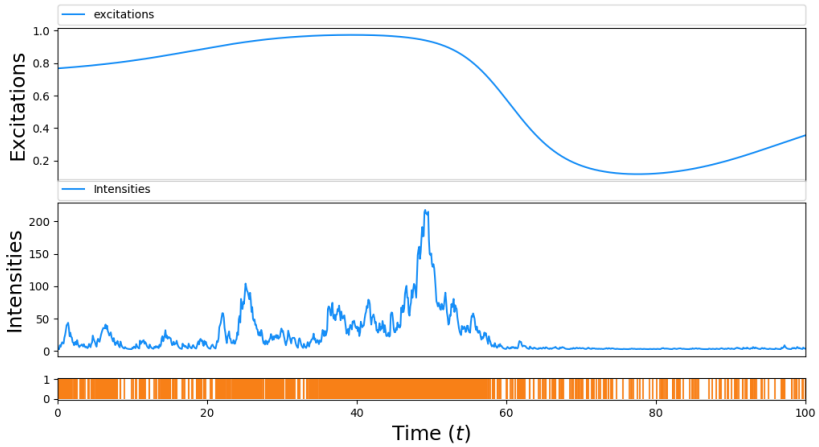
where the intensity function λ^* is a (locally) integrable function. The joint distribution will be given by $P(\mathbf{T}) = \prod_i f^*(t_i|\mathcal{H}_{t_{i-1}})$. If one expands directly using Eq. 19, products appears between the arrivals. In order to obtain a tractable form of the likelihood, we exploit the branching structure by introducing random variables $Z_{ij} = \mathbb{1}[T_i \in \Phi_j]$ which indicates whether event T_i is an offspring of event T_j and $Z_{i0} = 1$ if event T_i is and immigrant i.e., if it was generated from the base intensity λ_0 in Eq.19. Let $\mathbf{T}$,$\mathbf{f}$,$\mathbf{Z}$ represent the set of arrivals $\{T_i\}$ (data), the excitation functions $\sigma(f_i) = Y(T_i)$ and the branching structure Z_{ij}. Following (Lee et al. 2016), we express the likelihood function at time T as

$$P(\mathbf{T}|\mathbf{Z}, \mathbf{f}) = e^{-\Lambda_T} \prod_{i=1}^{N_T} \lambda_0^{Z_{i0}} \prod_{j<i} \times [\sigma(f_j)\mu(T_i - T_j)]^{Z_{ij}} , \tag{21}$$

[2]We define μ such that $\int \mu(t)dt = 1$



(a) Uniform Excitations

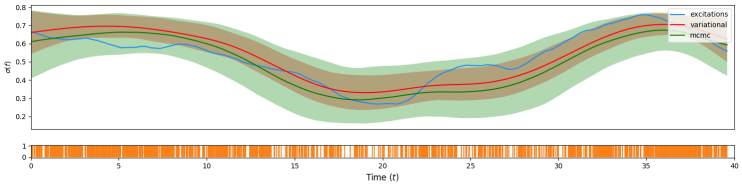


(b) Sigmoid Gaussian Process Excitations

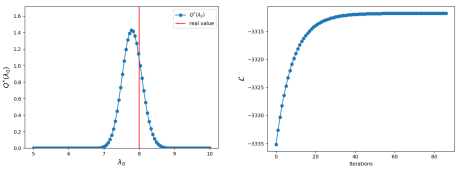
Figure 2 – Sample Hawkes process simulation based in both Constant and Gaussian excitations with a exponential memory kernel. First row corresponds to excitations (Y), second row to intensities λ and third row to arrivals $\mathbf{T}$

where the compensator was defined as $\Lambda_T = \int_0^T \lambda_v dv$. By definition, for each i, there is only one $j < i$ for which $Z_{ij} = 1$. The form of the Eq. 21 is not amenable for approximate inference due to the exponential term in the compensator, as well as the functional form of the sigmoid $\sigma(\cdot)$. In order to obtain efficient inference schemes, one would like that the likelihood Eq. 21 to be conjugate to the $\mathcal{GP}$ prior. In order to accomplish this we proceed via data augmentation for the expressions of the compensator and the sigmoid function. We begin our derivation by calculating a explicit form of the compensator Λ_T and note that:

$$\begin{aligned}
\Lambda_T &= \int_0^T \lambda_v dv \\
&= \int_0^T \lambda_0 dv + \int_0^T \sum_{t_i<T} Y_i \mu(v - t_i) dv \\
&= \Lambda_T^0 + \int_0^T \int_s^T Y_s \mu(v - s) dv dN_s \\
&= \Lambda_T^0 + \int_0^T Y_s \phi[s, T] dN_s \\
&= \Lambda_T^0 + \sum_{i=1}^{N_T} \sigma(f_i) \phi[t_i, T]. \qquad (22)
\end{aligned}$$



(a) Inferred Excitations



(b) $(\lambda 0)$ law **(c)** $\mathcal{L}(q)$

Figure 3 – Blue line corresponds to the simulated process, red line corresponds to inferred excitation intensity with the variational approximation, green corresponds to mcmc sampler $\sigma(f)$ and the 95% confidence bounds. Orange lines corresponds to observed arrivals. Sigmoid Gaussian Excitations as obtained for a synthetic data set. We also present the form of the variational approximation for the base intensity, and the variational bound during training.

Here, we define $\phi[t_i, T] = \int_{t_i}^{T} \mu(\tau - t_i)d\tau$ and $\Lambda_T^0 = \lambda_0 T$, since the base intensity is a constant. In the following we make use of the exponential kernel $\mu(t-\tau) = \alpha e^{-\alpha(t-\tau)}$. This memory kernel is widely use in the literature (Ogata 1988; Linderman and Adams 2015; Lee et al. 2016).

2.3.1 Poisson augmentation

We introduce the first set of augmented variables in the the compensator. Using the relationship $\sigma(x) = 1 - \sigma(-x)$ on can rewrite Eq. 22 as

$$e^{-\Lambda_T} = e^{-\Lambda_T^0} \prod_{i=1}^{N_T} \mathbb{E}_{\rho_i} \left[\{\sigma(-f_i)\}^{\rho_i} \right], \tag{23}$$

where the expectation is over independent Poisson variables ρ_i with distribution $\mathrm{Po}(\rho_i|\zeta_i) = e^{-\zeta_i} \frac{\zeta_i^{\rho_i}}{\rho_i!}$,. where we have set $\zeta_i = \phi[t_i, T]$, and we have used

$$e^{\zeta(x-1)} = E_\rho[x^\rho] = \sum_{\rho=0}^{\infty} x^\rho \mathrm{Po}(\rho|\zeta).$$

This allows us to get rid of the exponential term of $\sigma(f)$ due to the compensator. Notice that this expression is equivalent to the *generating functional* for a homogeneous Poisson process (Daley and Vere-Jones 2007b), another form of the Cambell's functional representation (Kingman 1993) for a counting measure $N(\cdot)$.

2.3.2 Pólya-gamma augmentation

To obtain the conjugate form of the likelihood, we must resolve the non-linear form of the sigmoid appearing in the new form of the compensator and the rest of the terms involving the branching structure. The Polya-Gamma augmentation scheme (Polson et al. 2013) achieves this by exploiting the following integral identity

$$c\frac{e^{ax}}{(1+e^x)^b} = c2^{-b} e^{\kappa x} \int_0^\infty e^{-w\frac{x^2}{2}} P_{PG}(w|b,0)dw, \tag{24}$$

where $\kappa = a - b/2$. And $P_{PG}(w|b,0)$ corresponds to the Pólya-gamma distribution. We know use the Polya gamma representation in Equation 23, yielding

$$e^{-\Lambda_T} = e^{-\Lambda_T^0} \prod_{i=1}^{N_T} \mathbb{E}_{\rho_i,\omega_i} \left[2^{-\rho_i} e^{-\frac{\rho_i f_i}{2} - \frac{f_i^2}{2}\omega_i} \right]. \tag{25}$$

2.3.3 The augmented likelihood

We can now express the full form of the likelihood incorporating the augmented variables. In order to do this, we insert the Polya gamma representations for the sigmoid function and Eq. 2521.

$$\begin{aligned} P(\mathbf{T}, \rho, \omega, \mathbf{\Omega}|\mathbf{Z}, \mathbf{f}, \theta) = e^{-\lambda_0 T} \prod_{i=1}^{N_T} \lambda_0^{Z_{i0}} \left[2^{-\rho_i} e^{-\frac{\rho_i f_i}{2} - \frac{f_i^2}{2}\omega_i} \right] \\ \times PG(\omega_i|\rho_i, 0)\mathrm{Po}(\rho_i|\zeta_i) \times \prod_{j<i} \left[2^{-Z_{ij}} \left[\mu(T_i - T_j)\right]^{Z_{ij}} \times e^{\frac{Z_{ij} f_j}{2} - \frac{Z_{ij} f_j^2}{2}\Omega_{ij}} PG(\Omega_{ij}|1,0) \right] \end{aligned} \tag{26}$$

This augmented likelihood incorporates the distribution over the auxiliary variables Ω_{ij}, ρ_i and ω_i. Here we defined $\theta = \{\lambda_0, \alpha, \theta_{\mathbf{K}}\}$, as the base intensity, the decay rate of the exponential kernel and the hyper parameters of the Gaussian process. The definition of the augmented model is completed by noting that the prior P(f) is a Gaussian process with hyperparameters $\theta_{\mathbf{K}}$ (due to our models assumptions) and P(Z) is a uniform distribution (since all prior points are valid parents of the subsequent arrivals).

2.4 Variational Inference

To solve the inference problem completely, one would like to obtain an analytical expression for the posterior $P(\mathbf{Z}, \mathbf{f}, \rho, \omega, \mathbf{\Omega}, \theta|\mathbf{T})$. This is not possible since we are not able to calculate all required marginals over the joint distribution Eq. 26 in order to obtain the model evidence $P(\mathbf{f})$. Instead, we resort to approximate variational inference (Jordan et al. 1999). We will define a tractable distribution family to approximate the posterior. This is achieved by maximizing the lower bound $\mathcal{L}(Q)$ defined below. This procedure will minimize the kullback leibler divergence between the unknown posterior and the proposed approximating distribution. We further exploit the structure of the joint distribution Eq. 26 by assuming independence among some of the variables, in what is known as the mean field approximation (Bishop 2006). The posterior density will be approximated by

$$P(\mathbf{Z}, \mathbf{f}, \rho, \omega, \mathbf{\Omega}|\mathbf{T}) \approx Q(\mathbf{Z}, \rho, \omega, \mathbf{\Omega})Q(\mathbf{f}, \lambda_0) \tag{27}$$

One can show that the structure of the model implies the further factorization $Q(\mathbf{Z}, \rho, \omega, \mathbf{\Omega}) = Q(\mathbf{Z}, \mathbf{\Omega})Q(\omega, \rho)$ and $Q(\mathbf{f}, \lambda_0) = Q(\mathbf{f})Q(\lambda_0)$ this leads to the following lower bound on the evidence

$$\mathcal{L}(Q) = \mathbb{E}_Q \left[\log \left\{ \frac{P(\mathbf{T}, \mathbf{Z}, \mathbf{f}, \rho, \omega, \mathbf{\Omega}, \lambda_0)}{Q(\mathbf{Z}, \mathbf{\Omega})Q(\omega, \rho)Q(\mathbf{f})Q(\lambda_0)} \right\} \right]. \tag{28}$$

Here Q refers to the probability measure of the variational posterior. We can maximize the bound by alternating the maximization over each of the factors (Bishop 2006). The variational calculation will yield the optimal solutions for each factors analytically as

$$\log Q^*(\mathbf{Z}, \mathbf{\Omega}) = \mathbb{E}_{Q(\omega,\rho)Q(\mathbf{f})Q(\lambda_0)}[\log P(\mathbf{T}, \mathbf{Z}, \mathbf{f}, \rho, \omega, \mathbf{\Omega}, \lambda_0)] \tag{29}$$

This results imply that to obtain the optimal distribution of one of the factors, one must calculate averages of the logarithm of the joint distribution over the remaining factors in the approximation.

In the following subsections, we explicitly express the functional form of the optimal distributions, and obtain the corresponding averages required in all the optimal equations. The rest of hyperparameters ($\alpha.\theta_{\mathbf{K}}$) are trained via gradients update of the lower bound. For ease of notation, averages over functions of one variable correspond to averages over the corresponding factors e.g. $\mathbb{E}[h(x)] = \mathbb{E}_{Q^*(x)}[h(x)]$ where $Q^*(x)$ is the optimal distribution over x and other variables have been marginalized if required.

2.4.1 Optimal Poisson Variables

We now use the optimal equation for the variables ρ and ω, factorizing over the variables per arrival, $Q(\rho, \omega) = \prod_{i=1}^{N_T} q(\rho_i, \omega_i)$

$$q(\rho_i, \omega_i) \propto 2^{-\rho_i} e^{-\frac{\rho_i \mathbb{E}[f_i]}{2} - \frac{\mathbb{E}[f_i^2]}{2}\omega_i} PG(\omega_i|\rho_i, 0)\mathrm{Po}(\rho_i|\zeta_i) \tag{30}$$

We can obtain the exact form using the conjugate structure.

$$\begin{aligned} q(\rho_i, \omega_i) = \mathrm{Po}\left(\rho_i|\zeta_i \frac{e^{-\mathbb{E}[f_i]/2}}{2\cosh(\sqrt{\mathbb{E}[f_i^2]}/2)}\right) \\ \times PG(\omega_i|\rho_i, \sqrt{\mathbb{E}[f_i^2]}). \end{aligned} \tag{31}$$

From this we need to compute averages like $\mathbb{E}[\omega_i]$ and $\mathbb{E}[\rho_i]$ in order to obtain the optimal distribution over the functions $Q^*(\mathbf{f})$. We get

$$\mathbb{E}[\omega_i] = \mathbb{E}[\rho_i] \times \frac{1}{2\sqrt{\mathbb{E}[f_i^2]}} \tanh\left(\frac{\sqrt{\mathbb{E}[f_i^2]}}{2}\right), \tag{32}$$

where we have used the close form expression of the averages over Pólya Gamma distributed random variables, and we have notice that the remaining structure is conjugate to a Poisson distribution. The average over the ρ variable is easily obtained as the corresponding intensity of the Poisson distribution

$$\mathbb{E}[\rho_i] = \zeta_i \frac{e^{-\mathbb{E}[f_i]/2}}{2\cosh(\sqrt{\mathbb{E}[f_i^2]}/2)}. \tag{33}$$

2.4.2 Optimal Branching Structure

We now write the optimal equation for $\mathbf{Z}$ and Ω. Another factorization is possible as $Q(\mathbf{Z}, \mathbf{\Omega}) = \prod_i q(\Omega_{i.} Z_{i.})$. We define the unnormalized form of each of the individual factors as

$$\tilde{q}(\Omega_{i.} Z_{i.}) = e^{Z_{i0}\mathbb{E}[\log \lambda_0]} \prod_{j<i} 2^{-Z_{ij}} \left[\mu(T_i - T_j)\right]^{Z_{ij}} e^{\frac{Z_{ij}\mathbb{E}[f_j]}{2}} e^{-\frac{Z_{ij}\mathbb{E}[f_j^2]}{2}\Omega_{ij}} PG(\Omega_{ij}|1, 0) P(Z_{i.}) \tag{34}$$

The prior distribution $P(\mathbf{Z}_{i.})$ is a constant as any other arrival prior to i can be the parent of i. To obtain the actual distribution, one needs to normalize to obtain $q(\Omega_{i.} Z_{i.}) = \mathcal{Z}_i^{-1} \tilde{q}(\Omega_{i.} Z_{i.})$, the normalization is given by:

$$\mathcal{Z}_i = \sum_{Z_{i.}} \int d\Omega_{ij} e^{Z_{i0}\mathbb{E}[\log \lambda_0]} \prod_{j<i} 2^{-Z_{ij}} \left\{\mu(T_i - T_j)^{Z_{ij}} e^{\frac{Z_{ij}\mathbb{E}[f_j]}{2}} e^{-\frac{\mathbb{E}[f_j^2]}{2} Z_{ij}\Omega_{ij}} P_{\mathrm{PG}}(\Omega_{ij}|1, 0)\right\} \tag{35}$$

This expression is easily computed noting that $Z_{ij} \in \{1, 0\}$ so just one term in the products remains. We then insert the exact averages of the Pólya Gamma distribution. The final result yields

$$\mathcal{Z}_i = e^{\mathbb{E}[\log \lambda_0]} \frac{1}{2} \sum_{j<i} \mu(T_i - T_j) \times e^{\frac{\mathbb{E}[f_j]}{2}} \cosh^{-1}\left(\frac{\sqrt{\mathbb{E}[f_j^2]}}{2}\right). \tag{36}$$

Now we calculate the averages over the branching structure

$$\mathbb{E}[Z_{ij}] = \mathcal{Z}_i^{-1}\frac{1}{2}\left[\mu(T_i - T_j)\right] e^{\frac{\mathbb{E}[f_j]}{2}} \cosh^{-1}\left(\frac{\sqrt{\mathbb{E}[f_j^2]}}{2}\right), \tag{37}$$

and the average over the product of the Pólya gamma variables and the branching structure

$$\mathbb{E}[Z_{ij}\Omega_{ij}] = \mathcal{Z}_i^{-1}\frac{1}{2}\left[\mu(T_i - T_j)\right] \times e^{\frac{\mathbb{E}[f_j]}{2}} \cosh^{-1}\left(\frac{\sqrt{\mathbb{E}[f_j^2]}}{2}\right) \frac{1}{2\sqrt{\mathbb{E}[f_j^2]}} \tanh\left(\frac{\sqrt{\mathbb{E}[f_j^2]}}{2}\right), \tag{38}$$

as well as the sum over the branch Z indicator corresponding to the homogeneous base function

$$\mathbb{E}[Z_0] = \mathbb{E}[\sum Z_{i0}] = e^{\mathbb{E}[\log \lambda_0]} \sum_i \frac{1}{\mathcal{Z}_i}. \tag{39}$$

2.4.3 Optimal Gaussian Processes

We assume the function $f(\cdot)$ is generated from a $\mathcal{GP}(0, K)$ with zero mean and covariance structure given by the kernel $\mathbf{K}(T_i, T_j)$ over arrival times. For ease of calculation we represent the optimal distribution as

$$Q^*(\mathbf{f}) \propto e^{U(\mathbf{f})} p(\mathbf{f}) \tag{40}$$

And we have defined the effective log likelihood

$$U(\mathbf{f}) = \sum_{i=1}^{N_T} \left[-\frac{\mathbb{E}[\rho_i]}{2} f_i - \frac{\mathbb{E}[w_i]}{2} f_i^2 + \sum_{j<i} \frac{\mathbb{E}[Z_{ij}]}{2} f_j - \frac{\mathbb{E}[Z_{ij}\Omega_{ij}]}{2} f_j^2 \right]. \tag{41}$$

We introduce some matrices to re write Eq. 41 such that the conjugate structure of the likelihood is clear

$$\mathbf{f}^T \hat{\Omega} \mathbf{f} = \sum_{i=1}^{N_T} \sum_{j<i} -\frac{\mathbb{E}[Z_{ij}\Omega_{ij}]}{2} f_j^2 = \sum_{jk} f_j \left\{ \sum_i -\mathbb{1}_{[j<i]} \frac{\mathbb{E}[Z_{ij}\Omega_{ij}]}{2} \delta_{jk} \right\} f_k, \tag{42}$$

where we defined the diagonal matrix

$$[\hat{\Omega}]_{jj} = \sum_i -\mathbb{1}_{[j<i]} \frac{\mathbb{E}[Z_{ij}\Omega_{ij}]}{2}. \tag{43}$$

We further define $\mathbf{Z}_{ij} = \mathbb{1}_{[j<i]} \frac{\mathbb{E}[Z_{ij}]}{2}$, $\mathbf{W}_{ii} = -\frac{\mathbb{E}[w_i]}{2}$, $\mathbf{R}_i = -\frac{\mathbb{E}[\rho_i]}{2}$, so we can write the potential $U(\mathbf{f})$ in its matrix form

$$U(\mathbf{f}) = \mathbf{R}^T\mathbf{f} + \mathbf{f}^T\mathbf{W}\mathbf{f} + \mathbb{1}^T\mathbf{Z}\mathbf{f} + \mathbf{f}^T\hat{\Omega}\mathbf{f} = \frac{1}{2}\mathbf{f}^T\Lambda\mathbf{f} + \mathbf{a}^T\mathbf{f}. \tag{44}$$

Here we defined $\Lambda = \mathbf{W} + \hat{\Omega}$ and $\mathbf{a} = \mathbf{R} + \mathbf{Z}^T\mathbb{1}$. To obtain the Gaussian process posterior, one must perform matrix inversion, which takes $\mathcal{O}(N^3)$ time. In order to obtain an efficient algorithm, we follow the standard procedure (Csató 2002; Titsias 2009) and approximate the Gaussian process posterior $Q(\mathbf{f})$ with a sparse approximation $Q^s(\mathbf{f})$.

$$Q^s(\mathbf{f}) \propto p(\mathbf{f}|\mathbf{f}_s) e^{\mathbb{E}[U(\mathbf{f})|\mathbf{f}_s]} p(\mathbf{f}_s), \tag{45}$$

This approximation depends on a finite set of function values $\mathbf{f}_s$ and a set of *inducing points* $\mathbf{t}_s$.

$$\begin{aligned}\mathbb{E}[U(\mathbf{f})|\mathbf{f}_s] &= \frac{1}{2}\mathbb{E}_0\{\mathbf{f}|\mathbf{f}_s\}^T\Lambda\mathbb{E}_0\{\mathbf{f}|\mathbf{f}_s\} + \mathbf{a}^T\mathbb{E}_0\{\mathbf{f}|\mathbf{f}_s\} \\ &= \frac{1}{2}\mathbf{f}_s^T\mathbf{K}_s^{-1}\mathbf{k}_{Ns}^T\Lambda\mathbf{k}_{Ns}\mathbf{K}_s^{-1}\mathbf{f}_s + \mathbf{a}^T\mathbf{k}_{Ns}\mathbf{K}_s^{-1}\mathbf{f}_s = \frac{1}{2}\mathbf{f}_s^T\Lambda_s\mathbf{f}_s + \mathbf{a}_s^T\mathbf{f}_s \end{aligned} \tag{46}$$

Where $\mathbf{K}_s = \mathbf{K}(\mathbf{t}_s, \mathbf{t}_s)$ and $\mathbf{k}_{N,s} = \mathbf{K}(\mathbf{T}, \mathbf{t}_s)$. This allows us to express the sparse approximation Eq. 45 as

$$Q^s(f) \propto p(\mathbf{f}|\mathbf{f}_s) e^{\mathbb{E}[U(\mathbf{f})|\mathbf{f}_s]} p(\mathbf{f}_s) = p(\mathbf{f}|\mathbf{f}_s) q(\mu_2^s, \Sigma_2^s) \tag{47}$$

Where $\Sigma_2^s = (K_s^{-1} - \Lambda_s)^{-1}$, $\mu_2^s = \Sigma_2^s \mathbf{K}_s^{-1} \mathbf{k}_{sN} \mathbf{a}$ and $\Lambda_s = \mathbf{K}_s^{-1} \mathbf{k}_{Ns}^T \Lambda \mathbf{k}_{Ns} \mathbf{K}_s^{-1}$. Finally, we calculate the averages required for the self consistent mean field equations

$$\mathbb{E}_{Q^*(f)}[f_i] = \mu_i = \mathbf{k}_{is} K_{ss}^{-1} \mu_2^s \tag{48}$$

And the square average as

$$\mathbb{E}_{Q^*(f)}[f_i^2] = \mu_i^2 + \sigma_i^2 = \mu_i^2 + K_{i,i} - \mathbf{k}_{is} K_{ss}^{-1} (\mathbf{I} - \Sigma_2^s K_{ss}^{-1}) \mathbf{k}_{is}^T \tag{49}$$

2.4.4 Optimal Base Intensity λ_0

We now evaluate the optimal distributions for the base intensity function λ_0. The functional form of this parameters permits to place a gamma distribution as a conjugate prior. We obtain for the optimal distribution

$$Q^*(\lambda_0) = \text{Gamma}(\lambda_0 | \alpha_\lambda, \beta_\lambda) = \frac{\beta_\lambda^{\alpha_\lambda}}{\Gamma(\alpha_\lambda)} \lambda_0^{\alpha_\lambda - 1} e^{-\beta_\lambda \lambda_0} \tag{50}$$

And $\alpha_\lambda = \mathbb{E}[\sum_i Z_{i0}] + \alpha_0$ and $\beta_\lambda = T + \beta_0$. For the averages we have $\mathbb{E}[\lambda_0] = \alpha_\lambda / \beta_\lambda$ and $\mathbb{E}[\log \lambda_0] = \psi(\alpha_\lambda) - \log(\beta_\lambda)$. Where α_0 and β_0 corresponds to the hyperparameters of the base intensity prior.

2.4.5 Evaluating the Bound

In order to check the convergence of the training algorithm and to train the hyperparameters, we obtain the close form expression of the lower bound. We expand the joint distribution as the product of a likelihood conditioning on the function values $\mathbf{f}$.

$$\mathcal{L}(q) = \mathbb{E}_Q \left[\log \left\{ \frac{P(\mathbf{T}, \rho, \omega, \mathbf{\Omega}, \mathbf{Z} | \mathbf{f}, \lambda_0) P(\mathbf{f}) P(\lambda_0))}{Q(\mathbf{Z}, \mathbf{\Omega}) Q(\omega, \rho) Q(\lambda_0) Q(\mathbf{f})} \right\} \right] \tag{51}$$

Once we perform the averages over $Q(\mathbf{f})$

$$\mathcal{L}(q) = \sum_{i=1}^{N_T} [\log \mathcal{Z}_i + \mathbb{E}[\rho_i] - \zeta_i] - \mathbb{E}[\lambda_0] T - \text{KL}(q_2^s(\mathbf{f}_s) || p(\mathbf{f}_s)) - \text{KL}(Q(\theta) || P(\theta)) \tag{52}$$

Expanding the Kullback Leibler divergences

$$\begin{aligned} \mathcal{L}(q) = -\mathbb{E}[\lambda_0] T + \sum_{i=1}^{N_T} [\log \mathcal{Z}_i + \mathbb{E}[\rho_i] - \zeta_i] - \frac{1}{2} \text{Tr} \left\{ K_s^{-1} (\Sigma_2^s + \mu_2^s (\mu_2^s)^T) \right\} \\ - \frac{1}{2} \log \det(2\pi K_s) + \frac{1}{2} \log \det(2\pi e \Sigma_2^s) + \Psi(\alpha_0, \beta_0, \alpha_\lambda, \beta_\lambda) \end{aligned} \tag{53}$$

Where $\psi(\cdot)$ is the polyagamma function. And we have defined

$$\begin{aligned} \Psi(\alpha_0, \beta_0, \alpha_\lambda, \beta_\lambda) = \alpha_0 \log \beta_0 - \log(\Gamma(\alpha_0)) + (\alpha_0 - 1) \mathbb{E}[\log \lambda_0] - \beta_0 \mathbb{E}[\lambda_0] + \alpha_\lambda \\ - \log \beta_\lambda + \log \Gamma(\alpha_\lambda) + (1 - \alpha_\lambda) \psi(\alpha_\lambda) \end{aligned} \tag{54}$$

To avoid numerical instabilities, the expressions of the form $\log \Gamma(\alpha)$ can be calculated using Stirling's approximation $\log \Gamma(\alpha) \sim \alpha \log \alpha - \alpha$ for big α.

2.5 Hyperparameter Estimation

The model hyperparameters are the kernel $\mathbf{K}$ hyperparameters, the location of the inducing points $\mathbf{t}_s$, the parameters of the base intensity prior α_0, β_0 and the memory kernel μ decay rate α. The kernel hyperparameters as well as α are trained by gradient ascent, we follow the gradient of the bound Equation 53 (See Appendix), performing one iteration after the evaluation of the self consistent loop Equations. The inducing points are selected on a grid in the domain $\mathcal{T}_T$. Finally,

we set the base intensity prior parameters using the intensity obtained assuming a homogeneous Poisson process generated the data. We need to differentiate the bound Equation 53 with respect to the kernel hyperparameters θ

$$\frac{\partial \mathcal{L}(q)}{\partial \theta} = \mathrm{Tr}\left[K_{ss}^{-1}\frac{\partial K_{ss}}{\partial \theta}K_{ss}^{-1}(\Sigma_2^s + \mu_2^s(\mu_2^s)^T)\right] - \mathrm{Tr}\left[K_s\frac{\partial K_{ss}}{\partial \theta}\right] \tag{55}$$

And with respect to the memory rate α.

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \frac{1}{2}\sum_i \mathcal{Z}_i^{-1}\sum_{j<i}\frac{\partial \mu}{\partial \alpha}(T_i - T_j)e^{\frac{\mathbb{E}[f_j]}{2}}\cosh^{-1}\left(\frac{\sqrt{\mathbb{E}[f_j^2]}}{2}\right) + \sum_i \frac{\partial \zeta_i}{\partial \alpha}\left(\frac{e^{-\mathbb{E}[f_i]/2}}{2\cosh(\sqrt{\mathbb{E}[f_i^2]}/2)} - 1.\right) \tag{56}$$

2.6 MCMC

For comparison, we derive a Gibbs sampler which on the limit of large samples solves the problem exactly (Geman and Geman 1984). This Gibbs sampler generates samples from the posterior creating a Markov chain. At each step, the algorithm samples a block of variables from the conditional posterior, given all other variables. In consequence to perform the Gibbs sampler we derive these conditional distribution from Eq. 21. If a particular conditional is not analytically tractable, we perform Metropolis Hasting sample for this particular step (Bishop 2006). We provide the steps for each variable:

The omega variables conditionals

$$p(\omega_i|\mathbf{T}, \rho, \omega_{j\neq i}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{f}, \theta) = P_{\mathrm{PG}}(\omega_i|\rho_i, \mathbf{f}_i) \tag{57}$$

$$p(\Omega_{ij}|\mathbf{T}, \rho, \omega_{j\neq i}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{f}, \theta) = P_{\mathrm{PG}}(\Omega_{ij}|1, \sqrt{Z_{ij}}\mathbf{f}_j) \tag{58}$$

$$\tag{59}$$

We sample the ρ variables via Hasting-Monte Carlo:

$$2^{-(\rho_i' - \rho_i)}e^{-\frac{(\rho_i' - \rho_i)f_i}{2}}\frac{\mathrm{Po}(\rho_i'|\zeta_i)P_{\mathrm{PG}}(w_i|\rho_i', 0)}{\mathrm{Po}(\rho_i|\zeta_i)P_{\mathrm{PG}}(w_i|\rho_i, 0)} \tag{60}$$

For the Gaussian processes

$$P(\mathbf{f}|\omega, \mathbf{T}, \rho, \mathbf{\Omega}, \mathbf{Z}, \theta) = \prod_{i=1}^{N} e^{-\frac{-\rho_i f_i}{2} - \frac{f_i^2\omega_i}{2}} \times \prod_{j<i} e^{-\frac{Z_{ij}f_j}{2} - \frac{Z_{ij}f_j^2\Omega_{ij}}{2}} P(\mathbf{f}) \tag{61}$$

For the branching structure:

$$\mu_{ij} = \begin{cases} P(Z_{i0}) = \frac{\lambda_0}{W_i} & \text{if } j = 0 \\ P(Z_{ij}) = \frac{\sigma(f_j)\mu(T_i - T_j)}{W_i} & \text{otherwise} \end{cases} \tag{62}$$

Where $W_i = \sum \lambda_0 + \sum_{j<i}\sigma(f_j)\mu(T_i - T_j)$. For the parameters $\theta = \{\alpha, \lambda_0\}$, we use a Metropolis-Hasting step with the following acceptance rates:

$$A(\lambda_0') = \left[\prod_{i=1}^{N(T)}\left(\frac{\lambda_0'}{\lambda_0'}\right)^{Z_{i0}}(\frac{\lambda_0'}{\lambda_0})^{\alpha_{\lambda_0}-1}\right] \times \exp\{-(\lambda_0' - \lambda_0)(T + \beta_{\lambda_0})\} \tag{63}$$

$$\begin{aligned} A(\alpha) = &\left(\frac{\alpha'}{\alpha}\right)^{\alpha_\alpha - 1}\left(\frac{\alpha'}{\alpha}\right)^{\sum_{i=1}^{N_T}\sum_{j<i}Z_{ij}} \\ &\times \exp\left\{-(\alpha' - \alpha)\left[\sum_{i=1}^{N_T}\sum_{j<i}Z_{ij}(T_i - T_j)\right] - (\alpha' - \alpha)\beta_\alpha \right. \\ &\left. - \sum_{i=1}^{N_T}\sigma(f_i)[(1 - e^{-\alpha'(T-T_i)}) - (1 - e^{-\alpha(T-T_i)})]\right\} \end{aligned} \tag{64}$$

Where we placed Gamma priors with hyper parameters α_{λ_0} and β_{λ_0} for the base intensity function, and α_α and β_α for the decay rate of the exponential memory kernel.

2.7 Prediction

The results obtained up until are applicable to any memory kernel function, for prediction, we will now assume an exponentail excitations kernel.

2.7.1 Number of Arrivals

We will compute the number of arrivals as

$$\mathbb{E}[N(t_j, t)] = \int_{t_i}^{t} \hat{\lambda}_t dt \tag{65}$$

where

$$\hat{\lambda}(t) \equiv \mathbb{E}[\lambda(t)|\mathbf{f}, \mathcal{H}_{t_j}] \tag{66}$$

is obtained by averaging the intensity (19) over all random events after time t_j. Using the definition of the intensity and its specific form for an exponential memory kernel and sigmoid excitations, we obtain

$$\begin{aligned}\lambda(t) &= \lambda_0 + \int_0^t \sigma(f_s) e^{-\alpha(t-s)} dN(s) \\ &= \lambda_0 + e^{-\alpha(t-t_i)}(\lambda(t_j) - \lambda_0) + \int_{t_i}^{t} e^{-\alpha(t-s)} \sigma(f_s) dN(s)\end{aligned} \tag{67}$$

where $dN(s) = \sum_k \delta(s - t_k) ds$ denotes the process of arrival times ($\delta(s)$ being the Dirac–measure) and we have used the scaling property of the exponential kernel. Since the number of point events between times t and t' is $\int_t^{t'} dN(s)$, we get $\mathbb{E}[dN(s)|\mathbf{f}, \mathcal{H}_{t_j}] = \hat{\lambda}(s) ds$ for $s > t_j$. Hence, by averaging both sides of the previous equation, we obtain

$$\hat{\lambda}(t) = \lambda_0 + e^{-\alpha(t-t_i)}(\hat{\lambda}(t_j) - \lambda_0) + \int_{t_j}^{t} e^{-\alpha(t-s)} \sigma(f_s) \hat{\lambda}(s) ds \tag{68}$$

Differentiating this equation w.r.t. t we find the linear differential equation

$$\frac{d\hat{\lambda}}{dt} = -\alpha(\hat{\lambda}(t) - \lambda_0) + \sigma(f_t)\hat{\lambda}(t) \tag{69}$$

with the solution

$$\hat{\lambda}(t) = e^{-\alpha(t-t_j)} + \int_{t_j}^{t} \sigma(f_s) \left\{ \alpha\lambda_0 \int_{t_j}^{t} \exp\{\alpha(s - t_j) - \int_{t_j}^{s} \sigma(f_\tau) d\tau\} + \lambda(t_j) \right\} ds \tag{70}$$

Notice that the stationary solution is obtained from $\frac{d\hat{\lambda}}{dt} = 0$ yielding

$$\hat{\lambda}_\infty = \frac{\lambda_0}{1 - \sigma(f_t)} \tag{71}$$

i.e. the base intensities times the branching factor. On the stationary solution the number of arrivals is roughly the number of exogenous arrivals times the size of the branching cluster. Our model, will prefer lowering the base rate intensity as compared to the normal Hawkes process, as the number of arrivals can be regulated through the branching process with the self excitations.

2.7.2 Arrival Time

For prediction and the creation of samples, we require $P(T) \equiv P(T|\mathcal{H}_|)$ the probability density that the next point event arrives at time T given the previous history until time t_j. First, notice that the probability of no point arriving between t_j and $t_j + \tau$ can be obtained as an integral over P(T). Second, we equate this to having no arrival in an interval, the same quantity can also be obtained from the intensity $\lambda(t)$. Hence, we get the equality

$$\int_{\tau}^{\infty} P(T)dT = \exp\left\{-\int_{t_j}^{t_j+\tau} \lambda(t)dt\right\} = \exp\left\{-\lambda_0\tau - (1-e^{-\alpha\tau})(\lambda(t_j)-\lambda_0)\right\} \equiv G(\tau) \quad (72)$$

In the last line, we have used (19) together with a simple rescaling of the intensity with an exponential kernel when there are no new arrivals. This result can be used to compute the arrival time density as $P(T) = -\frac{dG(T)}{dT}$. The expected arrival time is obtained as

$$\mathbb{E}[T] = \int_0^{\infty} \exp\left\{-\lambda_0\tau - \frac{1}{\alpha}(1-e^{-\alpha\tau})(\lambda(t_j)-\lambda_0)\right\} d\tau, \quad (73)$$

2.8 Empirical Results

2.8.1 Synthetic Data

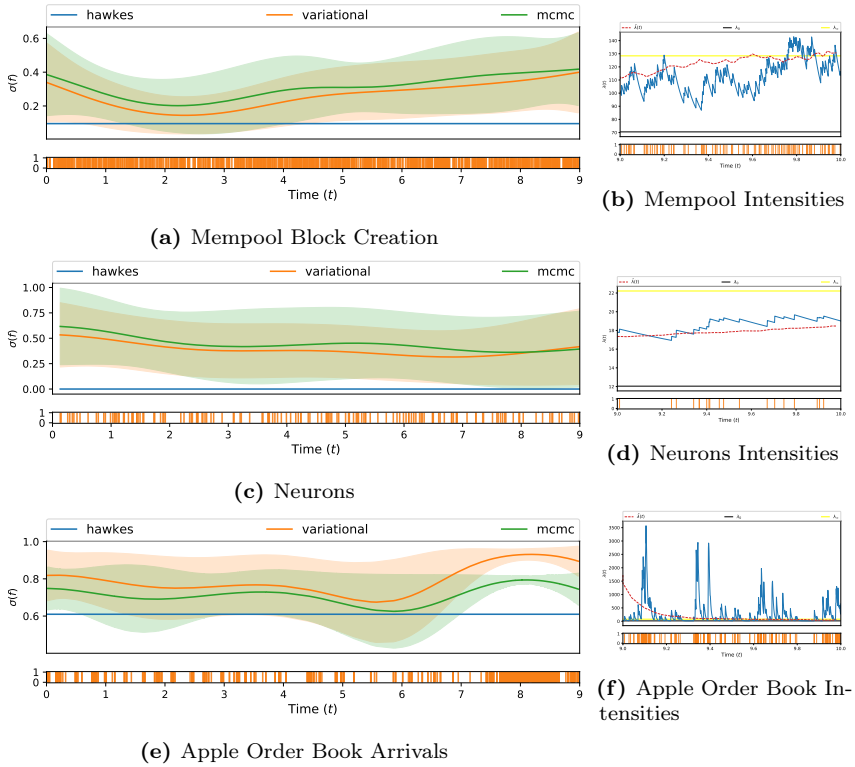
First, we test our algorithm on a synthetic data set. For the Gaussian process prior, we use the Matern kernel (Rasmussen and Williams 2006). We simulate the data with the same approach as (Lee et al. 2016), a sample from the Gaussian process prior instead of the excitations functions Y_i. For the memory kernel μ, we used the exponential $\mu(\cdot) = \alpha e^{-\alpha(\cdot)}$. We used a 10 inducing points $\mathbf{t}_s$ grid, since there was no change in the obtained bound by introducing more points. Results are shown for a realization of the process with 2384 arrivals in Fig. 3c.

2.8.2 Real World Data

Meempool The decentralized currency protocol known as Bitcoin (Nakamoto 2008) utilizes a peer-to-peer (P2P) architecture that enables users to send and receive transactions denominated in units of Bitcoin (BTC). To participate in the Bitcoin network the user runs a client software, such as the Satoshi client, which communicates with a set of peers. Transactions are broadcast by the Bitcoin client and received by the peer-to-peer network. They are confirmed after having been added to the "blockchain" - similar to a linked list with the subtle difference that it references the previous block using its hash rather than a pointer. This data structure contains blocks of all accepted transactions since the genesis of the system. The creation of each block defines a point process which we use for training our model.

Chalearn connectomics (Stetter et al. 2012): The data consist of a realistic simulation of neurons which generates calcium fluorescence recordings. The data also provides both the connectivity structure and spatial locations of the neurons. Excitatory behavior arises as a consequence of the network connectivity. For our algorithm, we used a squared exponential kernel for the $\mathcal{GP}$ prior. We show the results obtained in Figure 4. Different from previous work where the connectivity matrix is inferred from the dynamical behavior of the whole system (Linderman and Adams 2015), our approach is able to characterize the overall network excitations on individuals neurons.

Order Book Data: We include raw limited order book data, all events are recorded to seconds after midnight, with decimal precision of at least milliseconds. All data is based on NASDAQ historical total view ITCH-sample. We evaluate on 5 different companies, Apple (AAPL), Amazon (AMZ) Google (GOOG), Microsoft (MCFT) and Intel (INTC) for the first minutes after midnight



(a) Mempool Block Creation

(b) Mempool Intensities

(c) Neurons

(d) Neurons Intensities

(e) Apple Order Book Arrivals

(f) Apple Order Book Intensities

Figure 4 – Detail the Sigmoid Gaussian Hawkes Excitations model to empirical data set with variational (orange line) and mcmc inference (green line). We include the process intensities calculated sampling from the approximate posterior, as well as the numerical solution for its average (red line), and the stationary solution (yellow line)

2012-06-21. We used a total of 2500 arrivals per experiment.

Table 2.1 – Root Mean Square Error / Average Interarrival Time

results	amazon	mempool	neuron	apple	microsoft	intel	google
hawkes	0.668182	1.275561	1.305636	2.631194	0.791727	0.986070	0.969109
variational	0.668200	1.275561	1.305432	2.619131	0.791727	0.986070	0.968703
rnn(small)	7.881372	5.230715	1.727635	6.531822	3.859713	3.739753	8.085304
rnn (full)	2.602022	2.356998	0.749884	2.423415	4.175929	3.565702	2.275634
stochastic	0.883721	1.403117	1.566518	1.833392	1.396938	1.461080	1.195476

2.8.3 Results

We compare our variational procedure against our Monte Carlo Sampler (denoted MCMC), a simple Hawkes process trained with maximum likelihood (denoted HP), a Hawkes process model with Stochastic Excitations (Lee et al. 2016) (denoted Stochastic), trained from a Monte Carlo Sample and a Recurrent Marked Point Process model (Du et al. 2016a) (denoted RMTPP) defined with a recurrent neural network, trained via likelihood with back propagation through time. The specifics of this model will be left to section 4.2.2. Monte Carlo Samplers where sampled for at least 5000 cycles or until convergences, we show in Table 2.2 the training time of our algorithm as compared with MCMC inference. Due to the analytical updates for the self consistent equations, as well as the sparse Gaussian inference, our model outperform other Bayesian methods. We defined train and test data set, where the test dataset corresponds to all arrivals located in 10% of the support. For this region the avergae number of arrivals is also estimated (results on the appendix). All Bayesian models priors hyperparameters for α and λ_0 where defined from the simple Hawkes process solution. We present the results on Table 2.1, where we show the Root mean square error for the prediction

of the next event time given the prior history for all arrivals in the test dataset. As observed, our model outperforms most of the models in our datasets.

Table 2.2 – Inference Time (s)

Inference/# Arrivals	50	100	200	1000	2000
MCMC	6.010625	17.916019	77.118878	1809.011273	7900.422984
Variational	1.638125	1.075321	3.341016	91.939423	182.462920
HP	0.007731	0.002662	0.020379	0.048665	0.107135
Stochastic	4.669690	10.286880	23.965680	274.949215	1024.231485

2.9 Discussion and Outlook

In the present chapter, we extended the Hawkes process formalism by introducing correlations in the self-excitations as a function that we obtain from a Gaussian Process prior. This allows us to obtain great flexibility as different models can be leveraged through different kernels. We efficiently solve the inference problem presenting an augmented data likelihood through the use of Pólya-gamma random variables and a sparse Gaussian process approximation. The proposed form of the posterior yields a fast algorithm with for the close form solutions analytic updates. Our methodology consistently outperforms current methods in out of sample next point prediction in a variety of datasets.

One of the main limitations of the proposed methodology lies in the necessity for exponentials as memory kernels for the prediction procedures. The close form relies on the markovian structure obtained as a result of this expression. For other models, sampling might be suiting. The limitations of our proposed model offer possibilities for future work. It is known that neural dynamics exhibit inhibition phenomena; such characteristics can be incorporated by taking the sigmoidal function as a function of the sum of the memory terms. This approach will, however, remove the branching structure, possibly leading to scalability issues.

In the next chapter, we will develop another inference algorithm based on variational methods. Now, instead of resourcing to nonparametric methods for model flexibility, we will exploit deep neural networks and perform what is knowns as neural variational inference through reparametrization tricks.

Chapter 3

Switching Dynamical System

Many complex dynamical signals naturally feature an inherent compositional form, in the sense that their data generating process can be decomposed into different dynamical modes. For example, an NBA basketball player rapidly changes between moves as the game evolves, thereby composing complex two-dimensional trajectories; a handwriting signal is made up of successive strokes which differ in length, velocity, curvature and hence in their dynamics; multi-stable dynamical systems exhibit trajectories that can naturally be labeled according to their behavior around different attractors. Such inherent compositionality suggests that these systems' holistic dynamics decompose into a switching process determining the sequence of modes, the structure of the modes themselves, and the interplay between those two.

Switching dynamical systems (SDS) provides a natural way to perform unsupervised learning in time series. Akin to clustering methods which index data of similar structure in feature space, the switching mechanism in SDS indexes time series according to similar evolution patterns. NBA player trajectory tracking data and chaotic dynamical systems are two examples of phenomena that exhibit different dynamical modes. Intuitively, one should be able to uncover both offensive and defensive dynamic patterns of a player from his entire set of trajectories or uncover the different attractors characteristic of a chaotic system. Beyond indexing different dynamical regimes, however, the switching dynamics itself is of significant interest if one wants to provide a complete dynamical picture of complex time series. In recent years recurrent switching dynamical systems have provided tools to incorporate dynamical information into the switching process (see Section 3.1). However, these methodologies are hindered by the inherent limitations of linear dynamical systems– linear Gaussian transition functions ignore the pervasive non-linear and non-Markovian behavior common to real dynamical phenomena.

In this chapter, we overcome these limitations by exploiting recurrent neural networks (RNNs) within a framework for switching dynamical systems. This approach allows for an explicit description of non-linear and non-Markovian transition functions for the dynamics of both modes and switching. Indeed, within our model, the modes are learned through independent RNNs, whereas, similar to (mixture of) expert systems, the selection of modes is handled via a categorical distribution. Furthermore, the class probabilities of the categorical variables change dynamically: on the one hand, the class probabilities are conditioned on the hidden states of the (also independent) switching history, which we modeled via yet another RNN. On the other hand, the class probabilities are also informed by both hidden states and the dynamic modes' predictions through an attention mechanism. Finally, we enforce a finite entropy among the modes to balance them throughout the entire signal. We learn the model by introducing an approximate posterior distribution to perform inference over the dynamic categorical variables. The parameter optimization is then performed through maximum likelihood estimation of the corresponding bound.

3.1 Related Work

The study of different time regimes in time series has a venerable history in the Bayesian network community (Xuan and Murphy 2007; Oh et al. 2008). The modeling can be understood as an extension of the hidden Markov chain formalism where each latent state has an associated dynamical mode. Two main methodologies are applied. On one hand, change point models infer change point locations which differentiate the dynamical regimes (Saatçi et al. 2010). Within each new regime, parameters are learned for new dynamical models. On the other hand, one can incorporate the knowledge of past regimes and then switch among the dynamical modes (Linderman et al. 2017). Furthermore, non-parametric approaches allow these models to sample from an unknown number of dynamical modes (Fox et al. 2009). Such methods have also been exploited in the context of stochastic processes (Stimberg et al. 2012) wherein dynamical modes correspond to different parameter values for drift and diffusion operators in stochastic differential equations. In contrast to these works, the use of RNNs allows our model to describe non-linear dynamics with non-Markovian characteristics while still being able to capture the stochastic nature of the change point dynamics.

Motivated by the Bayesian dynamical network formalism we allow for stochastic dynamics along the switching. Variational auto-encoders (VAE) (Kingma and Welling 2013b) were first used in (Chung et al. 2015; Serban et al. 2017) to introduce variability into the transition functions of RNNs. These models are trained by maximizing a variational lower bound defined with respect to a set of approximating distributions. Estimating these lower bounds requires the calculation of averages over the approximating distributions, and the high variance of the derivatives of the such bounds is resolved via the reparametrization trick. Different from these approaches, our work introduces the stochastic nature of the transition *through the dynamics of the categorical variables* which characterize the switch. We do not provide a latent code for the transition operator but instead, a categorical variable to index the dynamic modes. More recently, variational auto-encoders and switching linear dynamical systems were used for Bayesian filtering in (Becker-Ehmck et al. 2019). In contrast to this work, our switching mechanism takes place in data space directly and allows for non-Markovian transitions within each dynamic mode.

3.2 Background

In this work we combine ideas from Switching Linear Dynamical Systems models (SLDS) with those from sequence modeling with RNNs in order to incorporate richer dynamical representations. Inference is accomplished within the Bayesian formalism via variational approximate inference. We now briefly review the different models components, as well as the inference framework, thereby laying the foundations of our model.

3.2.1 Switching Linear Dynamical System models

Switching linear dynamical system models aim to capture complex (non-linear) time series behaviour via a collection of so-called dynamical modes, each of which is approximated by a linear model — i.e. the complex signal is broken into a collection of simpler (linear) mappings. These models inherit the methodology of hidden Markov models and linear dynamical systems. We refer to (Linderman et al. 2017; Ackerson and Fu 1970) for further background.

In short, one assumes that at each time step t there is a corresponding categorical latent state z_t taking one of K different values and following the Markovian transitions

$$z_{t+1} \,|\, z_t \sim \pi_{z_t}, \tag{74}$$

where $\pi_{z_t} \in [0,1]^K$ gives the usual Markov transition probabilities. The classical approach (Linderman et al. 2017; Ackerson and Fu 1970) also introduces the continuous latent states $\mathbf{h}_t \in \mathbb{R}^p$

— these follow affine dynamics, with the different modes being indexed by z_t,

$$\mathbf{h}_{t+1} = \mathbf{A}_{z_{t+1}}\mathbf{h}_t + \mathbf{b}_{z_{t+1}} + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(0, \mathbf{Q}_{z_{t+1}}), \tag{75}$$

where $\mathbf{A}_k, \mathbf{Q}_k$ are matrices of the form $\mathbb{R}^{p\times p}$ whereas $\mathbf{b}_k \in \mathbb{R}^p$ and $k \in (1, ..., K)$. At last, the observed data points $\mathbf{x}_t \in \mathbb{R}^d$ are obtained via

$$\mathbf{x}_t = \mathbf{C}_{z_t}\mathbf{h}_t + \mathbf{d}_{z_t} + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, \mathbf{S}_{z_{t+1}}), \tag{76}$$

with $\mathbf{C}_k \in \mathbb{R}^{d\times p}$, $\mathbf{S}_k \in \mathbb{R}^{d\times d}$ and the drift terms $\mathbf{d}_k \in \mathbb{R}^d$.

An important remark is that each categorical state z_{t+1} depends only on the previous one z_t. This seems to limit the influence of the continuous latent variable $\mathbf{h}_t$ on the discrete switch — for instance, suppose that a certain critical subset $U \subset \mathbb{R}^p$ is such that, if $\mathbf{h}_t \in U$, then a discrete switch of the system (i.e. a particular change of z_t) is to take place. One may imagine that in such a scenario the switching would be difficult to capture unless the z_t are informed about the $\mathbf{h}_t$. To overcome this disadvantage Linderman et al. proposed an augmented model (recurrent SLDS or **rSLDS**) (Linderman et al. 2017) which makes use of the following generation scheme for z_t

$$z_{t+1} \,|\, z_t, \mathbf{h}_t, \{\mathbf{R}_k, \mathbf{r}_k\} \sim \pi_{SB}(\nu_{t+1}), \quad \nu_{t+1} = \mathbf{R}_{z_t}\mathbf{h}_t + \mathbf{r}_{z_t}. \tag{77}$$

Here π_{SB} is a certain stick-breaking distribution, $\mathbf{R}_k \in \mathbb{R}^{K-1\times p}$ captures the recurrent dependencies between z_t and $\mathbf{h}_t$, and $\mathbf{r}_k \in \mathbb{R}^{K-1}$ models the Markovian transitions between consecutive states z_{t+1} and z_t.

3.2.2 Modeling Time Series with Recurrent Neural Networks

Given a sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, RNNs process each $\mathbf{x}_t$ through the update of a hidden state $\mathbf{h}_t$ at each time step $t \in (1, \ldots, T)$. The update is implemented via a deterministic non-linear transition function f_θ thus

$$\mathbf{h}_t = f_\theta\left(\mathbf{h}_{t-1}, \mathbf{x}_t\right), \tag{78}$$

where $\mathbf{h}_t \in \mathbb{R}^p$, $\mathbf{x}_t \in \mathbb{R}^d$ and θ is the parameter set of f. In the present work we made use of Long Short Term Memory Network, explained in detail in Sec. 9. Given the set of hidden states $\mathbf{h}_t$ one can model the observed sequence by approximating its joint probability distribution function as

$$p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{<t}), \quad p(\mathbf{x}_t|\mathbf{x}_{<t}) = g_\varphi(\mathbf{h}_{t-1}), \tag{79}$$

where g with parameter set φ maps $\mathbf{h}_t$ to a probability distribution over outputs, and where $\mathbf{x}_{<t}$ denotes the dependence on the history.

3.3 Neural Variational Switching Dynamical Systems (NVSDS)

In this section we introduce the Neural Variational Switching Dynamical System model. Our main goal is to include the notions of non-Markovianity and non-linearity in the dynamical modes. In general terms, we proceed by substituting the transitions function of rSLDS with RNNs as well as MLP-parametrizations of categorical distributions for the switch distributions. We tackle inference via a variational approach, defining an approximate posterior over the categorical index variables. However, the RNN and MLP parameters are obtained as maximum likelihood with a direct optimization of the bound. An overview of our model is given in Fig. 5.

Suppose we have a complex signal $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ composed of K dynamical modes. Following the traditional mixture model approach we implement the dynamic switching between modes in terms of discrete latent variables $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_T)$. Specifically, at each time step t we have a K-dimensional latent variable $\mathbf{z}_t$ having a $1-$of$-K$ (one-hot encoding) representation — i.e. z_t^k is equal to 1 if $\mathbf{x}_t$ was generated from the kth dynamical mode and it is 0 otherwise.

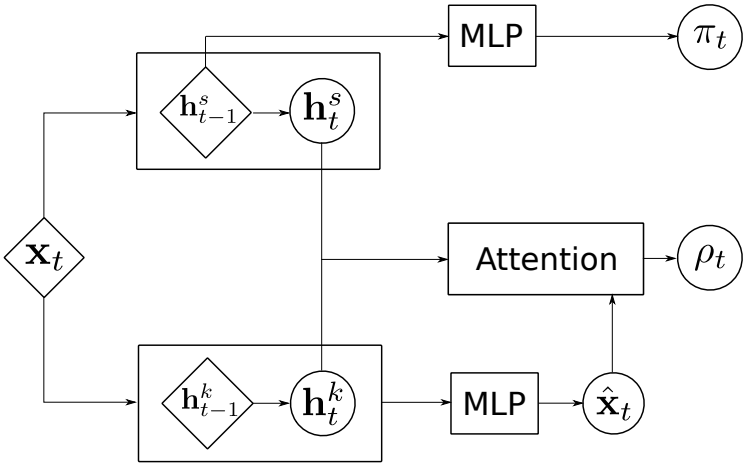


Figure 5 – Architecture of the NVSDS model. For a given sequence $\mathbf{x}_t$ the data is fed into the recurrent network modeling the modes dynamics (lower part) as well as the switching dynamics (upper part). The representations $\mathbf{h}$ obtained by the experts' dynamics are fed into an MLPs parametrizing a Gaussian distribution for the outputs $\hat{x}_t$. The experts' representations, their prediction as well as the hidden state of the switching are fed into the attention mechanism which finally parametrizes the categorical distribution.

Generative model — The prior distribution over $\mathbf{z}_t$ is specified in terms of a set of class probabilities $\pi^k_t = p(z^k_t = 1)$, which are required (by the switching dynamics) to have an intrinsic dynamic behaviour. Akin to Eq. (74) we introduce a RNN whose sole role is to govern the switching dynamics

$$\mathbf{h}^s_t = f_{\theta_s}\left(\mathbf{x}_t, \mathbf{h}^s_{t-1}\right), \tag{80}$$

with $\mathbf{h}^s_t \in \mathbb{R}^s$ encoding the independent switching dynamics and θ_s the parameter set of the transition function, which we implement using a long-short-term memory (LSTM) network (Hochreiter and Schmidhuber 1997). We then define the set of dynamic class probabilities as

$$\pi^k_t = \text{softmax}\left[g_{\theta_k}\left(\mathbf{h}^s_{t-1}\right)\right], \tag{81}$$

where the softmax function is computed over the K modes and g_{θ_k} denotes a multilayer perceptron (MLP) with parameter θ_k. We can now write the prior distribution over $\mathbf{z}_t$ as

$$p(\mathbf{z}_t) = \prod_{k=1}^{K} \left(\pi^k_t\right)^{z^k_t}. \tag{82}$$

On the other hand, we model each dynamic mode $\mathbf{x}^k_t$ as a normal distribution parametrized by MLPs. The latter are conditioned on the hidden state of a RNN which approximates the mode's transition function, as discussed in Section 3.2.2. Specifically, we write

$$p(\mathbf{x}^k_t \,|\, \mathbf{x}^k_{<t}) = \mathcal{N}(\boldsymbol{\mu}^k_t, \text{diag}\left[(\boldsymbol{\sigma}^k_t)^2\right]), \quad \left[\boldsymbol{\mu}^k_t, \boldsymbol{\sigma}^k_t\right] = g_{\varphi_k}\left(\mathbf{h}^k_{t-1}\right), \tag{83}$$

where both $\boldsymbol{\mu}^k_t, \boldsymbol{\sigma}^k_t \in \mathbb{R}^d$ and g_{φ_k} is again an MLP. Let us note here that Eq. (83) corresponds to our neural network generalization of Eq. (76). The mode's hidden state $\mathbf{h}^k_t \in \mathbb{R}^p$ follows from the state transition function

$$\mathbf{h}^k_t = f_{\varphi_k}\left(\mathbf{x}_t, \mathbf{h}^k_{t-1}\right), \tag{84}$$

with f implemented by a LSTM and φ_k the parameter set for the kth mode. Eq. (84) should be compared to Eq. (75). Different from rSLDS the RNN provides a deterministic transition function. The stochastic aspect of the dynamical modes is included through the output distributions.

The full generative model has the following joint probability distribution

$$p(\boldsymbol{z}_{\leq \boldsymbol{T}}, \boldsymbol{x}_{\leq \boldsymbol{T}}) = \prod_{k=1}^{K} \prod_{t=1}^{T} \left(\pi_t^k \, p\left(\mathbf{x}_t^k | \mathbf{x}_{<t}^k\right)\right)^{z_t^k} . \tag{85}$$

Inference — The posterior distribution over the categorical variables $\mathbf{z}_t$ of our model Eq. (85) is clearly intractable. We introduce the approximate posterior distribution

$$q(\mathbf{z}_t | \mathbf{x}_{\leq t}) = \prod_{k=1}^{M} \left(\rho_t^k\right)^{z_t^k} , \tag{86}$$

where the dynamic posterior class probabilities ρ_t^k are defined through an *attention mechanism*.

Attention mechanism — We take advantage of the freedom we have in defining ρ_t^k to enrich the switching process by incorporating contextual comparison of the modes' encoding. We do this through an attention mechanism — that is, we dynamically adapt the modes' selection given their current representation. Consider a per-mode hidden state representation

$$\mathbf{u}_k = \sigma\left(\mathbf{W}_k \, \mathbf{H}_t + \mathbf{V}_k \, \mathbf{h}_t^s + \mathbf{b}_k\right), \tag{87}$$

where $\mathbf{H}_t = \left([\hat{\mathbf{x}}_t^1, \mathbf{h}_t^1], [\hat{\mathbf{x}}_t^2, \mathbf{h}_t^2], \ldots, [\hat{\mathbf{x}}_t^K, \mathbf{h}_t^K]\right) \in \mathbb{R}^{K*(d+p)}$ contains both the prediction $\hat{\mathbf{x}}_t^k$ and hidden state $\mathbf{h}_t^k$ of each mode; $\mathbf{h}_t^s$ is the hidden state encoding the switching dynamics, Eq. (80); and $\mathbf{W}_k \in \mathbb{R}^{a \times K*(d+p)}$, $\mathbf{V}_k \in \mathbb{R}^{a \times s}$ and $\mathbf{b}_k \in \mathbb{R}^a$ are trainable variables. The function $\sigma(\cdot)$ denotes the hyperbolic tangent.

Now consider a per-mode context vector $\mathbf{c}_k \in \mathbb{R}^a$, which is also to be trained. We define the dynamic posterior class probabilities as

$$\rho_k^t = \text{softmax}\left[\mathbf{u}_k \cdot \mathbf{c}_k\right], \tag{88}$$

where the softmax function is taken over the modes. Different from common forms of attentions (Yang et al. 2016) (Bahdanau et al. 2014), within our approach each mode performs its own *translation* of the context, similar to what was proposed in (Schwab et al. 2018). As an illustrative example, one can think of a crowd of experts of different nationalities trying to understand a text in a foreign language. The common text (here $\mathbf{u}_k$) is the context known to all experts, each expert must in turn translate the context to its own native language prior (here $\mathbf{c}_k$) to decide whether the text in question corresponds to her area of expertise.

Learning — Inserting Eq. (85) and Eq. (86) into Eq. (5) we write the variational lower bound

$$\mathcal{L}[q] = \mathbb{E}_{p_D(\mathbf{x})} \left[\sum_{k=1}^{K} \sum_{t=1}^{T} \left\{\rho_t^k \, \log p(\mathbf{x}_t^k | \mathbf{x}_{<t}^k) + \rho_t^k \, \log \left[\frac{\pi_t^k}{\rho_t^k}\right]\right\}\right], \tag{89}$$

where we have performed the averages over the categorical distributions, using the fact that $\mathbb{E}_{q(\mathbf{z})} z_t^k = \rho_t^k$, and where $p_D(\mathbf{x})$ denotes the empirical data distribution.

Mode regularization — One common drawback of expert-like systems occurs during training. Due to inhomogenities in the initial conditions of the parameter space, one mode (expert) may happen to have a subtle advantage over the others in predicting the dynamics. Such initial conditions will compound over time as the categorical switch prioritizes this mode thus increasing its advantage over the others. In other words, the mode will be preferred not as a consequence of its knowledge of the dynamical regime, but as a consequence of training imbalances. One must therefore enforce dynamical diversity by imposing cost functions to be trained along side the maximization of the lower bound Eq. (89). This issue has been encountered and addressed in various (static) settings (Schwab et al. 2018; Shazeer et al. 2017; Bengio et al. 2015). We introduce the mode entropy

$$\mathcal{H}[\rho] = -\mathbb{E}_{p_D(\mathbf{x})} \sum_{k=1}^{K} \tilde{\rho}_k \, \log \tilde{\rho}_k, \tag{90}$$

where $\tilde{\rho}_k$ is the time-average posterior class probabilities. Below we demonstrate empirically our

intuition that maximizing this entropy helps avoiding the “elimination" of the expert modes during training. The regularized model then seeks to maximize $\mathcal{L}'[q] = \mathcal{L}[q] + \lambda_e \mathcal{H}[\rho]$, where λ_e is a hyperparameter.

An expectation-maximization solution to NVSDS (NVSDS-EM) — Instead of introducing the approximate posterior Eq. (86) one can directly optimize the bound by noticing $\mathcal{L}[q]$ is nothing but the negative Kullback-Leibler divergence between $q(\mathbf{z})$ and $p(\mathbf{x}, \mathbf{z})$. An optimal lower bound is then found by minimizing this divergence. Such a minimum happens only for $\log q(\mathbf{z}) = \log p(\mathbf{x}, \mathbf{z}) + \text{const}$. We can then simply write the approximate posterior as

$$q(\mathbf{z}) = \prod_{t=1}^{T} \prod_{k=1}^{K} \left(\frac{\rho_t^k}{\sum_k^K \rho_t^k} \right)^{z_t^k}, \quad \rho_t^k = \pi_t^k \, p(\mathbf{x}_t^k | \mathbf{x}_{<t}^k), \tag{91}$$

where π_t^k and $p(\mathbf{x}_t^k | \mathbf{x}_{<t}^k)$ are defined in Eq. (81) and Eq. (83), respectively. Note that in contrast to the NVSDS model, the computation of ρ_t^k does not exploit the $\mathbf{h}_t^k$ encoding of the modes through an attention mechanism.

3.4 Experiments

In this section we provide the experimental framework upon which we tested our model. We begin by briefly defining three baseline models against which we compared our results. We then specify both the architecture of the different neural networks composing the models, as well as the corresponding hyperparameters.

We evaluate our models in four datasets: (i) as a show-case or proof of concept we consider a dynamical system with a chaotic attractor — the Lorenz system (Lorenz); (ii) we execute a detailed model comparison on sinusoidal data with switching frequencies (Sine) and (iii) Handwriting data (HW); (iv) finally, we perform an exploratory analysis with the NVSDS model on a basketball dataset (Basket). Below we describe in detail each of these datasets and analyse our results.

3.4.1 Baseline Models

A simple approach to dissect a time-series into different regimes is to perform clustering on the hidden states of a RNN model trained on the time-series. Although the resulting dissection is static, it provides a useful baseline to compare the dynamic dissection of our model. We thus use k-means clustering together with an LSTM cell — We shall refer to this model as R-k-Means. We also consider the rSLDS model (Linderman et al. 2017), which we briefly introduced in Section 3.2.1. For background, framework and code utilities we refer to (Linderman). Finally we consider a standard mixture of experts model (see e.g. (Bishop 2006)) wherein each expert is modeled via an LSTM (Eq. (83)), and where the gating mechanism (π_k^t in our notation) is defined as

$$\pi_k^t = \text{softmax}\left[g_{\theta_k}(\mathbf{h}_t^k, \mathbf{x}_t^k) \right]. \tag{92}$$

Here $\mathbf{h}_t^k \in \mathbb{R}^p$ is the expert's hidden state, $\hat{\mathbf{x}}_t^k \in \mathbb{R}^d$ is the expert's prediction and g_{θ_k} is given by an MLP with parameters θ_k. We refer to this model as MoE.

3.4.2 Training Details

We choose the latent dimension and the discrete hidden state dimension of the rSLDS model to be two (four) for the Sine (Handwritting) dataset. For the R-k-Means model we train an LSTM with hidden size 16 (64) for the Sine (Handwriting) data. We set the number of clusters equal to the number of modes of the NVSDS model in the respective dataset. The hyperparameters of our

Table 3.1 – Hyperparameter specification for all experiments: Number of modes (K), hidden size per mode (p), hidden size of switching LSTM (s), attention dimension (a), number of layers in the switching MLP (N_l^s), and learning rate (λ).

Dataset	Model	M	p	s, a	N_l^s	λ
Lorenz	NVSDS	2	16	16	2	0.005
Sine	MoE NVSDS-EM NVSDS	2	16	32	1	0.005
HW	MoE NVSDS-EM NVSDS	4	64	32	2	0.005
Basket-all players	NVSDS-EM	16	32	32	2	0.005
Basket-single player	NVSDS-EM	8	64	32	2	0.005

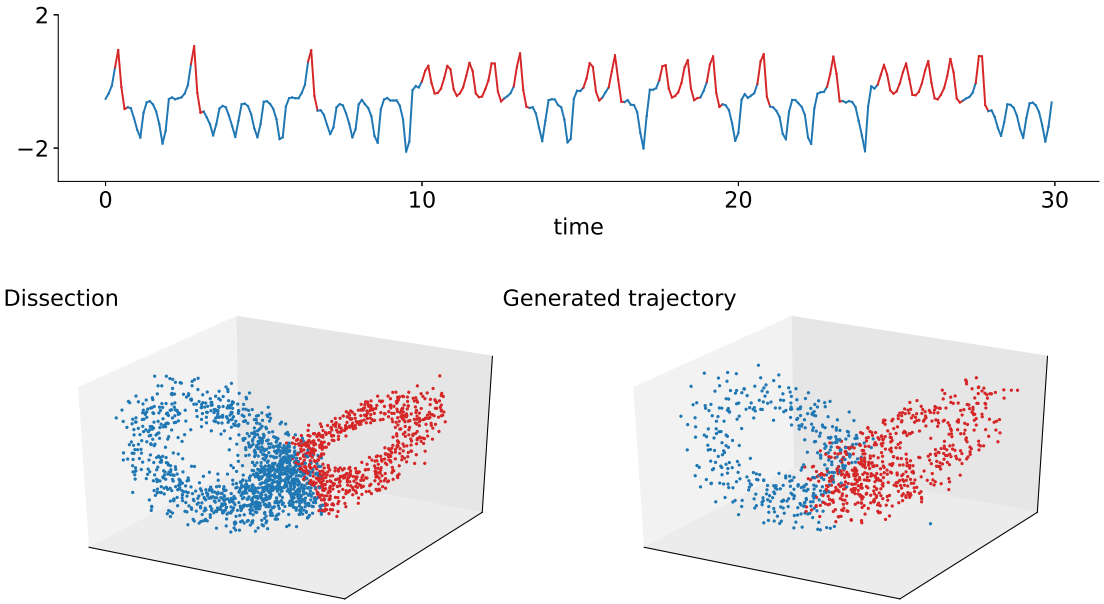


Figure 6 – Switching behavior in the Lorenz attractor. The upper panel corresponds to one dimension of the system over time. The solid line shows the one-step ahead prediction of the NVSDS model, the color corresponds to the dominating mode. Likewise the lower left panel shows one-step ahead prediction for the two-dimensional time series. The lower right panel corresponds to a generated trajectory.

models are given in Table 3.1. We also set λ_e, the hyperparameter controling the entropy regularizer in NVSDS and NSVDD-EM models, to one in all our experiments. Regarding the optimization of the neural networks parameters we use ADAM (Kingma and Ba 2014).

3.4.3 Lorentz Attracttor

Being an epitome of a chaotic dynamical systems, the Lorentz system was introduced as a model of atmospheric convection (Lorenz 1963). The system is characterized by two chaotic attractors — that is, separated regions of phase space with different dynamic behaviors. We simulated the Lorenz system with different initial conditions and generated 100 trajectories which correspond to our training set. We set K, the number of modes (experts) to two.

Intuitively, one would expect each attractor to be identified as one of the modes (experts) of the model. In Fig. 6 we show the behavior of the NVSDS model on a test trajectory. As expected, each

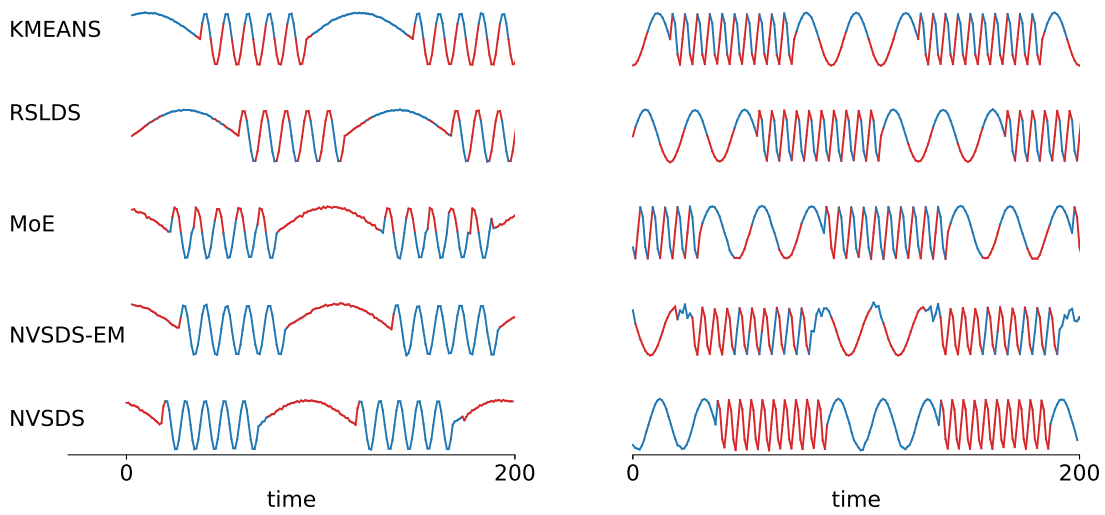


Figure 7 – Switching behavior in an oscillatory signal. The solid line shows the one-step ahead prediction of the different models. The color corresponds to the dominating expert (or chosen cluster in k-Means). The left and the right column show two signals with different frequencies. Only the NVSDS model successfully dissects the signal.

mode picks one attractor. We also generated trajectories in an open loop manner and observed an explicit separation between the two different attractors, forming the well known “butterfly" shape. These results, however, are meant to be a proof of concept. We do not show a comparison to the other models since all of them are able to correctly dissect the signal. Nevertheless our models (NVSDS, NVSDS-EM) are the first neural dynamical switching models successfully applied to this dataset.

3.4.4 Switching Oscillatory Dynamics

We now consider a synthetic dataset with two dynamical modes ($K = 2$): two sine functions with different frequencies, concatenated at regular time intervals. The switching behavior of the signal is therefore stationary. The goal is to dissect the signal into these modes, that is, uncover the two different frequencies. While this task is trivial for a human, and can easily be solved using signal processing methods, we have found it to be highly non-trivial for a switching dynamical system — which only sees one data point at a time. The guiding intuition is that, in a 2-expert model, each expert learns a different sine function. Fig. 7 shows our results. The R-k-Means, rSLDS and MoE models are not able to dissect the sine functions; they find instead a more local dissection into “up-down" states. The NVSDS and the NVSDS-EM models perfectly dissect the signal, showing the abstraction capabilities of these models. We asses these observations quantitatively by defining a binary target vector $\hat{\rho}$ indicating the dynamical mode which is present at a particular time-step. Evaluating the mean squared error between the predicted ρ and $\hat{\rho}$ yields a dissection error, which we average over multiple trials (different initial conditions). The NVSDS (0.09) and the NVSDS-EM (0.1) clearly outperform R-k-Means (0.26), rSLDS (0.4) and MoE (0.47).

The right column in Fig. 7 shows a second signal where both modes have full periods. The NVSDS model exclusively finds the correct dissection and learns to switch between frequencies. The comparison between the models suggests that both the non-linear transition function and the attention mechanism within the switching of the NVSDS play a key role in our results.

The main difficulty in this task is the existence of different signal dissections (minima) with very similar energies (in terms of the bound). The “up-down" solution found by the baseline models is but one of these. Indeed we find quite complex learning curves, with the models jumping back and

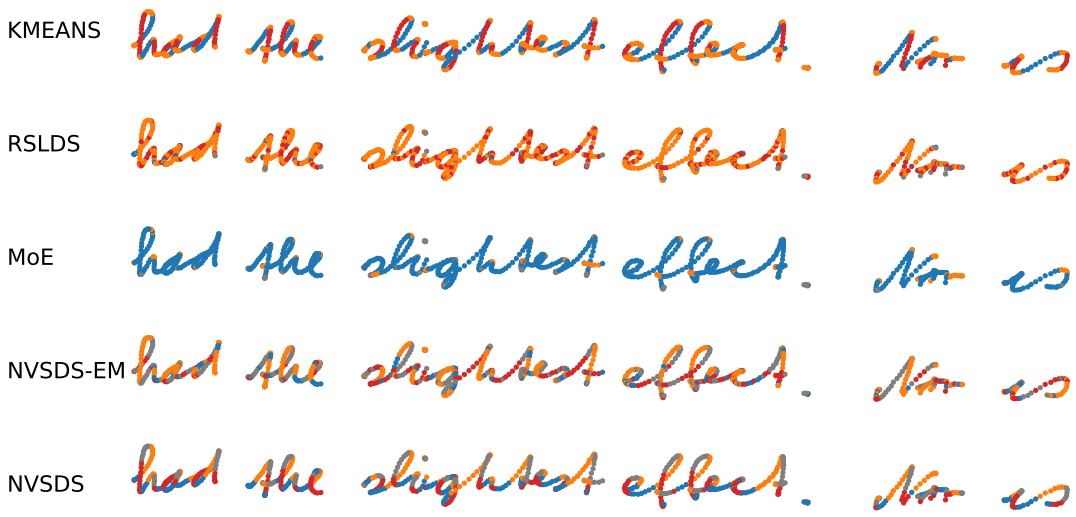


Figure 8 – Dissection of a handwriting signal. We show the original data colored according to the dominating expert.

fourth between different minima as the training progresses.

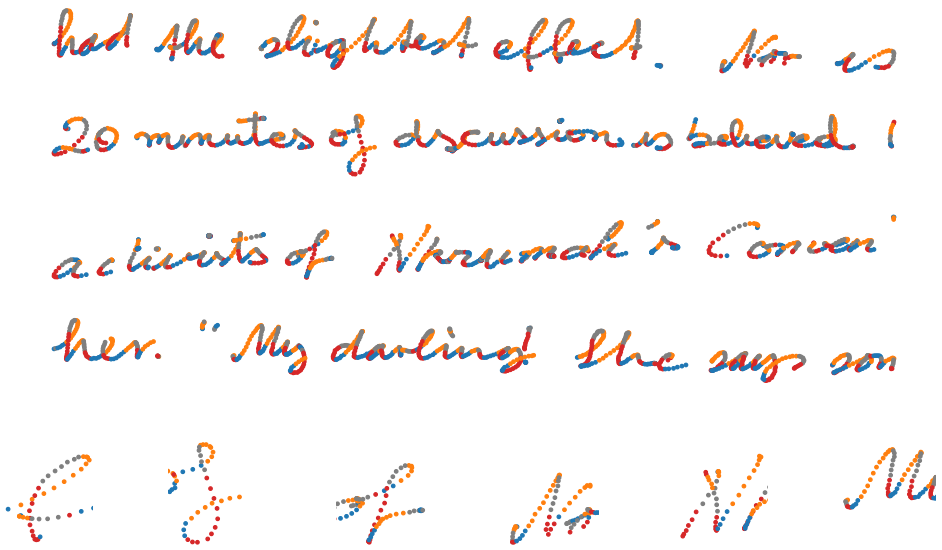


Figure 9 – Dissection of a handwriting signal for the NVSDS model for different sequences. The lower row shows particular letters from the complete sequences for easier comparison.

3.4.5 Handwriting

To show the performance of our methodology in datasets exhibiting the full range of complex behavior — non-linear, non-Markovian stochastic transitions — we concentrate on a handwriting signal from the IAM-OnDB dataset. This consists of 13,040 handwritten lines written by 500 writers (Liwicki and Bunke 2005). We train on 900 sequences of 1100 data points and test the behavior on held-out test sequences. We train the model by feeding in the velocity of the signal. After training we select a test sequence and perform one-step ahead prediction. We color the signal according to the predicted expert activity. The intuition is that each mode (expert) should pick a specific stroke style like fast or curved strokes.

Fig. 8 shows a comparison of the results we obtain for the different models. With R-k-Means we basically find upwards, downwards and side-way movements, which serves as a good starting point. The rSLDS model is unable to dissect the signal into different strokes. Presumably this drawback is due to the limiting assumptions of linearity and Markovinity inherent in the rSLDS model. The MoE model fails to dissect the dynamics, a single expert dominates the signal. Now, while the NVSDS-EM and NVSDS share the same modular architecture as the MoE model, they also profit from both the attention mechanism and the internal history of the switch dynamics. This advantage leads to an interpretable dissection of the signal: for example, a downward movement for the NVSDS model is consistently captured by the same sequence of experts (grey-red colored) and a upwards movement to the right is captured by the expert indicated by the orange color. A detailed comparison between the NVSDS-EM and the NVSDS model reveals that the NVSDS-EM model features a more stochastic switching behavior and therefore can be seen slightly inferior to the NVSDS model.

We further investigate the behavior of the NVSDS model in different writing styles in Fig. 9. Considering the same letter written by different writers (the three f's in the lower row) we observe that our model consistently dissect the letter into the same sequence of experts. Furthermore, we compare the disection of the capital letter 'N' (lower row). Here, the sequence of experts is different in the first stroke of the letter, which presumably reflects that these strokes where written in the opposite direction. In conclusion, our model is able to capture different strokes and thus identifying the building blocks of the individual letters.

Let us note that in this dataset we do not show the actual signal prediction of the models. The reason behind this is that we did not optimize our architecture to perfectly learn this non-continuous signal. Our actual predictions would result in non-readable writings. We are nonetheless only interested in a dissection of the given signal into meaningful dynamical modes.

3.4.6 Basketball Dataset

Finally, we perform an exploratory analysis by applying the NVSDS model to basketball data. Concretely, we consider player trajectories from games of the National Basket association (NBA). First, we study the 2015/2016 season for the Detroit Pistons team. This dataset consists of the court positions (x, y) at time t, for all players in 22 games. We selected a total of 199 trajectories. Among those, 100 trajectories are selected as the train set. We test our model on the 99 remaining trajectories. We train our model on the velocity of the players, which we compute from the data.

We present averaged expert specific vector fields (see caption for details) in Fig. 10. We train a model with $K = 16$ and present the most interpretable 8 patterns on the test set. We obtain a dissection of player movements based on speed and direction. For example, we uncover fast movements which cut to the upper right corner (expert 2) or horizontal movements below the basket (expert 1). We also observe movements away from the basket with different directions (expert 3 and expert 4), possibly showing defense behavior. Further, some experts pick up slow movements (expert 5 and 6) or even a standing still state (expert 7). In general, the experts obtain similar movement patterns as found in (Linderman et al. 2017). In contrast to the experiments in (Linderman et al. 2017), however, we train on a larger datasets showing the scaling abilities of the neural model developed in this work.

We now aim to investigate the movements of individual players, whereby we focus on two players from the California Golden State Warriors, Andrew Bogut and Stephen Curry. The former one is playing as a center whereas the latter plays as a point guard. Traditionally, center players tend to position themselves near the basket, whereas point guards tend to cover the whole court, as to organize the general strategies. For each of the two players we create a different dataset picking 16 games for Bogut and 17 games for Curry and train on these dataset separately. We show the two clearest patterns in Fig. 11 and uncover vertical (1b) and horizontal movements (2b) near the

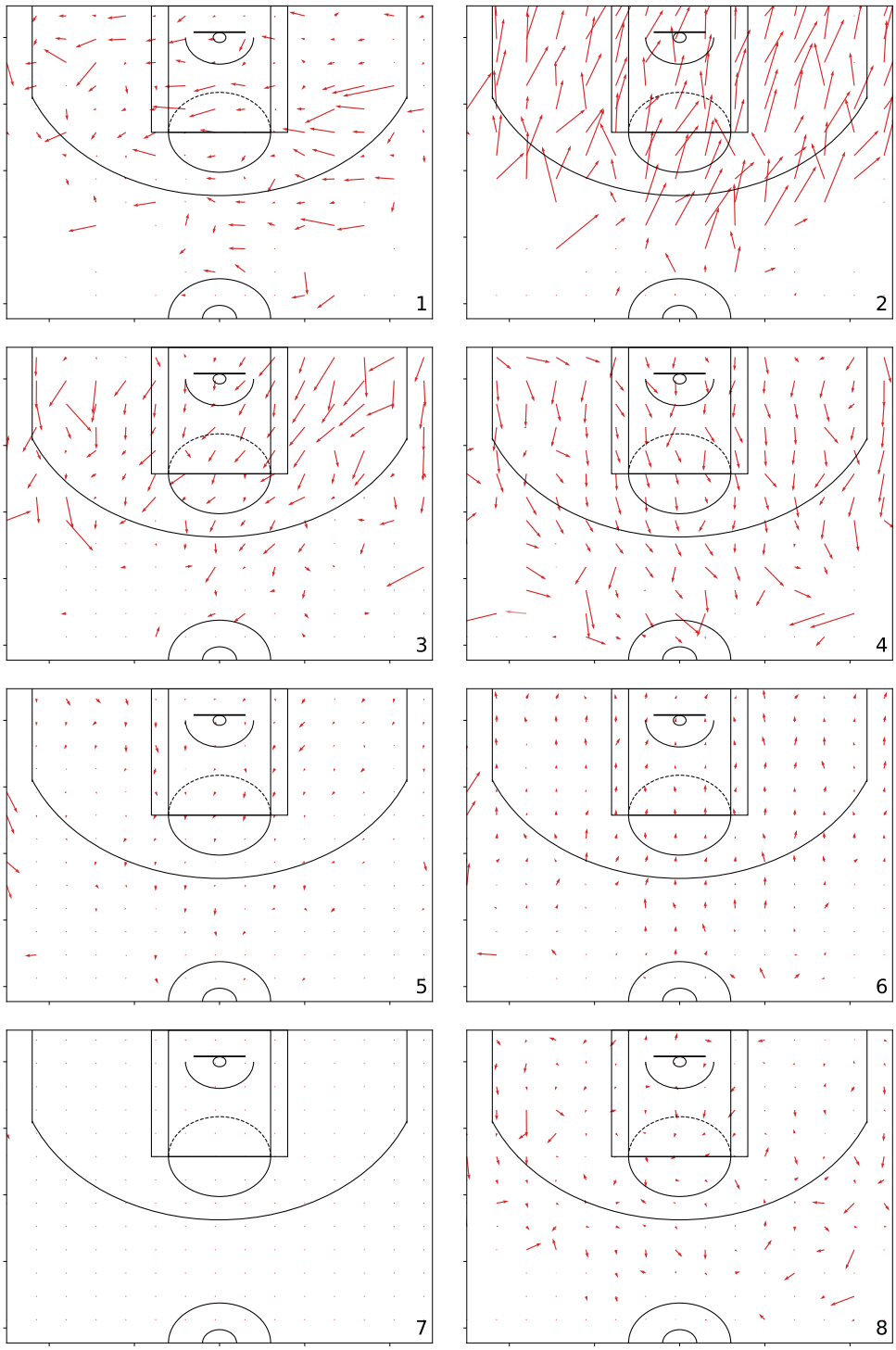


Figure 10 – Different movement patterns in the basketball dataset. We parcellate the court in 0.75m×0.75m quadrants. For each quadrant we pick the trajectories which 'belong' to a particular expert, i.e. for which the predicted ρ is largest. Then, we average the velocities and obtain for each expert an averaged flow field shown in the different panels.

basket for the center player Bogut. In contrast, the patterns for Stephen Curry extend over the whole court. This player is known for preferring 3-point shots slightly more than other players which is indicated by the pattern in expert 1c: Some velocity vectors seem to be tangential to the 3-point line, indicating that the player moves along the line to position himself for a shot. Furthermore, expert 2c seems to indicate counter-attack movements. Although qualitative in nature, these results highlight the ability of our model to provide insights into the dynamical modes of rather complex datasets.

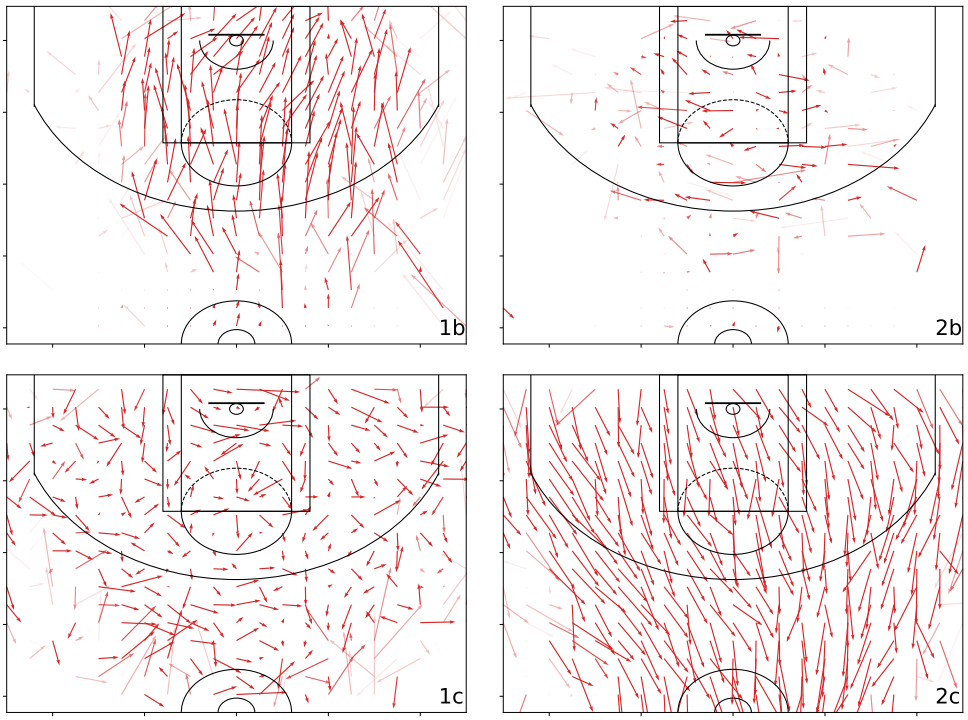


Figure 11 – Different movement patterns in the basketball dataset for individual Players. Visualisation is the same as Fig. 10 with 0.6m × 0.6m quadrants. Upper row corresponds to Andrew Bogut, lower row to Stephen Curry.

3.5 Discussion and Outlook

In the present chapter, we have provided a neural network solution for switching dynamical systems. Our methodology builds upon variational approximate inference for the categorical variables indexing of the dynamical modes. An attention mechanism and an entropy regularizer are introduced to improve the detection of the modes. We applied our methodology successfully in diverse empirical datasets.

Beyond the results shown in this chapter we have also made a couple of observations that we believe to be important for switching dynamical systems. We have noticed that dynamical modes with long-time scales are challenging to capture, e.g. to capture a complete letter in the handwriting dataset. There are several reasons for this behavior. One is technical: it was argued recently that the memory time-scale of the RNN is limited (Bai et al. 2018). We suspect that this small timescale hinders our architecture from uncovering switches on a larger time-scale. Another issue we have observed is the complicated energy-landscape: the time-scale of the switching is not explicitly incorporated in our loss function, therefore slow and fast switching solutions can have similar energy. Moreover, the modes' statistics is crucial: e.g. in the handwriting dataset, one could consider a particular user letter style as an individual mode. If only a few examples per user are provided, then this particular mode appears only rarely. The same holds true for a dynamical mode with a very long time-scale compared to the sample signal size. In this work, we have focused on the dissection of complex signals. Our model learns the dynamics of the switches, which provides a definite competitive advantage against simple detection algorithms, say MoE, i.e. we can predict the sequence of modes not just detecting them.

In terms of future work, a major goal is to exploit the NVSDS model to perform both outlier prediction and detection. We also plan to dispense with the RNN methodology and use Diluted Temporal Convolutions to overcome the time-scale problem.

In the next chapter, we will move away from variational inference and work with a different procedure.

We will resource to adversarial training, where, instead of providing a posterior approximation, one obtains samples from the data distributions. If used effectively, these methods provide more general distributions, since they are not limited by the approximation required in variational inference. More importantly, other divergences between distributions are used. It is known that the extensions through optimal transports point of view allows us to incorporate data distributions that are supported in low dimensional manifolds. The adversarial approach, when used successfully, provide better data samples.

Part II

Adversarial Training and Unsupervised Learning for Populations

The expressibility of an inference model is severely constrained by the functional form, which specifies both the likelihood, the prior, and in variational approximate inference, the approximate posterior. The main driving force behind this form is tractability and computational complexity at inference time. Although Bayesian non parametrics promise to extract more knowledge from data, these algorithms are still limited by the forms imposed on the process priors, as well as limited in practice, as sophisticated methods such as the introduction of auxiliary variables and inducing points are required in order as to render the computations of the averages and conditionals possible. After the enormous theoretical effort, one is confined to the domain applicability of the model at hand. Hence, one would like to shed away from the theoretical complications and still retain the model flexibility promised by nonparametrics methodology. In recent years, a new family of implicit methods, have arisen which deliver these desired characteristics. They are implicit as they do not require explicit forms for any of the components (likelihood or posteriors). Like MCMC methodology, they recover the data distribution by allowing the practitioner to sample from the data distribution.

Generative Adversarial Networks, or GANs for short, remove the requirement of explicit posterior or conditional forms and deliver high flexibility. Similar to the Variational Autoencoder family, GANs exploit the approximation capacity of multilayer perceptrons. Now, the approximate form of the data distribution.

$$s = \Phi_\theta(\cdot, \epsilon), \qquad \epsilon \sim p_\epsilon(\epsilon), \tag{93}$$

where $p_\epsilon(\epsilon)$ is a simple distribution, such an isotropic Gaussian or uniform distribution. The generator $\Phi_\theta(\cdot, \epsilon)$ is specified as a deep neural network with model parameters θ. This allows us to implicitly learn the true distribution of the data by defining the true data distribution approximation through a nonparametric generator. Although the generator has parameters θ the solution is nonparametric. One does not have direct access to the distributions, and the MLP parameters serve as an intermediary of the generator. Instead of learning the distribution, we approximate it by generating samples that are similar to the true distribution.

Since we are not able to evaluate the likelihood, a new training strategy is required. A discriminator network $D(\cdot)$ is introduced, which is trained to discern between real data and the one sampled by our generator. Training is accomplished via the game defined in the following min-max objective:

$$\min_G \max_D \mathbb{E}_{x\sim P_r(x)}[\log D(x)] + \mathbb{E}_{x\sim P_\epsilon(\epsilon)}[\log(1 - D(G(\epsilon)))] \tag{94}$$

The end solution of this game will minimmize the Jensen divergence between the data distribution and that attained by the generator. Computationally this approach will profit from the samek advantages of Deep Neural Networks as one is able to leverage the computational capacity of GPUs with the matrix multuiplication requiered in the applications of the back propagation algorithm. In the following chapter, we will make use of this methodology for the study of stochastic processes. Specifically, we will return our attention to point processes in the domain of queues systems. We are now interested in predicting the time a system can provide a service to a set of costumers. To study the dynamics, one is interested in the arrival time of the customers and the system's service time for each customer. We will exploit the adversarial methodology in a conditional sense. Given a representation of the arrival process (obtained in terms of neural networks), we will sample the service time using an adversarial generator. Now, this generator not only depends on the noise ϵ but on the obtained representation.

Chapter 4

Recurrent Adversarial Service Times

In this chapter, we return our attention to the point process formalism. Now, we provide as a contribution a new methodology for the queueing problem, namely an implicit methods to sample from the service time of a system. In the application area intended, data is rich, and one can provide flexible descriptions with almost no prior knowledge. Queueing systems study the interplay between two processes, that of costumers arrivals, and that of systems services. To accommodate the information contained in the costumer's process, we work with a neural network description of the point process, the Recurrent Market Point Process. This framework delivers a representation of the data in a hidden state vector, customary in the recurrent network formalism. This hidden vector encodes information, which is then conditioned in an adversarial solution to the service distribution i.e. we leverage the representation for further modeling purposes, establishing the dependence among the costumer's services with this conditionals.

We proceed to motivate the application area. The ultimate success of any service provider rests on its ability to quickly and efficiently satisfy its customers: a mobility system is only successful as long as its users arrive on time; a block-chain is reliable provided low latency of its transaction times is ensured; Internet services can only retain users if they provide a quick and fast response. To operate systems like these, one needs to understand not only *when* customers will require a service but also *how* the system can react and respond to demands. Moreover, one should be able to adapt the system to external events dynamically. Examples include sudden disruptions of a mobility system due to a car crash or weather conditions; or financial crises and breaking news affecting block-chain transactions.

Recent research has focused primarily on the client-side of a service system. For instance, research on dynamical recommender systems aims at understanding the change in user preference over time. These changes, together with trends as to the popularity of items, determine suitable recommendations (Wu et al. 2017; Jing and Smola 2017; Zhou et al. 2018). Another variant of this line of research analyzes Customer Dynamics using point process theory, in which user behavior is modeled using parametric forms such as Poisson or Hawkes processes. However, these parametric forms constrain the model's expressibility to capture the users' dynamic behavior. This drawback has lately been tackled employing flexible non-parametric models such as recurrent neural networks and adversarial neural networks (Mei and Eisner 2017a; Du et al. 2016b; Xiao et al. 2017).

Another line of research focuses on the service side of the system, in particular on service times. Corresponding approaches typically resort to queuing theory: a customer expresses a demand, and the system responds according to the available servers and server load. Within the queuing theory, the customer arrival dynamics are modeled with a point process and, yet again, parametric forms are assumed. Results are usually limited to moments of the system size distribution (Daw and Pender 2018; Asmussen 2008) or are, in some cases, based on Bayesian inference (Sutton and Jordan 2011). The latter is often neither flexible nor scalable enough to handle the millions of customers in modern service systems. The work presented here overcomes both of these limitations

by combining recurrent neural networks (RNN) for modeling the customer arrival process with flexible service time distribution models.

In recent years, adversarial methods have shown to capture complex data distributions, achieving impressive results in image and text generation (Goodfellow et al. 2014a). Here we use these methodologies to model *conditional* service time distributions, exploiting the information encoded in the customer arrival dynamic representations learned via RNNs.

We summarize the contributions of the present chapter as follows:

- **First deep solution to service times for queue systems**: to the best of our knowledge, this is the first approach exploiting the representation learning capabilities of deep neural networks for point processes to infer service time distributions modelled as both, multilayered parametrizations of known distributions or non parametric models through adversarial neural networks. The adversarial models capture multi modal and long tail distribution of service times.
- **General solutions**: our methodologies deliver a holistic solution for general families of arrival and service processes, superior to classical theoretical models that are constrained to some specific nature of the arrival or service process.
- **Dynamic Services**: we introduce solutions that characterize the independent dynamics of the service times, allowing for exogenous events to be characterized implicitly.
- **Bitcoin Mempool**: To the best of our knowledge, we provide the first deep and nonparametric solution for the prediction of the unconfirmed transactions in the network.
- **Predicting and sampling from point process**: additionally, we provide a new general framework for prediction of- and sampling from recurrent point process models.

Our work is divided as follows: In Section 2 we present the theoretical basis of our models which are then introduced in Section 3. Section 4 contains the empirical evaluation of our approach and we conclude in Section 5.

4.1 Related work

The theory of queues has a long history of both exact and approximated results, and these apply to single queues as well as to networks of them (Williams 2016). Often these studies focus on Markovian queues, i.e. queues for which both inter-arrival and service times are exponentially distributed. In such cases the queue-length process (that is, the number of clients in service process) is given by a continuous-time Markov chain, which makes it an object suitable for theoretical analysis (Bertsimas 1990). For example, recent efforts investigate infinite-server queues whose arrivals follow a (Markovian) Hawkes process, and whose service time distributions are either phase-type or deterministic (Daw and Pender 2018). Their results include exact moments and the moment generating function of the queue-length process. Similarly, infinite-server queues whose arrival process is driven by a Cox process have also been considered (Boxma et al. 2019). Different from this works, our approach is able to generate holistic solutions for general families of arrival and service distributions.

In contrast to queue-length processes, the departure process is *not* in general a discrete-time Markov chain, even for Markovian queues. This makes sense, simply because keeping a client longer in service, or finishing serving her sooner, can cause slow reallocation of resources which, in turn, may affect the departure times of clients who have arrived much later. Despite this fact, Sutton and Jordan (2011) were able to infer service time distributions using Markov Chain Monte Carlo methods, to model queuing networks with missing data. In fact, they argued that such long-range non-Markovian effects within the departure process take place only for “large" departure times, which were unlikely to occur. Their approach, however, is not scalable enough to handle the large datasets of modern service systems which, in turn, may encompass different time scales and worsen the non-Markovian effects inherent in the departure process. We avoid these issues altogether by directly learning a generative model of the service time distributions.

More recently, and most related to our work, Chapfuwa et al. (2018) modeled event-time distributions using covariate information via Generative Adversarial Networks (GANs). We build on top of this work to capture the dynamic character of the client arrival process, and the independent dynamics of the server system. Finally, early approaches use neural networks as meta-models for queuing networks (Chambers and Mount-Campbell 2002). Different from this research, we use deep neural networks to infer service time distributions.

4.2 Background

In this section we introduce the basic concept of queues and the underlying theory of point processes. We also introduce the Recurrent Point process model (RPP) (Du et al. 2016b), which serves as a starting point to our service time models. The RPP model provides a rich representation of the dynamics of costumers arrivals, modeling conditional intensity functions with non parametric state transitions via recurrent neural networks.

4.2.1 Queues

In order to define a queuing system one must specify the nature of the client arrival process, as well as the specifics of the service system. The arrival of clients may follow a Poisson process, but its rate may jump due to external events. Likewise, one system might allow for a fixed number of clients to be served at a time, or the service times might dynamically change with every incoming client.

The standard notation used to specify the characteristics of the different queuing systems consists of characters separated by slashes thus $\cdot/\cdot/\cdot$ (Kendall 1953). The first character describes the

customer arrival process, namely the inter-arrival distribution. Typical examples are "M" for memoryless (Poisson), "D" for deterministic times and "G" for general distributions. The second character specifies the service time distribution, and the third one the number of servers available to the system. For example, the $M/M/1$ queue denotes a queuing system with Poisson arrivals, exponentially distributed service times and a single server.

Let us denote the arrival time of the ith client as $a_i \in \mathbb{R}^+$. After arriving to the system, the client waits to be served up until the service dynamics chooses to start the corresponding service. We denote this waiting as $w_i \in \mathbb{R}^+$.

Once the service is completed. If for a given arrival a_i no departure is observed we set the departure time $d_i = \infty$. We define $\mathcal{D}$ as the set of *uncensored events*, i.e. the set of arrivals which have departure within our observation time. Accordingly, we define $\mathcal{C}$ as the set of *censored events* which have no departure within this window.

the client leaves the system at departure time $d_i \in \mathbb{R}^+$. The sequence of departure times also defines a point process, to which we refer in the following as *departure process*. The service time $s_i \in \mathbb{R}^+$ is defined as the amount of time the ith client spends being served after the waiting period, that is $s_i = d_i - a_i - w_i$. Finally, the response time $r_i \in \mathbb{R}^+$ corresponds to the total time the customer requires to be processed, including the waiting time i.e. $r_i = d_i - a_i$.

In this work we provide a general solution to the service time distribution of the $G/G/\infty$ queuing problem[1].

Service Time Distributions

We now explain two of the most common general service models. The practitioner uses these models in order to obtain theoretical results on the matter. We are going to use them in the next sections as ways to validate our methodology. We will use these models to create synthetic data upon which we will train our model.

Phase Distribution: Here we decompose the service as a series of exponential service steps. It is defined with the time taken between the initial state and the absorbing state [2] in a continuous time Markov chain. It can be shown that such *phase type* distributions can approximate any non negative continuos distribution. For n phases, given a continuous time Markov chain (CTMC)

$$\mathbf{\Gamma} = \begin{bmatrix} 0, \mathbf{0} \\ \mathbf{s}, \mathbf{S} \end{bmatrix}, \tag{95}$$

where $\mathbf{S}$ and $\mathbf{s}$ corresponds to a Markov chain's transitions probabilities, the $\mathbf{0}$ elements specify the absorbing state nature, since there is no probability for leaving those states. These states are synthetic and are only intended to define an approximation to how time evolves within the system.

Processor Sharing: A processor sharing queue is a queue model where the system handles infinite clients simultaneously but must reallocate resources with each new client arrivals or departure. It was introduced by Kleinberg (cite) to model computer systems that handle multiple clients simultaneously. One can think that each client in the system at any time instantaneously receives $1/U(t)$ service power, where $U(t)$ is the number of unfinished clients (clients in the queue),

$$s_i = \int_{a_i}^{d_i} U^{-1}(t) dt \tag{96}$$

[1] Due to the non-parametric nature of our solutions, the rich distributional forms we obtain allow for a solution of the $G/G/k$ problem, where the number of available servers "k" is unknown. Under this interpretation of the data, one should reinterpret our results as delivering the response time of the system.

[2] a type of first passage time

if one samples *a priori* the service times s_i, the departure times are specified as a solution to the system of equations defined with Equation 96 and

$$U(t) = \sum_{i=1}^{N(t)} \mathbb{1}_{\{a_i<t\}} \mathbb{1}_{\{t<d_i\}} \tag{97}$$

For the arrival process we will make use of the following models: (1) **Arrival processes**: we consider two different arrival processes, namely (i) the Hawkes Process (H) as presented in section 2.1.3, (ii) the Non-linear Hawkes Process (NH) (Zhu 2013), which is an extension of the Hawkes process that allows for inhibitory behavior through a non-linear function over the history of arrivals. $\lambda_{NH}(t) = \phi\left(\lambda_H(t)\right)$.

See (Sutton and Jordan 2011). The combinations of the two arrivals and the two service models provide four different synthetic queues (**H-PT**, **H-PS**, **NH-PT**, **NH-PS**). We will use the models in order to test our methodology after our model is presented.

4.2.2 Recurrent Point Process

Here we write the likelihood of the Poisson Process from an intensity defined through a recurrent neural network following the procedure stated in (Mei and Eisner 2017b), (Du et al. 2016c). We also follow our discussion in Section 2.1.1, and restate some formulae for ease of presentation. We define a point process on a compact support $\mathcal{S} \subset \mathbb{R}$. Formally, the likelihood is written down as an inhomogeneous Poisson Process between arrivals conditioned on the filtration $\mathcal{H}_j \equiv \{a_1, ..., a_j\}$[3] (Daley and Vere-Jones 2007a). For one dimensional processes the conditional intensity function reads

$$f^*(t) = \lambda^*(t) \exp\left(\int_{a_j}^{t} \lambda^*(t')dt'\right). \tag{98}$$

The functional dependence of the intensity function is given by a recurrent neural network (RNN) with hidden state $\mathbf{h}_j$, where an exponential function guarantees that the intensity is *non-negative*

$$\lambda^*(t) = \exp\left\{\mathbf{v}^t \cdot \mathbf{h}_j + w^t(t - a_j) + b^t\right\}. \tag{99}$$

Here the vector $\mathbf{v}^t$ and the scalars w^t and b^t are trainable variables. Bare in mind that although recurrent networks (Graves 2013) are defined over *sequences*, the Point Process likelihood Eq. (98) requires evaluation of the function over the whole support $\mathcal{S}$. Both approaches (Mei and Eisner 2017b) (Du et al. 2016c) bypass this problem by defining decaying continuous values between two arrivals a_j and a_{j+1}, either for *memory cells* in LSTMs or hidden layers. The update equation for the hidden variables of the recurrent network can be written as a general non linear function $\mathbf{h}_j = f_\theta(\mathbf{h}_{j-1}, \mathbf{a}_j)$ where θ denotes the networks parameters. Once we perform the integration in Eq. (98) one obtains

$$f^*(t) = \exp\left\{\mathbf{v}^t\mathbf{h}_j + w^t(t - a_j) + b^t + \frac{1}{w^t}\exp(\mathbf{v}^t \cdot \mathbf{h}_j + b^t) - \frac{1}{w^t}\exp(\mathbf{v}^t \cdot \mathbf{h}_j + w^t(t - a_j) + b^t)\right\}.$$

We can learn the model parameters by maximizing the joint model likelihood $\mathcal{L}_{\text{RPP}} = \sum_i \log f^*(\delta_{i+1}|\mathbf{h}_i)$ where $\delta_{i+1} = a_{i+1} - a_i$ denotes the interarrival time.

Prediction and sampling.

For both prediction and sampling we require $P(T|\mathcal{H}_j)$, the distribution that the next point arrives at T given the previous history until a_j. First, notice that the probability of no point arriving

[3] history of arrivals

between a_j and $a_j + \tau$ can be obtained as an integral over $P(T)$, say

$$\exp\left\{-\int_{a_j}^{a_j+\tau} \lambda(t)dt\right\} = \int_{\tau}^{\infty} P(T)dT \equiv G(\tau), \tag{100}$$

$$P(T) = -\frac{dG(T)}{dT}, \tag{101}$$

where we used the Poisson distribution for zero arrivals in the first expression. Solving for $G(\tau)$ we find

$$G(\tau) = \exp\left\{-e^{\alpha_j}\frac{1}{w^t}[e^{w^t\tau} - 1]\right\}, \tag{102}$$

with $\alpha_j = \mathbf{v}^t\mathbf{h}_j + b^t$. The average time of the next arrival is then given by

$$\mathbb{E}[T] = \int_0^{\infty} P(T)T\,dT = \int_0^{\infty} G(T)\,dT, \tag{103}$$

where we used both Eq. (101) and Eq. (102). Finally, in order to sample the next arrival time one can use inverse transform sampling on $P(T)$. To this end one requires the inverse of the cumulative function of $P(T)$. We calculate the cumulative function thus

$$F[P(T)] = \int_0^{\tau} P(T)dT = -\int_0^{\tau} \frac{dG(T)}{dT}dT = G(0) - G(\tau) \tag{104}$$

whose inverse function then follows

$$F^{-1}[P(T)](y) = \frac{1}{w^t}\left\{-\alpha_j + \log\left(w^t\left(\log\left(\frac{-1}{y-1}\right) + \frac{e^{\alpha_j}}{w^t}\right)\right)\right\}. \tag{105}$$

4.3 Models: Deep Service Times

In this section we introduce our models. Classical queueing studies follow a traditional form in which results are stated in terms of moments of distributions (service or waiting times). Due to the general flexibility of adversarial approaches, one is able to provide a more rich take on the matter. This makes comparision and validation difficult specially in our application areas where one does not know the direct model to compare against, given that there are a wide range of unknowns in the data. To overcome this, and to provide a more fair comparison. We start by introduing what we call the Neural Service times. This first models, make use of the advantages of neural networks by providing a flexible parametrization of classical distributions which are known as stationary solutions in the classical models. This guarantees that the classical knowledege is leverages upon the more modern approach. Our final solution, in the realm of adversarial training also posses deep learning characterictics, do to this general use of the functional approximations, we named our family of models under the umbrella of **Deep Service Times**.

Fig. 12 shows an overview of our service time distribution models, which take the hidden state of a *trained* RPP model as input thus providing a rich representation of the customer arrival dynamics. We provide two methodologies: (i) we propose parametric forms for the service time distribution where the parameters of known survival distributions are defined by multilayered perceptrons. We shall refer to these model as neural service (**NS-X**) models, where the **X** labels an specific survival distributions (see Section 4.3.1). This set of models are a natural generalization of classical stationary solutions to the queue problem (Kingman 1993) and will serve as our baseline models in what follows; (ii) we propose two adversarial solution: first, a static one in which the dynamics of the system is encoded only through the arrivals RPP process (called **AS** model). Second, a dynamical model, wherein the adversarial generator encodes the dynamics of the systems service via a non parametric state transition function parametrized by a recurrent neural networks (called **RAS** model).

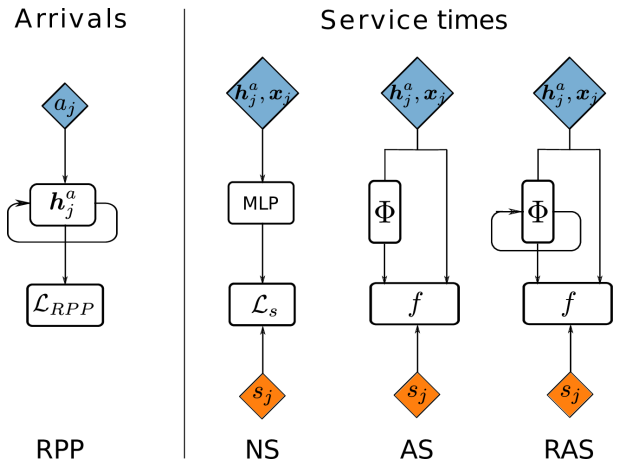


Figure 12 – Deep service time models. Left panel: the costumer arrivals a_j are modelled using the recurrent neural point process (RPP) with hidden state $\mathbf{h}_j$. This model is trained by maximizing the joint log-likelihood L_{RPP} defined in Section 4.2.2. Right panel: the service time models take the hidden state of the arrival model $\mathbf{h}_j$ and covariates $\mathbf{x}_j$ as input and infer the service time distribution. These models are trained either by maximizing the Log-Likelihood L_s or by solving a minimax game between critic f_φ and generator Φ_θ.

4.3.1 Neural Service Times

We start with the costumer arrival point process a_i. We consider the RPP model as defined in Eq. (99) and denote its hidden state representation as $\mathbf{h}_j^a = g_\eta(\mathbf{a}_j, \mathbf{h}_{j-1}^a)$, where η labels the set of parameters of the RPP network.

To model the distribution of times each costumer will be in service we introduce the generative model

$$s_i \sim \Phi_\theta(s|\mathbf{h}_i^a, \mathbf{x}_i),$$

with parameter set θ. This model capture the complicated dependencies in the arrivals dynamics a_i — encoded through the hidden states $\mathbf{h}_i^a$, and any other covariates $\mathbf{x}_i$ in the system. This conditional form allows our model to leverage the dynamical information of the arrival process. We define Φ_θ as one of the following five distributions: Gamma (G), Exponential (E), Pareto (P), Chi-square (C) or Log-normal (L), whose parameter set $\mathcal{P}$ are defined via multilayer perceptrons. Thus for the **NS-G** model we have $s_i \sim \text{Gamma}(\alpha_i^a, \beta_i^a)$ with $\mathcal{P} = [\alpha_i^a, \beta_i^a] = \text{MLP}_\theta([\mathbf{h}_i^a, \mathbf{x}_i])$. The neural service models can then be interpreted as a marked RPP where the marks are continuous and have a dynamical character whose distribution corresponds to that of the service times. We train these models by maximizing the log-likelihood of our generated service times with respect to the uncensored data set $\mathcal{D}$.

Censored events. To capture censored events we introduce the probability of obtaining an expected remaining service time bigger than the observation window $T_i = T - a_i$

$$\bar{\Phi}(s_i) = \int_{T_i}^{\infty} \Phi(\tau|, \mathbf{h}_i^a, \mathbf{x}_i) d\tau \,.$$

The complete log-likelihood of the **NS-X** model then reads

$$\mathcal{L}_s = \sum_{\mathcal{D}} \log \left\{ \Phi(s_i|\mathbf{h}_i^a, \mathbf{x}_i) \right\} + \sum_{\mathcal{C}} \log \left\{ \bar{\Phi}(s_i) \right\}.$$

4.3.2 Adversarial Service Times

As stated in the introduction of the current section, the expressibility of our model is severely constrained by any functional form imposed on $\Phi(s)$. In order to allow for more general service time

distributions we consider Generative Adversarial Networks (GAN) (Goodfellow et al. 2014a), we now adapt the general formulation as presented in Eq. 93, the approximate (service time) samples s_i are drawn as

$$s_i = \Phi_\theta(s|\epsilon, \mathbf{h}_i^a, \mathbf{x}_i), \qquad \epsilon \sim \mathbb{P}_\epsilon. \tag{106}$$

Here $\mathbb{P}_\epsilon$ is a simple distribution, e.g. isotropic Gaussian or uniform distribution, and the *generator* Φ_θ is modeled by a deep neural network with parameter set θ, conditioned on both the arrival dynamics and the system's covariates. In our experiments we define Φ_θ as a 3-layer perceptron and add a noise term drawn from a Gaussian $\mathcal{N}(0,1)$ at each of this layers, as to increase the variance in the samples from Φ_θ (Chapfuwa et al. 2018).

This class of models are trained by minimizing specific distances (or divergences) between the empirical distribution — here the distribution of *uncensored* events $\mathbb{P}_\mathcal{D}$, and the distribution $\mathbb{P}_\theta$ of the generated samples $\{\Phi_\theta(s)\}$. Each such distances differ on the impact they have on the convergence of $\mathbb{P}_\theta$ towards the empirical distribution, and thus on the training stability. Here we choose to minimize the Wasserstein-1 distance (WGAN) (Arjovsky et al. 2017), which has been shown to be continuous everywhere and differentiable almost everywhere, as opposed to e.g. the Jensen-Shannon divergence minimized in the original GAN formulation.

Using the Kantorovich-Rubinstein duality (Villani 2009) to compute the Wasserstein-1 distance one can express the WGAN objective function $\mathcal{L}$ as:

$$\mathcal{L} = \min_\theta \max_{f_\varphi \in \mathfrak{L}_1} \mathbb{E}_{s\sim\mathbb{P}_\mathcal{D}}\left[f_\varphi(s)\right] - \mathbb{E}_{s\sim\mathbb{P}_\theta}\left[f_\varphi(s)\right], \tag{107}$$

where the maximum is over all 1-Lipschitz functions $\mathfrak{L}_1$, defined as functions whose gradients have norms at most 1 everywhere. Within the WGAN formulation the critic function f_φ is modeled by a deep neural network with parameter set φ and needs to fulfill the Lipschitz constrain. In order to enforce it we follow (Petzka et al. 2018) and add a regularization term of the form

$$\mathcal{L}_1 = \mathbb{E}_{s\sim\mathbb{P}_i}\left[\left(\max\left\{0, |\nabla_s f_\varphi(s)| - 1\right\}\right)^2\right], \tag{108}$$

where $\mathbb{P}_i$ is implicitly defined as sampling uniformly along straight lines between pairs of points sampled from the empirical $\mathbb{P}_\mathcal{D}$ and the generator $\mathbb{P}_\theta$ distributions (Gulrajani et al. 2017). Minimizing Eq. (107) under an optimal critic function with respect to θ minimizes the Wasserstein distance between $\mathbb{P}_\mathcal{D}$ and $\mathbb{P}_\theta$ — this defines the adversarial game.

In our experiments the critic is defined as a 3-layer perceptron and is also conditioned on the covariates $f_\varphi = f_\varphi(s, \mathbf{x})$.

Censored events. To train Φ_θ to learn the distribution of censored events $\mathbb{P}_\mathcal{C}$ we follow (Chapfuwa et al. 2018) and consider a second regularizer which penalizes sampled service times smaller than the censoring time T, that is

$$\mathcal{L}_2 = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_\mathcal{C}, \epsilon\sim\mathbb{P}_\epsilon}\left[\max\left\{0, T - \Phi_\theta(s|\epsilon, \mathbf{h}^a, \mathbf{x})\right\}\right]. \tag{109}$$

We also correct for situations in which the proportions of uncensored events is low through

$$\mathcal{L}_3 = \mathbb{E}_{\epsilon\sim\mathbb{P}_\epsilon, (\tilde{s},\mathbf{x})\sim\mathbb{P}_\mathcal{D}}\left[\mathrm{abs}\left\{\tilde{s} - \Phi_\theta(s|\epsilon, \mathbf{h}^a, \mathbf{x})\right\}\right]. \tag{110}$$

Our full objective function reads $\tilde{\mathcal{L}} = \mathcal{L} + \sum_{i=1}^{3} \lambda_i \mathcal{L}_i$, where $\lambda_1 = 10$ and $\lambda_3 = 1$ throughout all experiments whereas λ_2 changes depending on the data sets.

4.3.3 Recurrent Adversarial Service Time

The response of a service systems to newly arrived customers intuitively has a dynamic component (e.g. the dynamic reallocation of resources depending on the number of arrivals still on service, the response to past events disrupting the service, etc). In order to both capture such a dynamic

Table 4.1 – Main results on synthetic data sets.

	NH-PT		NH-PS		H-PT		H-PS	
mean	0.052		0.0004		0.0061		2.125e-5	
	error	KS	error	KS	error	KS	error	KS
NS-G	**0.207**	0.218	0.020	0.520	0.254	0.879	**2.18e-5**	0.401
NS-E	0.209	0.154	0.0006	0.082	0.432	0.979	3.06e-5	0.501
NS-P	0.210	0.330	0.0007	0.610	0.443	0.981	2.09e-5	0.501
NS-C	0.372	0.242	0.061	0.527	0.037	0.988	0.029	0.511
NS-L	5.293	0.525	6.158	0.479	4.282	0.971	8.500	0.555
ATE	0.219	0.193	0.0371	0.870	0.348	0.376	3.96e-3	0.199
RATE	0.218	0.124	0.0031	0.062	0.302	0.315	1.37e-4	0.136
AS	0.215	0.113	0.0016	0.448	0.250	0.235	1.24e-4	0.121
RAS	**0.207**	**0.094**	**0.0005**	**0.042**	**0.242**	**0.212**	1.09e-4	**0.110**

Table 4.2 – Main results on empirical data-sets.

	Github		NY		Stackoverflow	
mean	0.0113		0.0068		0.0193	
	error	KS	error	KS	error	KS
NS-G	0.073	0.396	0.025	0.154	0.378	0.480
NS-E	0.074	0.458	0.007	0.251	0.379	0.509
NS-P	0.071	0.604	0.008	0.367	0.378	0.466
NS-C	0.096	0.341	0.182	0.632	0.403	0.533
NS-L	6.37	0.496	0.155	0.627	15.93	0.595
ATE	0.217	0.517	0.078	0.126	0.382	0.233
RATE	0.112	0.240	0.098	0.165	0.388	0.492
AS	**0.071**	0.039	**0.006**	0.094	0.383	**0.226**
RAS	0.072	**0.034**	**0.005**	**0.030**	**0.369**	0.281

response and implicitly characterize exogenous events we approximate the system's transition function with a *stochastic* recurrent neural network

$$\mathbf{h}_i^\Phi = g_\theta(\epsilon_i, \mathbf{h}_i^a, \mathbf{x}_i, \mathbf{h}_{i-1}^\Phi), \qquad \epsilon_i \sim \mathbb{P}_\epsilon, \tag{111}$$

where g_θ is a RNN with parameter set θ and $\mathbf{h}_i^\Phi$ is the hidden state encoding the independent dynamic character of the service system. The model is informed about the incoming arrival through the hidden state $\mathbf{h}_i^a$ of the arrival RPP model and the arrival covariates $\mathbf{x}_i$. Its noisy component, on the other hand, comes from $\mathbb{P}_\epsilon$, an isotropic Gaussian sampled at each arrival time.

We then define the generator $s_i = \Phi_\theta(s|\mathbf{h}_i^\Phi, \epsilon)$ as a 3-layer perceptron with the RNN's hidden representation Eq. (111) as input and an additional noise terms ϵ added in each layer. We train the model by minimizing Eq. (107) together with the regularizers $\mathcal{L}_i$ as above. Let us note here that recurrent generative models with adversarial training has been considered before (Mogren 2016), (Hyland et al. 2018). In our experiments the critic function $f_\varphi = f_\varphi(s, \mathbf{x}_i)$ remains static and is defined once more as a 3-layer perceptron.

Alg 1 summarizes the RAS model algorithm. Note that the parameters of the RPP model (ρ in line 9 of Alg 1) are fixed and optimal.

4.3.4 Bitcoin Mempool

Bitcoin is a complex protocol. We provide here a brief sketch of the components necessary to understand the analysis presented in this work, however due to the many moving parts of the

Algorithm 1: Recurrent Adversarial Service Time

Data: Dataset $\mathcal{D} = \{(a_i, s_i, \mathbf{x}_i)\}_{i=1}^{N}$; Critics Iterations per Generator Iterations n_c
while θ not converged **do**
  for $i = 1, ..., N$ **do**
    Draw $\{(a_i, s_i, \mathbf{x}_i)\} \sim P_{\mathcal{D}}$
    for $k = 1, ..., n_c$ **do**
      Draw $\epsilon_i, \tilde{\epsilon}_i \sim P_\epsilon$
      Draw $\delta \sim \text{Uniform}(0, 1)$
      Update $\mathbf{h}_i^a \leftarrow g_\rho(a_i, \mathbf{h}_{i-1}^a)$
      Update $\mathbf{h}_i^\Phi \leftarrow g_\theta(\epsilon_i, \mathbf{h}_i^a, \mathbf{x}_i, \mathbf{h}_{i-1}^\Phi)$
      $\tilde{s}_i \leftarrow \Phi_\theta(\tilde{\epsilon}_i, \mathbf{h}_i^\Phi)$
      $\hat{s}_i \leftarrow \delta s_i + (1 - \delta)\tilde{s}_i$
      $L_i \leftarrow f_\varphi(\tilde{s}_i) - f_\varphi(s_i)$
      $+\lambda \left(\max\left\{0, |\nabla_{\hat{s}_i} f_\varphi(\hat{s}_i)| - 1\right\}\right)^2$
      $\varphi \leftarrow \text{Adam}(\nabla_\varphi L_i)$
    Draw $\epsilon_i \sim P_\epsilon$
    $\theta \leftarrow \text{Adam}(\nabla_\theta(-f_\phi(\Phi_\theta(\epsilon_i, \mathbf{h}_i^\Phi))))$
return Φ_θ

system this is necessarily a superficial overview. Interested parties are referred to (Nakamoto 2008) for a more complete picture of the system. The decentralized currency protocol known as Bitcoin was proposed by Satoshi Nakamoto (Nakamoto 2008). The system utilizes a peer-to-peer (P2P) architecture that enables users to send and receive transactions denominated in units of BTC. Users are represented in the network by a public/private key pair. Units of BTC can be transmitted to a user by specifying a hash of that user's public key as the receiving party, providing a degree of pseudo-anonymity. Users can generate many public keys, i.e. receiving addresses. The corresponding private keys are used to sign (authorize) transactions. Private keys are stored in a "wallet" either locally or provided as a hosted service. To participate in the Bitcoin network the user runs a client software, such as the Satoshi client, which communicates with a set of peers. Transactions are broadcast by the Bitcoin client and received by the peer-to-peer network. They are confirmed after having been added to the "blockchain" - similar to a linked list with the subtle difference that it references the previous block using its hash rather than a pointer. This data structure contains blocks of all accepted transactions since the genesis of the system.

Every full node running a Bitcoin client maintains a complete copy of this public blockchain. The block generation process confirms new transactions. It necessitates the satisfaction of a computationally expensive "proof of work" puzzle. A valid solution to this puzzle entitles the party that deliverers it to the wider network the privilege to issue themselves a reward in the form of newly minted coins. The information available through the graph structure of the Bitcoin P2P network is limited due to the dynamic block formation process. Each node only has direct knowledge of the peers to which its client is connected. The graph of all transactions can be constructed entirely from the publicly available blockchain, wherein the nodes of the graph correspond to Bitcoin addresses and the edges to transactions performed between those addresses. In this work we empirically study significant global properties of the Bitcoin transaction graph.

With a total market capitalization in excess of $100,000,000,000 (ElBahrawy et al. 2017) Bitcoin is the world's largest blockchain-based cryptocurrency. The plethora of alternative public blockchain-based cryptocurrencies, many of which are based largely on the open source Bitcoin specification, are amenable to the analyses herein presented.

In the following we modify our approach to analyze data from a specific type of queue: the transaction queue of the Bitcoin network. Bitcoin transactions are confirmed after having been added to the *Blockchain*. The creation of each block defines a point process which can be understood as a departure process for the transactions, thus encouraging the modelling of the system dynamics as a queue system. Specifically, we analyze the Bitcoin *mempool*, the set of unconfirmed transactions

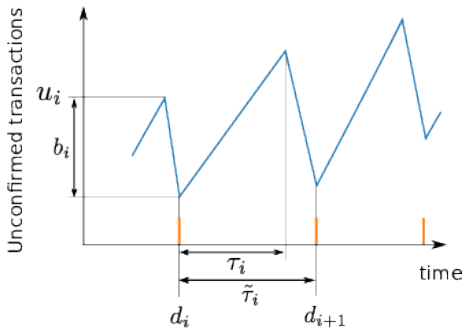


Figure 13 – Explanation of the mempool dataset creation (a), Raw Mempool Data View

u in the Bitcoin network. Here, the creation of a block at time d_i generates a sudden drop b_i in the number of unconfirmed transactions u_i (Fig. 13). The set of unconfirmed transactions plays the role of *waiting clients*, and the creation of a block specifies the simultaneous *departure* of many clients (transactions).

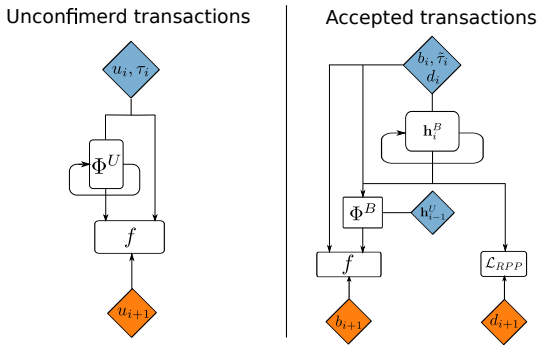


Figure 14 – Deep service time models overview. Panel (a) (Left): the customer arrivals a_i are described with the RPP model with hidden state $\mathbf{h}_i^a$. It is trained by maximizing $\mathcal{L}_{\text{RPP}}$. (Right): The service time models take $\mathbf{h}_i^a$ and covariates $\mathbf{x}_i$ as input. They infer the service time distribution either by maximum likelihood (NS) or adversarial training (AS and RAS model)

We first model the independent dynamics of the number of unconfirmed transaction with the generative model

$$u_{i+1} = \Phi_\theta^U(u|\mathbf{h}_i^U), \qquad \mathbf{h}_i^U = g_\theta(\epsilon, \tau_i, u_i, \mathbf{h}_{i-1}^U). \tag{112}$$

Here g_θ is a RNN with parameters θ, $\mathbf{h}_i^U$ encodes the history of u_i and τ_i is shown in Fig. 17a. We present two approximations to the mempool problem: (i) a parametric one, which we called Neural Meempool Service (**NMS**) for which $\Phi_\theta^U = \text{Gamma}(\alpha_i^u, \beta_i^u)$ with $[\alpha_i^a, \beta_i^a] = \text{MLP}_\theta(\mathbf{h}_i^U)$ and $\epsilon = 0$; (ii) and a nonparametric one, which we called Adversarial Mempool Service (**AMS**), in which Φ_θ^U is given by a 3-layer perceptron and for which $\mathbf{h}_i^U$ is a now random variable with $\epsilon \sim \mathbb{P}_\epsilon = \mathcal{N}(0, 1)$ in Eq. (112).

Now, the creation of the blocks which form part of the Blockchain defines a departure process which we describe using a RPP model with intensity function

$$\lambda_d^*(t) = \exp\{\mathbf{v}_M^t \cdot \mathbf{h}_j^M + \mathbf{v}_U^t \cdot \mathbf{h}_j^U + w^t(t - d_j) + b^t\}, \tag{113}$$

where $\mathbf{h}_i^U$ contains the dynamic information of the unconfirmed transaction process whereas $\mathbf{h}_i^M = \tilde{g}_\phi(b_i, d_i, \tilde{\tau}_i, \mathbf{h}_{i-1}^M)$, with $\tilde{g}_\phi$ a RNN describing the departure dynamics. Finally we introduce a generative model for the accepted transaction thus

$$b_{i+1} = \Phi_\phi^M(b|\epsilon, \mathbf{h}_i^M, \mathbf{h}_{i-1}^U), \tag{114}$$

with $\epsilon = 0$ and Φ_ϕ^M a Gamma function in our **NMS** formulation, or $\epsilon \sim \mathcal{N}(0, 1)$ and Φ_ϕ^M a 3-layer

perceptron in our nonparametric **AMS** version. We train the **NMS** model via maximum likelihood and **AMS** using Eqs. 107 and 108; the RPP block-creation model is trained as described in Section 4.2.2. The complete overview of the Bitcoin Mempool models is given in Fig. 14 and the algorithm for the AMS model is given in what follows

We now present the algorithms for the two components of the Adversarial Mempool Service (AMS) model. Alg. 2 shows the algorithm for the Unconfirmed Transactions (UT) model. After training the latter we used its hidden states $\mathbf{h}_i^U$, which encode the history of unconfirmed transactions as input to the second component of the AMS model: the Block Marked-RPP (BMRPP) model. The algorithm for the BMRPP model is presented in Alg. 3.

Algorithm 2: Adversarial Mempool – Unconfirmed Transaction

Data: Dataset $\mathcal{D} = \{(u_i, \tau_i)\}_{i=1}^N$. Critics Iteration per Generator Iterations n_c

while θ *not converged* **do**
  for $i = 1, ..., N$ **do**
    for $k = 1, ..., n_c$ **do**
      Draw $\{(u_{i+1}, \tau_i)\} \sim P_{\mathcal{D}}$
      Draw $\epsilon_i, \epsilon_i' \sim P_\epsilon$
      Draw $\delta \sim \text{Uniform}[0, 1]$
      Update $\mathbf{h}_i^U \leftarrow g_\theta(\epsilon_i, \tau_i, u_i, \mathbf{h}_{i-1}^U)$
      $\tilde{u}_{i+1} \leftarrow \Phi_\theta^U(\epsilon_i', \mathbf{h}_i^U)$
      $\hat{u}_{i+1} \leftarrow \delta u_{i+1} + (1-\delta)\hat{u}_{i+1}$
      $L_i \leftarrow f_\varphi(\tilde{u}_{i+1}) - f_\varphi(u_{i+1}) + \lambda_1 \left(\max\{0, |\nabla_u f_\varphi(\hat{u}_{i+1})| - 1\}\right)^2$
      $\varphi \leftarrow \text{Adam}(\nabla_\varphi L_i)$
    end
    $\theta \leftarrow \text{Adam}(\nabla_\theta - f_\varphi(\Phi_\theta^U(\epsilon_i, \mathbf{h}_i^U)))$
  end
end
return Φ_θ^U

4.4 Experiments

In this section we provide the experimental framework upon which we tested our model. First we introduce the datasets which were use in the experiments. We provide synthetic datasets with established models for both the arrivals and the service processes, as well as empirical datasets which demonstrates the ability of our approach to handle diverse application areas in an flexible and scalable manner. Finally we specify the details of the neural networks architectures implemented for the experiments, as well as learning parameters and any other hyperparameters as required in the model specification above.

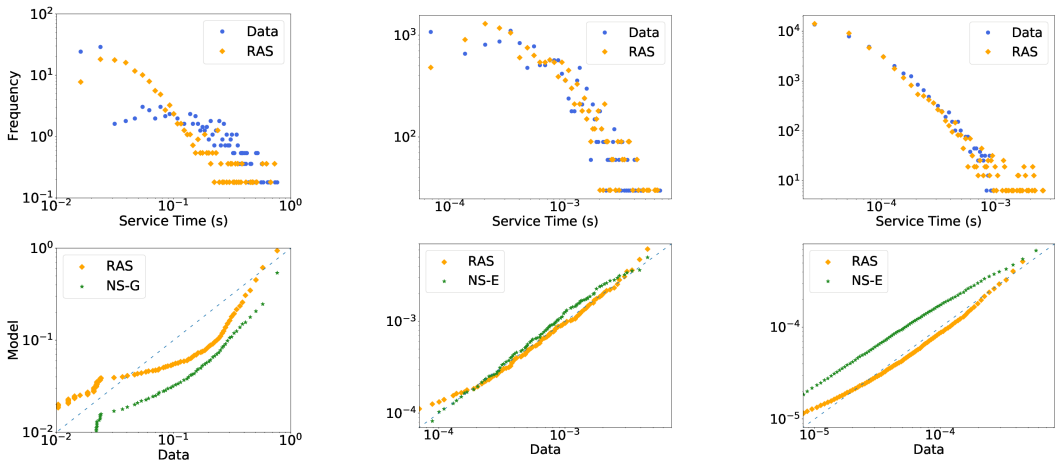
4.4.1 Empirical datasets

We now introduce the empirical datasets upon which we tested the proposed methodology. As such temporal point process are able to model a wide range of phenomena. In the realm of computer science, much interest is devoted to how internet services such as Google and Amazon, handle thousands of request per day. Queues network provide a model for this services. In our work however we concetrate on the area of the social web as well as mobility services. Here the service is an abstract representation to the users of webpages that provide as whole the service in the given web pages. To our knowledge this is the first work of its kind which provide an understanding of the social web as a queueing abstraction. Through this abstraction one is able to gain the richness of a decoupled description of the systems dynamics. Costumers and Service is differeciated and such an abstraction allows to highlight the interplay which gives rise to a notion of response, as

Algorithm 3: Adversarial Mempool – Block Marked-RPP

Data: Dataset $\mathcal{D} = \{(u_i, b_i, b_{i+1}, d_i, \tau_i, \tilde{\tau}_{i-1}, \mathbf{h}_i^U)\}_{i=1}^N$. RPP parameters $\Theta = \{\phi, \mathbf{v}_B^t, \mathbf{v}_U^t, w^t, b^t\}$. Critics Iteration per Generator Iterations n_c

while Θ, ρ *not converged* **do**
  for $i = 1, ..., N$ **do**
    Draw $\{(u_i, b_i, b_{i+1}, d_i, \tau_i, \tilde{\tau}_{i-1}, \mathbf{h}_i^U)\} \sim \mathbb{P}_{\mathcal{D}}$
    Draw $\epsilon_i \sim \mathbb{P}_\epsilon$
    Update $\mathbf{h}_i^B \leftarrow g'_\phi(b_i, d_i, \tilde{\tau}_{i-1}, \mathbf{h}_{i-1}^B)$
    $\lambda_d^*(t) \leftarrow \exp\{\mathbf{v}_B^t \cdot \mathbf{h}_j^B + \mathbf{v}_U^t \cdot \mathbf{h}_j^U + w^t(t - d_i) + b^t\}$
    $\mathcal{L}_{\text{RPP}}^i \leftarrow \log f^*(\delta_{i+1} | \mathbf{h}_i^U, \mathbf{h}_i^B)$
    $\Theta \leftarrow \text{Adam}(\nabla_\Theta \mathcal{L}_{\text{RPP}}^i)$
    for $k = 1, ..., n_c$ **do**
      Draw $\epsilon_i \sim \mathbb{P}_\epsilon$
      Draw $\delta \sim \text{Uniform}[0, 1]$
      $\tilde{b}_{i+1} \leftarrow \Phi_\rho^B(\epsilon_i, \mathbf{h}_i^B, \mathbf{h}_i^U)$
      $\hat{b}_{i+1} \leftarrow \delta b_{i+1} + (1 - \delta)\tilde{b}_{i+1}$
      $L_i \leftarrow f_\varphi(b_{i+1}) - f_\varphi(\tilde{b}_{i+1}) + \lambda_1 \left(\max\left\{0, |\nabla_b f_\varphi(\hat{b}_{i+1})| - 1\right\}\right)^2$
      $\varphi \leftarrow \text{Adam}(\nabla_\varphi L_i)$
    end
    $\rho \leftarrow \text{Adam}(\nabla_\rho - f_\varphi(\Phi_\rho^B(\epsilon_i, \mathbf{h}_i^B, \mathbf{h}_i^U)))$
  end
end
return Φ_ρ^B

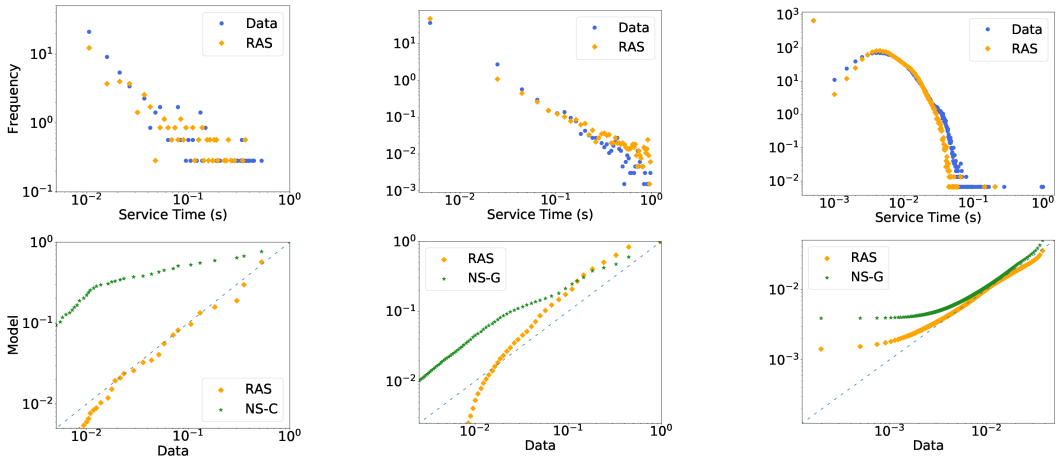


(a) NH-PT **(b)** NH-PS **(c)** H-PS

Figure 15 – Comparison among the probability distributions obtained form our best models (indicated in the panel caption) and the theoretical models (*pareto* - green, *lognormal*-blue and *gamma* - orange). The distributions are inferred on the sequence from the test set that yields the best KS value respectively for each dataset. The lines represent the distributions obtained by fitting the theoretical models using maximum likelihood.

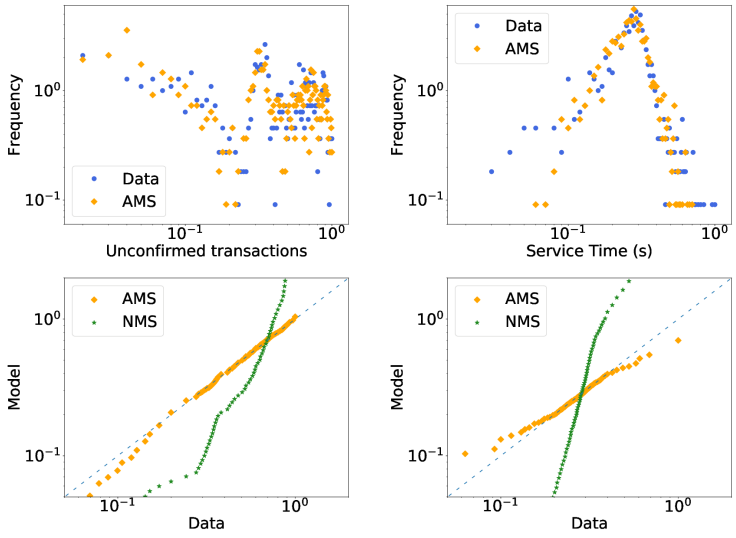
well as an intrinsic concept of system memory.

Stackoverflow: a questions answering platform for programmers. We define the costumers arrivals as the point in time when questions are posted by the users of the web page. We define the service time as the elapsed time between a question an its subsequent accepted answer time. This view, establish the ensemble of users which provide answers as the service system. We analyze a total of 2×10^7 questions.



(a) Github **(b)** Stackoverflow **(c)** NY

Figure 16 – Comparison among the probability distributions obtained form our best models (indicated in the panel caption) and the theoretical models (*pareto* - green, *lognormal*-blue and *gamma* - orange). The distributions are inferred on the sequence from the test set that yields the best KS value respectively for each dataset. The lines represent the distributions obtained by fitting the theoretical models using maximum likelihood.



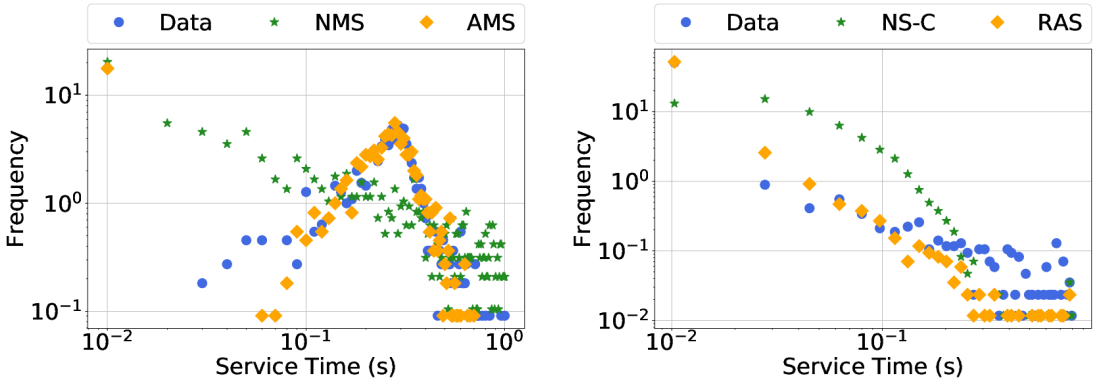
(a) Transactions Adversarial **(b)** Service Adversarial

Figure 17 – Comparison among the probability distributions obtained form our best models and the theoretical models (*pareto* - green, *lognormal*-blue and *gamma* - orange). The distributions are inferred on the sequence from the test set that yields the best KS value respectively for each dataset. The lines represent the distributions obtained by fitting the theoretical models using maximum likelihood. Inferred probability distributions for transactions (a) and services (b) datasets.

Github: The version control repository and internet hosting service. As costumers arrivals, we defined the creation of an issue in a given repository. Its departure time is the moment the given issue was closed. under this view, the set of costumers associated with a given repository, are though of the service system. We analysed the top 500 repositories in the platform in 2015 [4]. For a total of 1.5×10^6 different issues.

New York City Taxi Dataset (NY): The dataset contains data of individual taxi trips in New York city. Costumers arrivals are defined as starting time of the trip and the departure time

[4] ranked by the number of issues



(a) Mempool Service **(b)** Github

Figure 18 – Comparison of the probability distributions obtained from different models

is defined as the final time of the trip. Here, the service system is provided by both the taxi providing the service and the transportation network of roads, streets and highways pertaining to the Manhattan, Brooklyn, Queens, Bronx and Staten Island zones in the New York city. We analysed about 1.1×10^7 trips. [5]

Meempool: The dataset consist of all unconfirmed transaction in the mempool dataset as observed by one miner for the time period between January 2017 until June 2018. Here, the service system consists of the whole bitcoin miner network. This is the only data set which was used for the Bitcoin mempool model. And the details of the arrivals and service is provided in the model specification above. [6]

4.4.2 Training details and evaluation metrics

For the purpose of optimizing the neural networks parameters, we use the ADAM stochastic optimization (Kingma and Ba 2014) method with a learning rate of 10^{-4} (except for the neural mempool model where we use 10^{-5}). We split the data into training and test sets. The test set is defined as $\sim 5\%$ of the time series. For all the arrival models, we used the Gated Recurrent Units (Cho et al. 2014) for the non parametric state transition functions Eq. 4.2.2. The dimension of the GRU is 64 for the PS models, 16 for the PT and Github dataset, 128 for the New York dataset and 256 for the Stackoverflow dataset. For the RAS model, LSTMs where used as state transitions in the arrival model. The MLP for the NS model has two hidden layers with 256 dimensions, whereas the adversarial generator and the critic in the AS and RAS model have 3 hidden layers of 100 units. For the mempool dataset, the NMS model requires perceptrons with dimension 32 and LSTM units of size 16 for the unconfirmed transactions. For the service model the dimensions of the perceptrons is 32 and for the LSTM it is 62. The AMS model has 20 units for the unconfirmed transactions generator and critic, and 20 units for the transition function. For the service dynamics, the adversarial model required 10 hidden units for the generator and critic and 10 units for the LSTM transitions.

We evaluate the performance of our models using two different metrics evaluated on the test set. We consider the prediction error defined as $1/N \sum_i |s_i - \langle \tilde{s}_i \rangle|$, where s_i denotes the empirical value and $\langle \tilde{s}_i \rangle$ denotes the prediction obtained by Monte Carlo sampling, where we use the number of samples to be 20. Moreover, we calculate the Kolmogorov-Smirnov (KS) statistics between the empirical and the predictive distribution.

[5] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[6] https://jochen-hoenicke.de/queue

4.4.3 Results

In the experiments section below we shall compare our RAS model with the the Adversarial Time-to-Event (ATE) model of Chapfuwa et al. (2018). The latter can be understood as defined by Eq. (111), but with both $\mathbf{h}_i^\Phi$ and $\mathbf{h}_i^a$ replaced by $\mathbf{x}_i$. We also introduce RATE, a recurrent version of the ATE model, namely Eq. (111) without $\mathbf{h}_i^a$.

Table 4.3 – Mempool dataset results.

	Unconfirmed Transactions		Departures	
mean	0.2484		0.2159	
	error	KS	error	KS
PAR	0.2513	0.2417	0.1173	0.2781
LNO	0.2687	0.2527	0.1173	0.4760
GAM	0.1948	0.2550	0.1173	0.4760
NMS	0.1725	0.1472	0.3080	0.2481
AMS	**0.0271**	**0.0236**	**0.0016**	**0.0290**

Table 4.2 compares the predictive error and the KS statistics for the different models in both the synthetic and empirical datasets. The RAS model outperform the NS and the AS models regarding the KS values in most of the data sets. This indicates an improvement of the models with the inclusion of independent dynamics for the service time distribution. For sound evaluation of our models we have also fitted the data with theoretical models *Pareto* (**PAR**), *Lognormal* (**LNO**) and *Gamma* (**GAM**). Such long tail distributions are common stationary distributions for service times in closed form solutions. Note that these theoretical models cannot capture the censored arrivals. The NS models represents the closest representation to these theoretical estimates, due to the gamma form of the distribution. Therefore the NS model is a natural extension of the theoretical ones and serves as a baseline in the predictive error Table 4.2.

Table 4.2 leads us to also make the following observations: (i) *our service time distributions depend on the arrival process.* Indeed, note that conditioning the service time models on the customer RPP process significantly improves their performance in either metric (AS vs ATE; RAS vs RATE). These models successfully use the vector representation $\mathbf{h}_i^a$ encoding the arrival dynamics; (ii) *service systems have their own dynamics.* RAS outperforms the AS model in almost all datasets. This corroborates our assumption that including a model for the service system's dynamic response helps in describing the system's behavior.

In order to provide an intuition for the qualitative behavior of the models we show the inferred distributions in comparison to the empirical and theoretical distributions in Fig. 16. The adversarial solutions capture the complex data distributions while the theoretical estimates fail to recover the short time service time (upper left point in the distributions) as well as providing no solution for censoring events i.e. arrivals without known service.

In Table 4.3 we present the results for the mempool model. The adversarial solution clearly outperforms all other models both in the error as well as in the distribution shape as stated in the KS statistics. Here, we can also include the theoretical models in the prediction error, because there are no censored events in the dataset. We can see the qualitative behavior of the mempool models in Fig. 17, where we compare the AMS model with the empirical distribution and the theoretical models. One can immediately notice the superiority of the adversarial model over the other models, which is the only model that capture the multimodal nature of the mempool datasets distributions.

Fig. 18 shows the comparison of the proposed models among each other for the Github and Mempool Service dataset. Only the adversarial models correctly capture the distribution for the short-term as well as the long-term response of the system, while the neural models fails in both regimes.

4.5 Relations to formal analysis of queuing systems

Being non-parametric, our constructions do not allow a formal theoretical treatment in the spirit of traditional queuing system results (Williams 2016). However, one could employ neural network analysis to obtain estimates of service times, which could eventually be related to formal results in the queuing community. To illustrate, assume the simplified service time model[7] $\tilde{s}_i := \theta^k \circ \sigma^k \dots \sigma^1 \circ \theta^1(\mathbf{h}_i)$, where $\mathbf{h}_i := \text{concat}(\mathbf{h}_i^a, \boldsymbol{\epsilon})$, with $\mathbf{h}_i^a$ defined in (4.2.2) and $\boldsymbol{\epsilon} \in \mathbb{R}^h$ with its elements sampled from P_ϵ as above; $\sigma = \text{ReLU}$ and θ^j labels the weights of the jth layer.

Lemma 1 *Suppose that the simplified model above fits the data within a mean average error of ε. Then the average service time is bounded above as follows*

$$\langle \tilde{s}_i \rangle \leq M^k \limsup_i \|\mathbf{h}_i^a\| + \frac{M^{k+1} - 1}{M - 1} + \varepsilon. \tag{115}$$

Here M is a positive constant bounding the operators norm $\|\theta^j\|$ for all j. This straightforward estimate is proved and related to the average queue length in the Appendix 10.

4.6 Discussion and Outlook

In this chapter, we present a novel deep nonparametric solution for queue systems' service time distributions for general arrivals distributions. Our solutions incorporate censoring of services times and rich estimates for complex distributions and independent dynamical representations for the service systems dynamics. Our methodologies outperform theoretical results and reproduce complex distributions for service times, providing richer representations that can recover multi modal and long tail distributions. We also present a tailored solution for the bitcoins mempool queues systems. Future lines of work include: incorporating richer representations for the arrivals processes. For example, in mobility systems, the geographic information and user interactions can be encoded through hidden networks of relations. Moreover, one could make the system's response explicit by providing new remaining service times with each new arrival.

In the next chapter, we will study once again point process models. Now, however, instead of prediction, we are interested in extracting knowledge from data. We will develop an unsupervised learning algorithm that uncovers regularities from data via a spectral clustering procedure. We also include an outlier detection procedure. As one is interested in sparse point processes, the goal is different. Now the inference will be performed through a spline interpolation approach for the intensity function. This requires fewer parameters than the recurrent neural network and allows for the study of realizations with sparse data.

[7] Note that this simpler model is close to our AS baseline model.

Chapter 5

Temporal Patterns for Point Processes

In this chapter, we return our attention to an unsupervised task. Similar to Switching Dynamical Systems one is possed with the goal of extracting structure from data. Now, however, we provide a solution on the level of the population of time series instead of the local nature of the switching dynamics above. We aim at extracting common behaviors for different time series, here temporal point processes. Hawkes process with Gaussian process excitations and Adversarial services, the previously studied models for point process, aimed at improving the prediction for the next arrivals times or the corresponding service time. In this chapter, is the aggregated behavior which comes to the foreground. Unlike switching dynamical systems, we do not find the different dynamical classes in one time series, but among the timer series. After clustering is performed, the categorical variable will index the time series pertaining to a particular dynamical class. As is common among unsupervised and clustering tasks, there is an inherent subjectivity to what constitutes the right clustering—posted differently, what criteria define the objective function to be optimized? Due to the stochastic nature of point process, we developed a similarity measure that characterizes local translational invariance and scale invariance.

From the point of view of application, we once again focus on data from web services due to their data richness. Understanding dynamics on the user level is thus pivotal for effective site management because knowledge as to typical usage patterns allows for the design of targeted advertisements or improved allocation of resources. Work on analyzing temporal Web user data often employs either pure data-driven approaches or models of behavior (Petrovic et al. 2011; Gao et al. 2015). Our work in this chapter covers a middle ground: we develop an unsupervised algorithm that allows us to uncover patterns of behavior in a population of users, detect deviations from these patterns, and identify outliers within the population.

Moreover, most work on user behavior analysis focuses on aggregated data of whole populations of Web users. Examples include models that seek to understand how collective attention to memes, tags, or news items evolves over time (Dezsö et al. 2006; Wu and Huberman 2007; Yang and Leskovec 2011; Cunha et al. 2011; Lehmann et al. 2012; Weng et al. 2012; Bauckhage et al. 2013; Radinsky et al. 2012). In spite of the relevance of the individual user, the bulk of the studies concerning dynamical patterns concentrates on the aggregate behavior (Dezsö et al. 2006; Wu and Huberman 2007; Yang and Leskovec 2011; Cunha et al. 2011; Lehmann et al. 2012; Weng et al. 2012; Radinsky et al. 2012). Models seek to understand how particular memes, tags or news items evolve over time. Yet, understanding the underlying *dynamic atoms*, i.e. the dynamic behaviors of individual users, presents different challenges. First of all, individual user activities are discrete and sparse. Second of all, individual user activities depend on individual preferences as well as on the design of the site or service and thus are very diverse. When faced with the problem of identifying individual dynamical patterns, analysts thus require flexible as well as scalable methods.

The theory of temporal point processes provides an ideal framework for this setting since individual

user actions can be understood in terms of arrivals, which are governed by an intensity function. Unlike the correlation structure uncovered in the Self Excitation chapter, we want to provide a less specific characterization and allow for fast inference. Different versions of hawkes process have been applied to populations data. (Kobayashi and Lambiotte 2016; Zhao et al. 2015b). However, previous work in this direction requires the (manual) specification of functional representations of the intensity and its dependencies, devoided of a non parametric form that informs the intensity.

Therefore, in this chapter, we aim at a considerably more flexible methodology that does not require manual tweaking to be applicable to a wider range of use cases. To extract descriptive (process) knowledge from given activity data, we propose an algorithm that generalizes the k-spectral centroid algorithm (Yang and Leskovec 2011) in that it involves kernel operations which can cope with the inherent randomness of the intensities of a Poisson process. To cope with the large variety of possible user behaviors, we develop a spline interpolation model that can be substituted for the intensity function of a Poisson process. This way, our approach provides a scalable and flexible approach towards learning intensities and circumvents the need to have to specify functional forms or dependencies. In summary, our technical contributions in this chapter are as fourfold:

- *Poisson process intensity inference* we introduce a scalable and flexible methodology for handling temporal Poisson process intensities based on spline interpolation.
- *Similarity measure for time series* we propose a way of measuring similarities among time series that is invariant to translations of local patterns.
- *Time series clustering* given our similarity measure, we extend the K-SC algorithm towards a flexible, piecewise variant that allows for efficient computation and thus scales to a large amount of data.
- *Outlier detection for point processes* given cluster centers determined by our new algorithm, we propose an entropy measure to determine how well a cluster prototype represents a certain (intensity) time series; this allows for the detection of outliers within a sample of temporal patterns.

In practical experiments, we apply our framework to three large datasets. In particular, we analyze question answering behaviors on StackOverflow, transactions within the BitCoin network, and push events on Github. Our data comprises observations of 2.6, 24, and 1 million individual activities, respectively. Our results indicate that the proposed framework can uncover diverse yet interpretable prototypic behavior patterns.

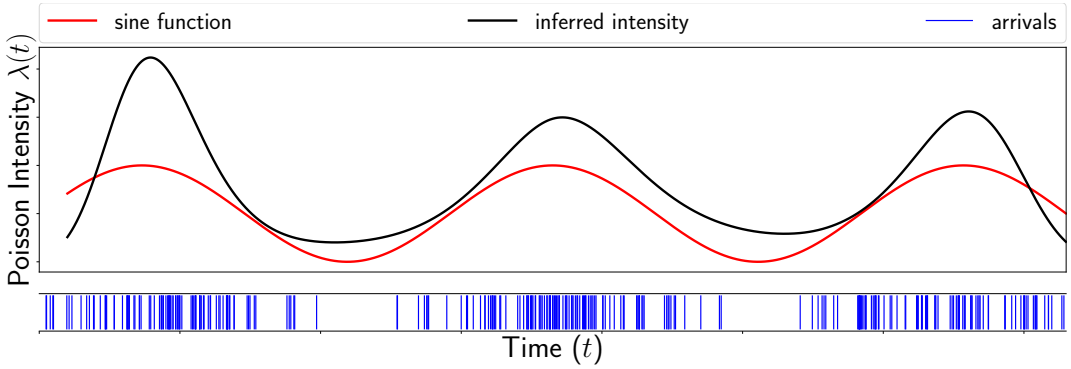


Figure 19 – Example of an inhomogeneous Poisson process with intensity function $x(t) = sin(t)$ (red) and inferred intensity function obtained from running our splines procedure (black).

Our goal is to uncover prototypic dynamics from sets of discrete time series. Each of the time series we are concerned with reflects the dynamical behavior of an individual user or agent with respect to a Web service and is thus comprised of sets of arrivals. In this section, we first review the basics of the Poisson process modeling for our purpose and discuss how to do parameter inference for such models. We then introduce our clustering algorithm as well as our procedure for outlier detection.

5.0.1 Problem Definition

Throughout, we assume that an *arrival* is a point in time when a user or agent interacts with a Web service. Examples of possible actions include a click on a link or the playing of an online video but also the posting of an answer on a StackOverflow site, a transaction from a given Bitcoin address to another, or a push event in a Github repository.

We also assume that we are given N users with arrivals a_i^u, where $u \in 1, \ldots, N$ indexes the user and $i \in 1, \ldots, M_u$ indexes the observed arrivals for each user.

For each a^u we obtain the corresponding intensity function $\lambda^u(t)$ using the splines interpolation as explained below. We define $\lambda_i = \lambda(a_i)$. The intensity function is formally defined as in Chapter 2.1.1. Here it is important to note that learned intensity functions encode the dynamical behavior of individual users. Given these, our goal is to be able to cluster intensity functions in such a way that functions in the same cluster represent agents with similar dynamical patterns. The number of desired clusters K is given a priori. Once a clustering has been computed, clusters centers represent a notion of *normal behavior* and outlier time series are detected based on their dissimilarity to the cluster centers.

5.0.2 Fast Intensity Inference Using Splines

Since our goal is to characterize many patterns of behavior, we need an efficient method to represent the general shape of instensities as opposed to a detailed description of functional form or time dependencies. Since a fast and very flexible method is to use spline interpolation, we adapt the procedure in (Ogata and Katsura 1988) to temporal processes.

For a given set of a user's arrivals $A_u = \{a_i^u\}$, we maximize the likelihood $L(A_u \mid \theta) = P\big(a_1^u, \ldots, a_{M_u}^u \mid \lambda(t;\theta)\big)$. Its log-likelihood is given by

$$\log L(A_u \mid \theta) = \sum_i \log a_i^u - \int_0^T \lambda(t;\theta)\, dt \tag{116}$$

where the support is defined over $[0, T]$. We approximate $\lambda(t;\theta) = \sum_j \theta_j F_j(x)$ in terms of a spline

expansion.

Due to the large number of parameters of the spline model, a direct maximization of $L(A_u|\,\theta)$ will likely entail over-fitting which, in turn, would induce rapid local changes in the intensity function $\lambda(t)$. We therefore regularize the likelihood and introduce *roughness penalties* defined via functionals

$$\Phi_1(\lambda) = \int \left(\frac{d}{dt}\,\lambda(t')\right)^2 dt' \tag{117}$$

and

$$\Phi_2(\lambda) = \int \left(\frac{d^2}{dt^2}\,\lambda(t')\right)^2 dt' \tag{118}$$

so that the penalized log-likelihood becomes

$$L_p(A_u \mid \theta) = L(A_u \mid \theta) - \big\{w_1\Phi_1(\lambda) + w_2\Phi_2(\lambda)\big\}. \tag{119}$$

The new hyperparameters w_1 and w_2 allow for weigthing the different contributions of the roughness terms and can be obtained through type II maximum likehood techniques. To maximize the penalized likehood in (119), we resort to the L-BFHS-B algorithm for large-scale bound-constrained optimization (Zhu et al. 1997). A didactic example as to the kind of results we obtain in this way is shown in Fig. 19.

5.0.3 A Dynamic Piecewise Time Series Similarity Measure

To be able to cluster spline representations of the intensity functions of arrivals of individual users into groups refelcting similar behavior, we need a suitable notion of similarity.

In order to illustrate the requirements a suitable similarity measure should fulfill in our context, Fig. 20 shows examples of Poisson process intensities that were learned from real data. From these plots, we recognize that different peaks of intensities may differ in shape, size, and location.

Since we are interested in uncovering patterns of user behavior, we require our similarity measure to consider two time series as similar when they have similar shapes, irrespective of their difference in size.

Furthermore, our application context demands *local translation invariance* for different time scales. This means that we would like to differentiate among users who work in a particular month, week, or number of days. The time scale which we are able to differentiate should depend on parameters given as input to the algorithm. For example, if we are interested in locating users who work during particular seasons, say summer and winter, users who only differ by a few weeks should be considered similar. In other words, *a user which has the same workload in the first week of June and the last week of November should be similar to a user which also has the same workload, but in the last week of June and the first week of November.*

This requirement also arises due to the nature of our spline algorithm. Our procedure for inferring the intensities is sensitive to the sparsity of data.

To address these problems and requirements, we therefore propose a similarity measure which is able to identify intensities located at different but close minima as similar.

Consider the two time series, i.e. learned intensity functions, $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^D$ and $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^D$, where D is the dimensionality of the series. We divide them into $0 < R \leq D$ equal parts and obtain $\boldsymbol{p} = \{\boldsymbol{p}_s\}_{s=1}^R$ and $\boldsymbol{q} = \{\boldsymbol{q}_t\}_{t=1}^R$.

These different sections or pieces of the series allow us to handle similarities depending on the time scale of our choice. In order to allow a similarity among different sizes, we introduce a rescaling

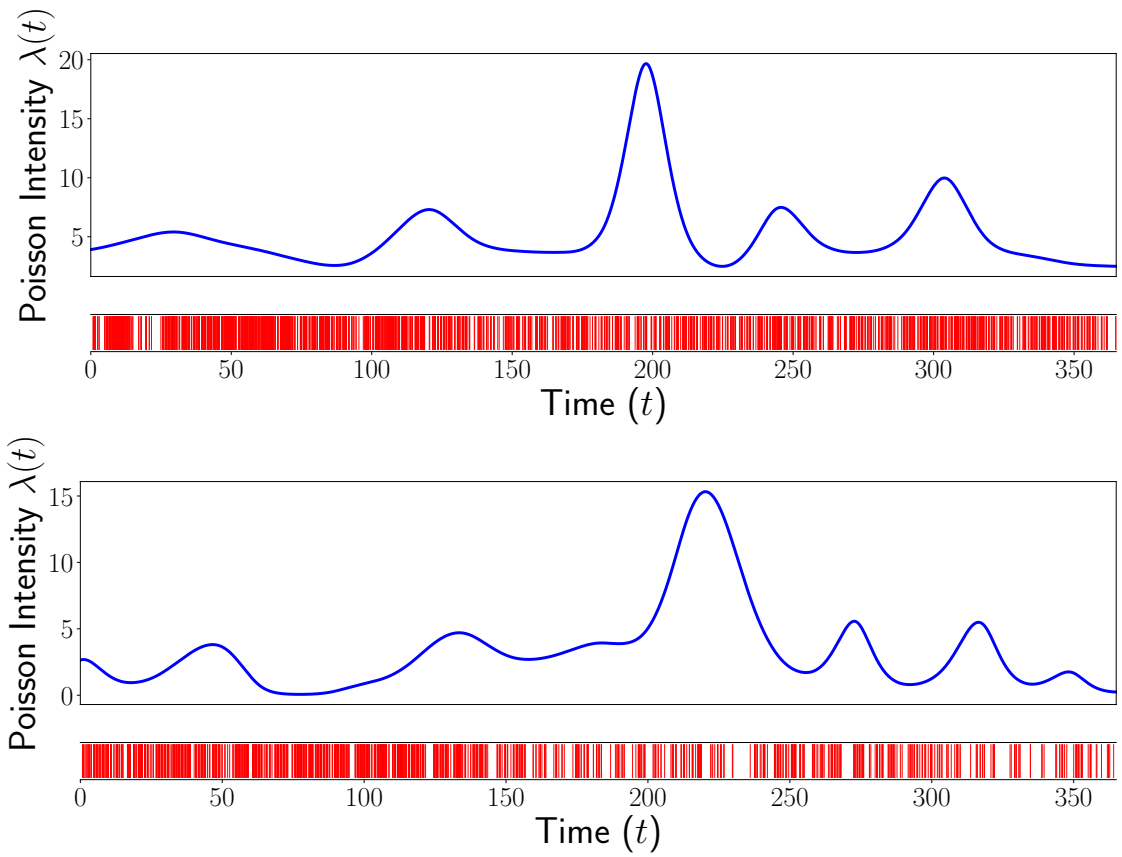


Figure 20 – Examples of intensities inferred from users arrivals, which, in spite of variations in the position of some peaks, should be recognized as similar due to our requirements.

between any two pieces of the time series $\tilde{d}_\alpha$ with α as a scaling factor

$$\tilde{d}_\alpha\left(\boldsymbol{p}_s, \boldsymbol{q}_t\right) = \frac{\|\boldsymbol{p}_s - \alpha \boldsymbol{q}_t\|^2}{\|\boldsymbol{p}\|^2} \tag{120}$$

To handle similarities among different pieces, we introduce a temporal kernel $k_{[0,1]}$ which should weight every piecewise contribution among different sections of the series and its particular choice depends on the application at hand. Finally, we aggregate our metrics and define the Dynamic Piecewise Time similarity measure

$$d^2\left(\boldsymbol{p}, \boldsymbol{q}\right) = \frac{1}{2R} \min_\alpha \sum_{s=1}^{R} \sum_{t=1}^{R} k_{[0,1]}\left(s,t\right)\, \tilde{d}_\alpha\left(\boldsymbol{p}_s, \boldsymbol{q}_t\right) \tag{121}$$

Note that the role of the kernel is to weight different pairs $\boldsymbol{p}_s, \boldsymbol{q}_t$ according to the sectioning times s, t. In our case, if we want to uncover common user behavior patterns, we would like closer times to correspond to a more important similarity contribution. In order to reproduce this behavior, we use the common RBF kernel

$$k_{[0,1]}\left(s,t\right) = c^2 \exp\left(-\frac{\left(s-t\right)^2}{2l^2}\right) \tag{122}$$

Another possible choice for a temporal kernel could be the uniform kernel $k_{[0,1]}\left(s,t\right) = \mathbb{I}_{\{|s-t|<\delta\}}$. This kernel compares every piece from the first time series with pieces of the second that are close in time and the parameter δ defines how close two pieces have to be.

As we need to perform similarity calculations many times when running the algorithm, computational efficiency is key. We therefore rewrite the double sums in (121) in terms of matrix products. That

Algorithm 4: K-PSC$(\boldsymbol{p}, K, R, \sigma)$ clustering

Data: time series $\boldsymbol{p}_i$, $i = 1, \ldots, N$, number of clusters K, number of pieces R, length scale of an RBF kernel σ
Initial cluster assignments $C = \{C_1, \ldots, C_K\}$
$G = \text{ceil}\left(\frac{D}{R}\right)$
$\boldsymbol{K} \leftarrow k_{[0,1]}(s,t)$ where $s, t = 1, \ldots, R$
$\tilde{\boldsymbol{K}} \leftarrow \boldsymbol{K} \otimes \boldsymbol{I}_G$
$\boldsymbol{L} \leftarrow \text{diag}\left[\underbrace{k(0,0), \ldots, k(0,0)}_{G}, \ldots, \underbrace{k(R,R), \ldots, k(R,R)}_{G}\right]$
repeat
  $\tilde{C} \leftarrow C$
  for $j = 1$ **to** K **do**
    $\boldsymbol{M} \leftarrow \sum_{\boldsymbol{p}_i \in C_k} \left(\boldsymbol{L} - 2\boldsymbol{A}_i^T \tilde{\boldsymbol{K}} + \boldsymbol{A}_i^T \boldsymbol{L} \boldsymbol{A}_i\right)$
    $\boldsymbol{\mu}_j \leftarrow$ The smallest eigenvector of $\boldsymbol{M}$
    $C_j \leftarrow \emptyset$
  end
  for $j = 1$ **to** N **do**
    $j^* \leftarrow \underset{j=1,\ldots,K}{\text{argmin}}\ d^2(\boldsymbol{p}_i, \boldsymbol{\mu}_j)$
    $C_{j^*} \leftarrow C_{j^*} \cup \{i\}$
  end
until $\tilde{C} = C$
return $C, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$

is, we let let $G = \text{ceil}(\frac{D}{R})$ be the number of points in the different sections of $\boldsymbol{p}$ (or $\boldsymbol{q}$) of the time series $\boldsymbol{\lambda}$ (or $\boldsymbol{\lambda'}$) and introduce

$$\boldsymbol{K} = \begin{bmatrix} k(0,0) & \cdots & k(0,R) \\ \vdots & \ddots & \vdots \\ k(R,0) & \cdots & k(R,R) \end{bmatrix}.$$

We then consider the Kronecker product $\tilde{\boldsymbol{K}} = \boldsymbol{K} \otimes \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix of size $G \times G$, and the diagonal matrix

$$\boldsymbol{L} = \text{diag}\left[\underbrace{k(0,0), \ldots, k(0,0)}_{\text{G}}, \ldots, \underbrace{k(G,G), \ldots, k(G,G)}_{\text{G}}\right].$$

To minimize $d^2(\boldsymbol{p}, \boldsymbol{q})$ w.r.t. α, we solve $\frac{\partial}{\partial \alpha} d^2(\boldsymbol{p}, \boldsymbol{q}) = 0$ and obtain

$$\alpha = \frac{\boldsymbol{p}^T \tilde{\boldsymbol{K}} \boldsymbol{q}}{\boldsymbol{q}^T \boldsymbol{L}\, \boldsymbol{q}} \tag{123}$$

which then allows us to rewrite the distance measure in (121) simply as

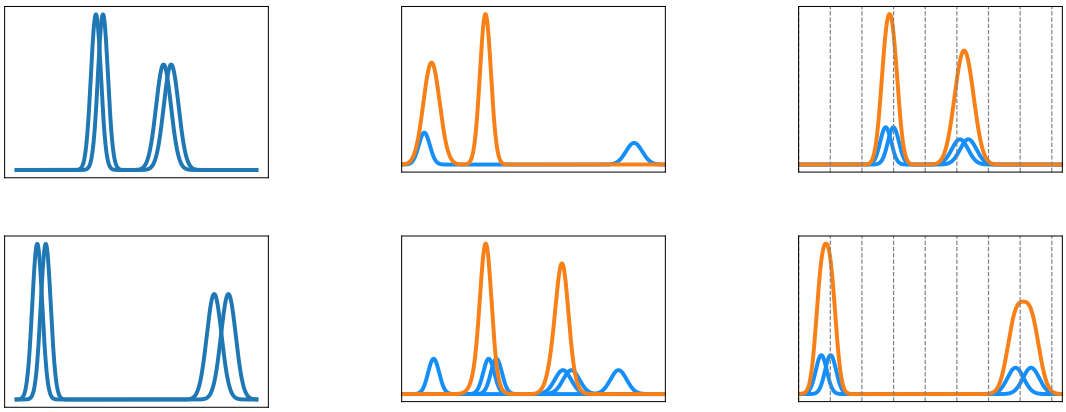
$$d^2(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{2R\|\boldsymbol{p}\|^2} \left(\boldsymbol{p}^T \boldsymbol{L}\, \boldsymbol{p} - 2\alpha \boldsymbol{p}^T \tilde{\boldsymbol{K}} \boldsymbol{q} + \alpha^2 \boldsymbol{q}^T \boldsymbol{L}\, \boldsymbol{q}\right). \tag{124}$$

5.0.4 A K-Piece Wise Spectral Centroid Algorithm

Next, we describe our algorithm for clustering according to the above similarity measure; for brevity, we will refer to it as the K-Piece Wise Spectral Centroid or K-PSC.

K-PSC is an iterative algorithm similar to the k-means or the k-SC algorithm (Yang and Leskovec 2011), but it is specifically tailored towards efficiently finding clusters under the DPT similarity measure in (121). Similar to the conventional k-means algorithm, it iterates over two steps, assignment and refinement. In the assignment step, time series of activity rates are assigned to the closest cluster based. In the refinement step, clusters centroids are recalculated (see Alg.5).

With respect to the refinement step, we observe the following: we are given a data set of time series $\boldsymbol{p}_i$, a number of clusters K, the number of pieces R for time series subdivision, as well as kernel parameters. Defining clusters C_k in terms of cluster prototypes or centroids $\boldsymbol{\mu}_k$, the goal is



(a) (a) time series **(b)** (b) K-SC centroids **(c)** (c) K-PSC centroids.

Figure 21 – Four time series and cluster centroids obtained from the K-SC and K-PSC algorithm. Our K-PSC methodology is invariant to local translation.

to minimize

$$F = \sum_{k=1}^{K} \sum_{\boldsymbol{p}_i \in C_k} d^2\left(\boldsymbol{\mu}_k, \boldsymbol{p}_i\right). \tag{125}$$

To determine suitable centroids $\boldsymbol{\mu}_k$, we thus need to solve

$$\boldsymbol{\mu}_k^* = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{\boldsymbol{p}_i \in C_k} d^2\left(\mu, \boldsymbol{p}_i\right). \tag{126}$$

which, after some algebra, is equivalent to

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \boldsymbol{\mu}^T \frac{\boldsymbol{M}}{2R\|\boldsymbol{\mu}\|^2} \boldsymbol{\mu}, \tag{127}$$

where

$$\boldsymbol{M} = \sum_{\boldsymbol{p}_i \in C_k} \left(\boldsymbol{L} - 2\boldsymbol{A}_i^T \tilde{\boldsymbol{K}} + \boldsymbol{A}_i^T \boldsymbol{L} \boldsymbol{A}_i \right) \tag{128}$$

and

$$\boldsymbol{A}_i = (\boldsymbol{\lambda}_{\boldsymbol{i}}^T \boldsymbol{L} \boldsymbol{\lambda}_{\boldsymbol{i}})^{-1} (\tilde{K} \boldsymbol{p}_i^T \otimes \boldsymbol{p}_i). \tag{129}$$

The solution $\boldsymbol{\mu}_k^*$ to the minimization problem in (127) then is the eigenvector $\boldsymbol{u}_m$ corresponding to the smallest eigenvalue ψ_m of matrix $\boldsymbol{M}$.

Note that the well known k-means algorithm updates centroids by averaging over the data in the corresponding cluster. However, since our similarity measure is scale invariant, not all elements of a cluster contribute equally to the cluster averages in our scenario. Our algorithm therefore needs to take scales into account. and this is what (128) accomplishes.

Figure 21 compares results of our K-PSC algorithm to those of the original K-SC algorithm. Similar to our methodology K-SC rescales the time series but it also shifts the positions of their peaks. This creates a rather undesirable clustering for our application as seen in the merging of three intensities in the bottom of the second column.

5.0.5 Outlier Detection

In order to identify patterns of user or agent behavior that deviate from normal behaviors, i.e. whose intensity functions differ considerably from that of the closest cluster centroid $\boldsymbol{\mu}_k$, we follow the formalism in (Benkabou et al. 2016). To this end, we extend the objective in (125) by entropy term,

namely

$$F_o = \sum_{k=1}^{K} \sum_{\boldsymbol{p}_i \in C_k} w_i \, d^2 \left(\boldsymbol{\mu}_k, \boldsymbol{p}_i \right) + \phi \sum_i w_i \log w_i, \tag{130}$$

Here, the weights w_i define how much a given time series belongs to a cluster. Minimizing F_o can be accomplished using the same assignment and refinement steps as above. The weights w_i, however, can be updated independently and when minimizing (130), we find

$$w_i = \frac{\exp\{-D_i/\phi\}}{\sum_j \exp\{-D_j/\phi\}} \tag{131}$$

where $D_i = d^2 \left(\boldsymbol{\mu}_k, \boldsymbol{p}_i \right)$ if $x_i \in C_k$.

This procedure is akin to a maximum entropy principle: we must maximize the amount of uncertainty encoded in the assignments w_i so as to obtain an unbiased definition of the distribution given by w_i (Jaynes 1957). The parameter ϕ enters as an inverse Lagrange multiplier which fixes the information which is known, namely the distance to the cluster centers. If the time series is in effect and outlier, it will be poorly represented in the centroids and, as a consequence, the weight w_i will be small.

5.0.6 Scalability

Having discussed all components of our framework, we next analyze its overall complexity.

With repsect to the spline interpolation for the intensity of the arrivals, we note that the intensity function $x^u(t)$ for a user u is defined over $[0, T]$ and requires spline basis function over S interpolation points. Maximizing the penalized likelihood L_p in (119) is accomplished using L-BFHS-B (Zhu et al. 1997) which runs in $O(SQ)$ where Q is the number of quadrature points required for numeric integration. (The derivatives in the regularization term can be performed analytically.) Alltogether, accounting for the number M_u of arrivals per user, the full calculation of each step takes $O(M_u + SQ)$. In comparison, competing models with the same degree of flexibility but based on Gaussian processes (Samo and Roberts 2015b) require efforts of $O(M_u \, P^2)$ where P is a typically large number of inducing points.

With respect to the problem of clustering the learned intensity functions $x^u(t)$ of N users into K clusters, we observe the following. If the length of the time series is L, the assignment step of our clustering algorithm requires $O(NKL)$ runtime. The refinement steps requires the computation of the $\boldsymbol{M}$ in (128) and its eigenvectors. In contrast to the original the K-SC algorithm, our matrix $\boldsymbol{M}$ requires the calculation of the kernel matrix $\tilde{\boldsymbol{K}}$ and thus $O(NL^3)$ complexity. This translates into a final complexity of $O(\max(NL^3, KL^3))$. Improvements are achieved via approaches similar to incremental k-means thorugh the discrete Haar Wavelet transform (Yang and Leskovec 2011).

5.0.7 Results on StackOverflow Data

Next, we describe temporal patterns obtained from intensity functions inferred for the StackOverflow data.

For inference of the intensities we used 12 splines knots so that different peaks occurring within a time window of two moths will be deemed similar. Figure 24 shows prototypic intensities or user activity time series we obtained from K-PSC clustering with $K = 18$. Despite a high degree of variability in the behavior of individual users, we can recognize several tendencies. The clusters in the first row of Fig. 24 correspond to rather peaked behavior where the width of the peaks varies between about 2 to 3 months. Since an active profile on StackOverflow is believed to boost careers opportunities in the tech industry, peaked behaviors like these might indicate users who seek to gain reputation quickly so as to land a job. The second row of Fig. 24 shows that there

are StackOverflow users whose activities either decline or grow over time. We also observe several peaks with different volumes and outliers to such patterns represent users with less clear patterns of growth or decline. The clusters in the third row of the figure are indicative of user behaviors with patterns of seasonality. Except for the clusters in Figs 24m and 24n, we see patterns of high volume work at the beginning and the end of the year but with limited activity during the summer.

Overall, peaks of prototypic time series determined for this data show peaks located at rather complementary points in time. If they were aggregated, the different peaks would cover the whole period studied here. Our results thus indicate a rather interesting behavior as it seems that prototypic answering behaviors are characterized by peak activity periods. From the point of view balancing the work on StackOverflow, our results therefore suggest that rather than expecting individual users to work more in low activity periods, more users would need to be enticed to work in these periods at all. Dynamical awarded scores rewarding users for participating at particular times instead of others could potentially achieve this goal.

5.0.8 Results on BitCoin Data

Our BitCoin data set is richer and of different nature than the StackOverflow data because here arrivals indicate points of activities addresses rather than users. In other words, although such addresses reflect individual entities, these entities need not be individual humans but could also represent banks or gambling sites. In this sence, the behavior contained in the BitCoin data does not exclusively reflect human behavior.

Again, we used splines with 12 knots for intensity inference. For the selection of kernel parameters, we chose the model with lowest clustering entropy. This was obtained for $R = 6$ different sections and convolution kernel parameters of $b = 0.1$ and $l = 10^{-5}$. We show our results in Fig. 25. As before we have a wide range of dynamical natures. In the first row of the figure, we observe peaks of activity of around two months. As opposed to the StackOverflow data, outliers identified in the BitCoin data differ more clearly from the corresponding cluster centers. In particular, we observe replicas (other peaks) as well as substantially varying widths of peaks. In the second row of Fig. 25, we observe activities covering most of the observation period of 6 months but with sudden drops of activities. In the last row, except for Fig. 25m, there are intensity prototypes which describe activities that increase noticeably towards the end of the observation period. This is well in line with the development of the value of a BitCoin over the course of the year 2017. In other words, it seems as if the surge in value led to an increase of trading as users tried to capitalize.

5.0.9 Results on Github Data

Next, we present results for the Github data set. This dataset has a different granularity as we only studied arrivals in a period of a month.

We selected 8 knots for spline interpolation so that the polynomials can capture 2 peaks per week. The entropy criterion yielded similar parameters as above and we partitioned the time series into $R = 7$ parts and used convolution kernel parameters of $b = 10^{-5}$ and $l = 0.001$. In Fig. 26, we selected prototypic examples of six out of $K = 18$ behavior clusters.

Figures 26a, 26b, and 26c show highly active users who work either at the beginning or the end of the month or over about three quarters of the observation period. Outliers to these prototypes represent dips the plateaus or abrupt decays of activity before the end of the month. Figures 26d, 26e, and 26f, on the other hand, show patterns of periodic activity where users commit their work at the beginning and at the end of the month. Here, outliers typically represent a phase difference in these periods.

In this section, we evaluate our approach in several large, real world data sets. We first introduce

Figure 22 – Cluster quality (in terms of entropy and silhouette coefficient) for different clusterings of the StackOverflow data.

these data sets and then present and discuss our results.

5.0.10 Datasets

To develop an understanding of temporal patterns in the individual behavior of users of Web sites or services, we gathered data sets from three application domains.

StackOverflow is a questions answering platform for programmers and technology professionals. Here, we gathered data about 2,500 users and 2,600,000 answers they posted between January and June 2017. For this data set, we consider an arrival to be the point in time when a user answered a question.

BitCoin is a cryptocurrency and payment system based on the decentralized blockchain technology. Here, we collected data as to 600,000 addresses within the Bitcoin network and the 24,000,000 transactions these were involved in between January and June 2017. For this data set, an arrival is defined as the point in time when a particular address sent a Bitcoin.

Github is a popular version control repository and Internet hosting service. Here, we crawled data about 110,000 different users who initiated about 1,000,000 push events during January 2017. Correspondingly, for this data, an arrival is understood to be the point in time where a push event occurred.

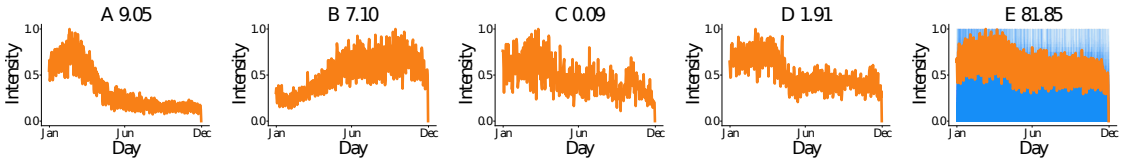
5.0.11 Experimental Setup

For each of our three data sets, we performed intensity inference via spline interpolation as discussed in section 5.0.2.

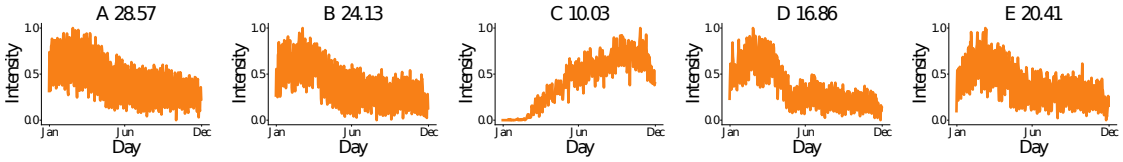
We maximize the likehood until the Akaike information criteria (Akaike 1998) exceed a threshold and the average of the number of arrivals given by

$$m(t) = \int_0^t \lambda(\tau)\, d\tau \tag{132}$$

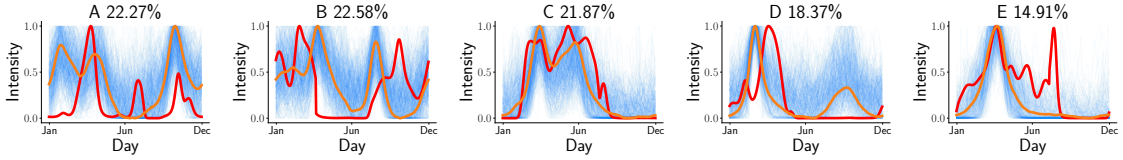
lies within less then a 0.1% difference of the empirical value. The number of knots for the spline basis functions were selected such that there was at least one maximum per polynomial per week.



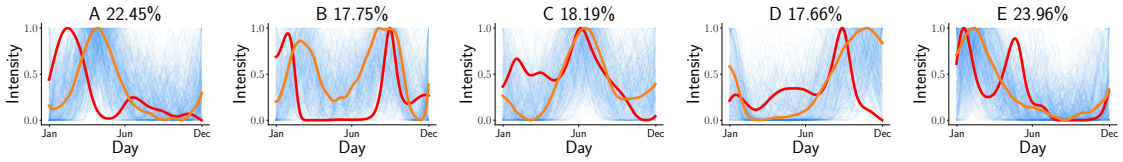
(a) (a) clusters of common user behavior obtained using K-Means and the L_2 norm



(b) (b) clusters of common user behavior obtained using K-Means and dynamic time warping



(c) (c) clusters of common user behavior obtained using K-SC



(d) (d) clusters of common user behavior obtained using the K-PSC algorithm proposed in this chapter

Figure 23 – Five clusters of common behaviors of StackOverflow users as determined by K-means, K-SC, and K-PSC. The bold orange line in each panel represents the cluster centroid, the bold red line represents the outlier with the smallest value of w_i, and the transparent blue lines represent intensities of users assigned to the corresponding cluster. Percentages at the top of each panel indicate the size of the cluster relative to the total number of users of the site.

With respect to kernel parameter selection, we choose the model with lower entropy, i.e. the model that led to cluster centers which yield the biggest information encoding gain for the system behavior. We also selected the number of clusters to be determined based in the entropy criterion. As an example, Fig. 22 plots cluster quality measures against the number of clusters for the StackOverflow data.

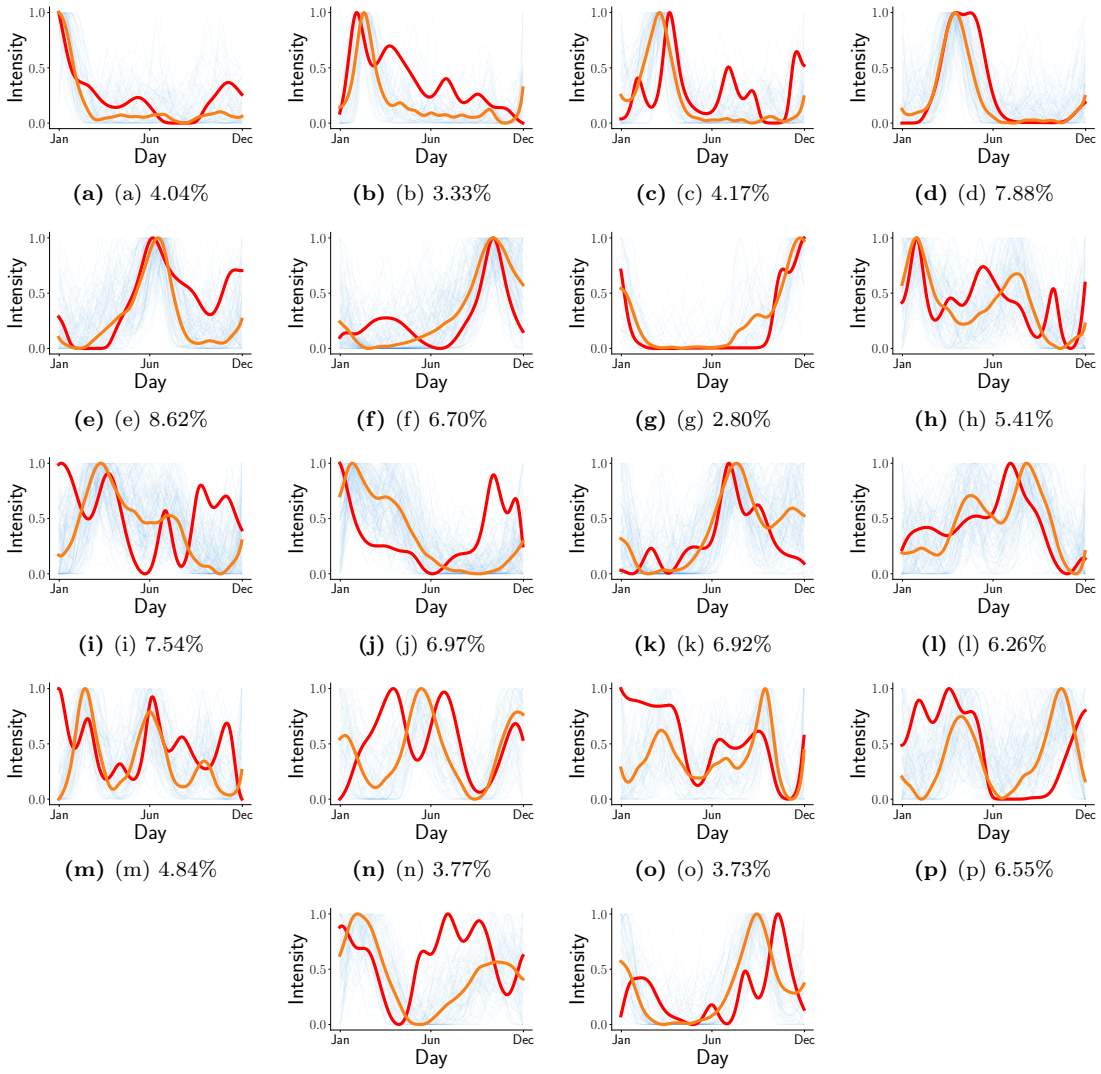
Once clustering had been performed as discussed in section 5.0.4, we determined outliers according to the approach in section 5.0.5. That is, we determined the weights w_i as prescribed by (130) and defined outliers to be those intensities whose w_i had values in the bottom 0.1% of all weights.

5.0.12 K-PSC versus K-SC or K-Means

First, we compare the results of our K-PSC algorithm against two baselie procedures, namely K-SC and K-means.

We considered the StackOverflow data set and determined $K = 5$ clusters. Fig. 23 shows results for K-means with Euclidean distance and dynamic time warping distances (Müller 2007) as well as for the K-SC algorithm and for our K-PSC approach. Both K-means variants were computed from vectors indicating numbers of arrivals per day, i.e. without any intensity learning via spline interpolation.

The prototypes resulting from K-means (Figs. 23a and 23b) are not clearly distinguishable and thus do not yield easily interpretable results. This is likely, because individual user activity data are sparse and too irregular for for a mode seeking algorithm to succeed. Yet, problems due to



(a) (a) 4.04% **(b)** (b) 3.33% **(c)** (c) 4.17% **(d)** (d) 7.88%

(e) (e) 8.62% **(f)** (f) 6.70% **(g)** (g) 2.80% **(h)** (h) 5.41%

(i) (i) 7.54% **(j)** (j) 6.97% **(k)** (k) 6.92% **(l)** (l) 6.26%

(m) (m) 4.84% **(n)** (n) 3.77% **(o)** (o) 3.73% **(p)** (p) 6.55%

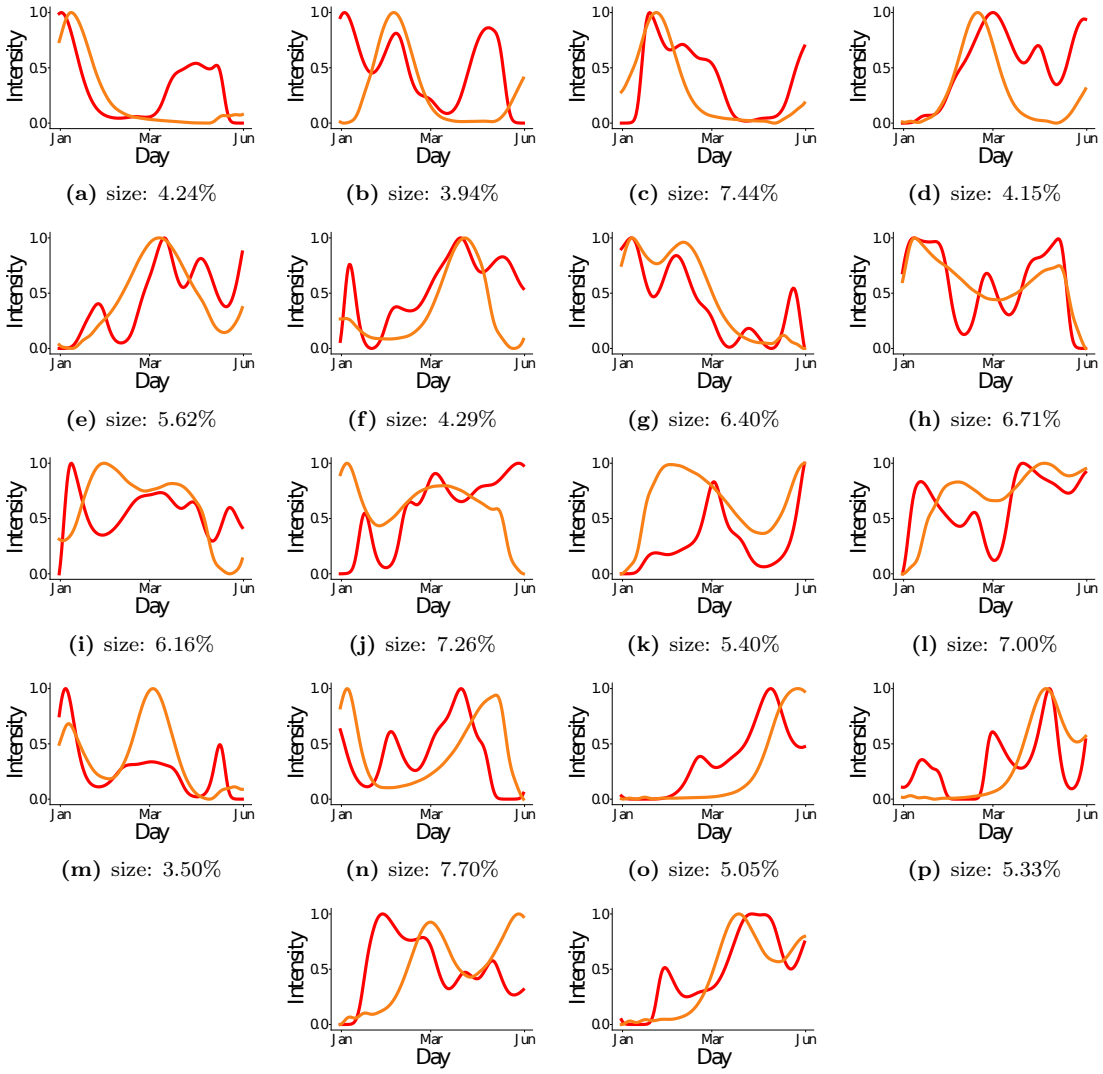
(q) (q) 6.58% **(r)** (r) 3.90%

Figure 24 – Eighteen clusters representing **common behaviors of StackOverflow users**. The bold orange line in each panel represents the cluster centroid, the bold red line represents the outlier with the smallest value of w_i, and the transparent blue lines represent intensities of users assigned to the corresponding cluster. Percentages at the top of each panel indicate the size of the cluster relative to the total number of users of the site.

sparseness can be avoided using our spline approximation of arrival intensity functions.

One of the key advantages of our methodology is that the model allows us to extract information about dynamical patterns not merely from arrivals but from inferred distributions of the numbers of events. While this is beneficial for K-SC and K-PSC alike, we observe that prototypes found by the original K-SC procedure tend to show several modes of activity while most K-PSC prototypes show just one mode. This is because K-PSC scales every time-series differently in order to find cluster centroids. As this decreases the influence of outliers, it leads to more clearly distinguishable centroids. In other words, compared to K-SC, our K-PSC approach can identify patterns w.r.t. their location in time.

In Tab. 5.1, we list clustering qualities according to two measures, namely entropy as defined in (130) and silhouette scores. Higher silhouette scores and low entropies are indicative of good clustering. According to these measures, our model outperforms the others and yields better cluster centroids both in terms of closeness to the centroids (silhouette) and encoding of the dynamical information (entropy).



(a) size: 4.24% **(b)** size: 3.94% **(c)** size: 7.44% **(d)** size: 4.15%

(e) size: 5.62% **(f)** size: 4.29% **(g)** size: 6.40% **(h)** size: 6.71%

(i) size: 6.16% **(j)** size: 7.26% **(k)** size: 5.40% **(l)** size: 7.00%

(m) size: 3.50% **(n)** size: 7.70% **(o)** size: 5.05% **(p)** size: 5.33%

(q) size: 4.45% **(r)** size: 5.37%

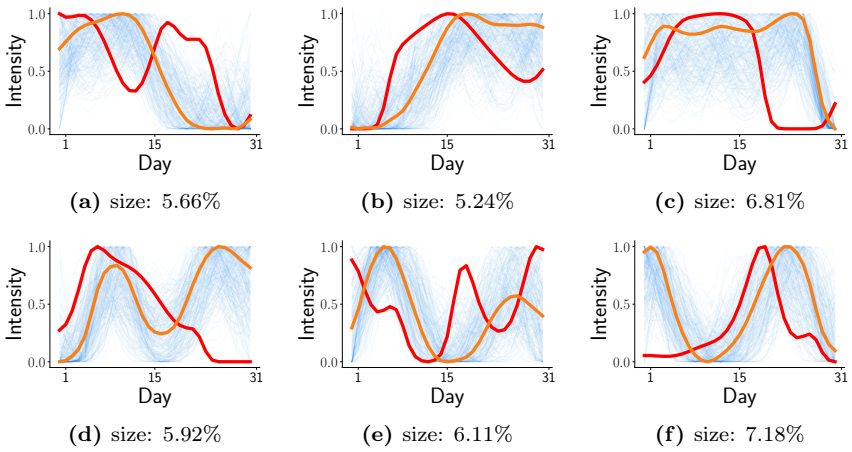
Figure 25 – Eighteen clusters representing **common transaction patterns of BitCoin addresses**. The bold orange line in each panel represents the cluster centroid, the bold red line represents the outlier with the smallest value of w_i, and the transparent blue lines represent intensities of transactions assigned to the corresponding cluster. Percentages at the top of each panel indicate the size of the cluster relative to the total number of transactions.

Table 5.1 – Cluster quality measures.

	5 Clusters		20 Clusters	
	Silhouette	Entropy	Silhouette	Entropy
K-means L_2	0.0881	8.9	0.0777	8.6
K-means DTW	0.0951	8.2	-0.0820	7.9
K-SC	0.0604	7.2	0.0472	7.1
K-PSC	**0.1029**	**7.1**	**0.0873**	**7.0**

Differing from our spline-based approach towards characterizing Poisson intensities for individual Web user behavior analysis, previous related work often focused on behavior analysis on the population level (Dezsö et al. 2006; Wu and Huberman 2007; Yang and Leskovec 2011; Cunha et al. 2011; Lehmann et al. 2012; Weng et al. 2012; Bauckhage et al. 2013; Radinsky et al. 2012). Moreover, the focus of works like these was mainly on specific functional forms that would allow for describing the dynamics of growth and decline of activities commonly observed in Web user data.

In the work reported here, however, we are interested in distinguish among *empirical* functional



(a) size: 5.66% **(b)** size: 5.24% **(c)** size: 6.81%

(d) size: 5.92% **(e)** size: 6.11% **(f)** size: 7.18%

Figure 26 – Six out of eighteen clusters representing **common behaviors of GitHub users**. The bold orange line in each panel represents the cluster centroid, the bold red line represents the outlier with the smallest value of w_i, and the transparent blue lines represent intensities of users assigned to the corresponding cluster. Percentages at the top of each panel indicate the size of the cluster relative to the total number of users of the site.

forms. Previous related work considered inter event times and found power law distributions arising from bursty behavior for email as well as for telephone communication (Barabasi 2005). The presence of a double Poisson process accounting for different time scales of user behavior has also been proposed in (Malmgren et al. 2008). More recently, similar approaches were used to characterize the intensity function for retweets, citations and Hastags dynamics (Gao et al. 2015; Kobayashi and Lambiotte 2016). These approaches are based on reinforced Poisson processes or Hawks processes and characterizations of rate functions are achieved through competing dynamics of loss of attention as well as *richer get richer* mechanisms.

The perspective of the individual user, however, has largely been ignored in analyses such as those above. This is likely due to the natural sparsity of individual user data as well as to the lack of flexible methods for incorporating variability into individual rate functions. Both these issues make it more difficult to specify functional forms for individual users than for a population as a whole. Here, however, we tackled both these problems trough our use of spline interpolation for Poisson process modeling.

5.0.13 Discussion and Outlook

In this chapter, we presented a framework for Poisson point process clustering and outlier detection. It allows us to study the dynamical behavior of individual users of Web sites or services, a setting where observed activities are naturally sparse, such as answers on StackOverflow or commits on Github.

In more specific terms, we were concerned with what kind of dynamical classes of individual user behavior can be identified. In practical experiments, we applied our approach to two large data sets crawled from StackOverflow and Github and a substantial sample of BitCoin transaction data. Our methodology uncovered a wide range of prototypic patterns ranging from bursty and seasonal activities, over increasing and decreasing activities with replicas, to uniform activities.

Contrary to previously studied methods using predefined functional forms for the Poisson intensities, our approach provides more flexibility and is, therefore, better able to cope with the considerable diversity of individual users' behavior.

To cluster the resulting time series data, we proposed a similarity measure that allows for variations in the location and width of peaks. The K-Piecewise Spectral Centroid (K-PSC) algorithm we

proposed for time series clustering is independent of the particular method used to identify a Poisson process's intensities. Moreover, when formulated in terms of tensor- and matrix operations, it allows for the use of special-purpose linear algebra packages for highly efficient computation. In an evaluation against baseline methods, we also found it to yield more concise results.

Unlike previous chapters, we do not explicitly include self excitations or clustering in the modeling of the point processes. Ona could perform inference of the process using the methodologies in Chapter 1, and use the clustering algorithm in the obtained excitation function. Depending on the memory kernel's behavior, a direct application of the splines algorithm to self-exciting data might leads to a noise detection of and intensity associated with inhomogeneous processes.

In the next part of the dissertation, we will work with representation learning instead of inference methods for stochastic processes. The dynamics will be studied not in itself but in how the representation structures affect them. In the last chapter, we will design representations constrained on properties of stochastic processes defined with them, the application case, will be that of black box explanation.

Part III

Taxonomies, Representations and Stochastic Processes

For the final part of the dissertation, we will study the relation between semantic representations and stochastic processes. Here the notion of semantics is understood under two different paradigms. One, similar to knowledge bases, ontologies, taxonomies, or population-designed folksonomies, are knowledge structures that appear in several web pages for the population to organize user content. Under the approach of representation learning, we also study semantics as representations learn from an autoencoder. The latter view is related to dimensionality reduction and compression, where one develops procedures to extract the data's most relevant coordinates. Different from other chapters, we do not aim at developing an inference procedure. For the folksonomies, we are interested in how these structures affect user behavior. For the auto encoders, we introduce inductive bias to uncover the behavior of black-box classifier decisions.

Chapter 6

Dynamical Inheritance

This chapter dwells in a different direction from the previous agenda. Although a learning algorithm is provided, the focus is on the relationship among knowledge structures as provided by a population and its consequent dynamical behavior. This chapter's character is rather phenomenological in nature, and different from the machine learning nature of the previous chapters. One can argue that the contribution lies in the realm of sociophysics or quantitative social analysis. The task is to establish relationships between user behavior and the knowledge structures they produce and consume in a particular site.

Question Answering sites are Web platforms where users can pose questions to a general population. They gained notoriety over the last decade, and popular sites such as Yahoo Answers, Quora, or the Stack Exchange family of sites such as Stackoverflow and Mathoverflow provide online communities with seamless mechanisms to organize and share knowledge. These platforms' content is usually organized by their users who use tags, hashtags, keywords, or other identifiers to categorize questions they post. This chapter asks if and how the information contained in such knowledge folksonomies influences user behaviors and activities?

Our focus is on the stack exchange family of questions answering sites, where questions address topics in areas such as physics, chemistry, or biology which are rigorous scientific disciplines whose sub-fields can be organized in hierarchical taxonomies that reflect degrees of specialization (think for instance of the ACM Computing Classification System). Hence, when posting questions on a stack exchange site, users typically choose tags that pertain to sub-fields of a particular scientific discipline. The co-occurrence pattern of such tags and their frequencies can, therefore, be assumed to be related to hierarchical structures. Uncovering these structures and understanding their dynamics or evolution over time, therefore, provides insights into the online communities' collective behavior and can help to balance resources, i.e. to recommend questions or tags to users.

6.1 Summary of Contributions

In this chapter, we propose an algorithm to learn hierarchical taxonomies. This algorithm considers the cooccurrences of tags assigned to questions. Questions come with n-tuples of tags that define our data points, and our algorithm infers hidden hierarchies only from sets of co-occurring tags, i.e. from all the n-tuples a given tag appears in. Two main principles are used in order to identify subject hierarchies automatically. First, items or keywords which occur high in the hierarchy are expected to co-occur with a larger set of different tags. Second, parent child relationships are established provided that a child appears in the parent's co-occurrence set and that children of the child also co-occur with the parent. Our latent hierarchy discovering algorithm thus follows ideas from agglomerative clustering. We initialize all available items as the tree leaves and then run a bottom-up procedure to identify parent-child relationship. We define a user tagging process that incorporates the taxonomical organization from the beginning. This allows us to simulate synthetic user behavior,

which provides experimental ground for testing our algorithm. The proposed algorithm unveils the underlying knowledge structure of the analyzed question-answering sites. We finally link the taxonomies with the dynamical behavior of the users posting questions. Our extensive empirical evaluation indicates that the tagging process of parent nodes is highly dependent on the tagging process of their descendants. Thus will give rise to a dynamical inheritance phenomenon.

6.2 Related Work

In research on question answering sites, a common approach towards mining tag relationships and their influence on user behavior is to define a network structure over the tags, either from projections of bipartite tag-question pairs, tag-user pairs (Yang et al. 2008; Adamic et al. 2008; Li et al. 2012; Nam et al. 2009), or through other similarity measures (Begelman et al. 2006). Once networked relations are modeled, traditional graph mining algorithms are employed to discover relevant tags, clusters, and motifs. Although these methods have proven relevant for selecting important or central tags, the relationship between structure and the temporal dynamics have been largely ignored. Under the graph model, there is no notion of inheritance in that a tag does not *pertain* to another tag – a relationship that we found to be relevant in our empirical analysis.

On the other hand, algorithms that yield hierarchies from user annotations have been developed for a variety of sites already (Koller and Sahami 1997; Schmitz 2006; Tibély et al. 2013). These approaches were successful in detecting structure but focused on the algorithmic aspects without providing deeper insights into the behavior of the system in question. However, the dynamics of tagging have been studied in the context of generative models which realize preferential attachment mechanisms (Halpin et al. 2007; Golder and Huberman 2006). These approaches provide plausible explanations as to how the behavior of users leads to a Pareto distribution for the frequencies of tags. In our current work, we focus on the "activities" of tags. That is, does the frequency in which a tag appears depend on the frequency of its co-occurring tags? For example, on the stack exchange site for mathematics, a user posting questions in the field of *linear algebra* might later on post questions about *eigenvectors* or *eigenvalues* and we are interested in uncovering such developments.

We approach this problem by considering tag patterns to be influenced by hierarchies. This way, we can devise a generative model that incorporates the dynamics of the tagging process. Generative models for topic modeling often adhere to the Bayesian framework (Ramage et al. 2009; Blei et al. 2010) which provided tools for detecting hierarchies. The generative process is seen from the point of view of distributions rather than from a dynamical perspective. This framework also comes with the added drawbacks of only handling small taxonomies due to the complexity of approximate inference, and inferring the topics mainly from the content. Ideally, however, tags themselves indicate topics. Our approach focuses directly on tags and their classification and is able to capture different branching sizes for different taxonomies.

6.3 Stack Exchange Data

The stack exchange family of question answering (QA) sites covers a wide range of topics. Regardless of its subject, every site allows subscribed users post questions to their community. Answers are submitted and rated by the community and can be deemed acceptable by the user who posted the question. A major incentive for users to answer questions is a reputation building feature where a user's reputation grows with favorable ratings and increasing numbers of accepted answers. Indeed, reputation scores may nowadays boost careers; *Stackoverflow*, the main site of the stack exchange network, is known to be used as a recruitment platform where companies are looking for knowledgeable talent and experts.

In this chapter, we focus on the analysis of 5 different sub communities of the stack exchange network: Biology, Physics, Finance, Statistics, Math Overflow, and English (see Tab. 6.1).

Our goal is to uncover emergent knowledge structures from sets of questions and to investigate their relationship to user behavior on the different stack exchange sites. Locally, individual questions only refer to a reduced set of subjects which are constrained by the site and the specific concern of the user. It is in the aggregated behavior of users where knowledge taxonomies become apparent (Bhat et al. 2014). Since the tags themselves are expected to identify topics, our main focus is on

Table 6.1 – Statistics of the data from the different stack exchange sites studied in this chapter.

stack exchange site	observed since	# questions	# answers	# users	# tags	# n-tuples
Biology	2011	8958	11628	10631	642	5643
English	2009	57112	147517	91621	947	19381
Finance	2010	4098	6416	8155	495	3186
MathOverflow	2009	63161	102447	46379	2602	28639
Physics	2010	39355	63579	39117	824	24095
Statistics	2009	42921	47755	40324	1032	28232

Site	Ratio #NT/#Q	Ratio #NT/#U
Biology	0.63	0.53
English	0.34	0.21
Finance	0.78	0.39
MathOverflow	0.45	0.62
Physics	0.61	0.62
Statistics	0.66	0.70

Table 6.2 – Main statistics for the Stack Exchange Sites studied

using the overall set of tags, i.e. n-tuples of tags assigned to questions, and not on analyzing the content of questions. Prior work on tag dynamics on induced tag ontologies (Halpin et al. 2007; Ramage et al. 2009) indicate that tags are indeed a reliable proxy for content classification.

Table 6.1 summarizes statistics as to the data we consider in this chapter (observation periods, numbers of questions, answers, users, and tags crawled, and number of observed n-tuples).

Clearly the numbers of site specific sets of observed n-tuples or combinations of tags (which are limited a maximum number of tags the site allows users to use) exceed the number of different observed. If we were to consider, say, all possible pairs of tags possible, we would have $\binom{N}{2} \sim N^2$ choices where N is the number of tags. For the N in our data sets this is in the order of 10^6 whereas the observed number of actually observed different tuples varies only around 10^4.

Table 6.2 lists the ratio between the number of different observed n-tuples and the number of questions and users. These ratios indicate that questions and users apply sets of repeating tuples. This statistics thus suggests that users are not randomly selecting any pairs of tuples but are guided by some latent relationship among the tags.

The main empirical motivation behind our work is the striking similarity between tag frequency distributions and the distribution of the size of sets of co-occurring tags (see Figure 27). Although in principle co-occurrence should by guided by semantic relationships whereas frequency by interest of the populations, the striking similarity between the distributions point to a deeper relationship that we aim to explain in the current work. For both distribution, we fitted the histograms trough maximum likelihood estimation to main statistical distribution functions *Log Normal, Normal, Gamma, Pareto, Weibull, and Exponential.* By comparing Kolmogorov-Smirnov (KS) statistics, we determined the best fitting model to be the Log Normal distribution (shown as solid lines in the figure). Similar to prior work on other community sites, the Log Normal is a fat tailed distribution which, for a certain range of parameter values, behaves as the commonly found Pareto distribution (Halpin et al. 2007). This distribution also complies with our hypothesis that a hierarchical structure modulates the tagging process. If the number of tags which co-occur with a given tag is proportional to the number of leafs in a taxonomic sub-tree emanating from it, then we can obtain this number by the successive multiplication of random variables. These random variables depend on the branching factor of the taxonomy. As it is known (Gallager 2012), the Log Normal process does indeed account for multiplicative processes involving uniform random variables.

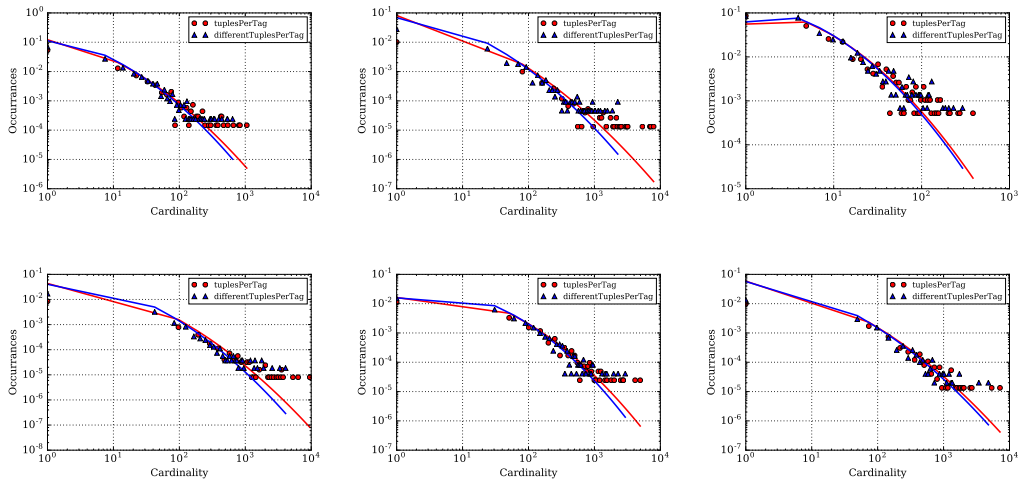


Figure 27 – For each of the studied stack exchange sites, this figure shows the histogram of the number of questions per tag (red circles) and the size of the set of tags which co-occur with each tag (blue triangles). Solid lines correspond to maximum likehood estimations of log normal distributions which show the best fits to the empirical data.

6.4 Definitions and Concepts

We begin our technical discussion by defining the main concepts and notation required below. Let $T \equiv \{t_1, t_2, \ldots\}$ denote the set of different tags or items or keywords contained in our n-tuple data. An n-tuple α is defined as an item set of the form $\alpha \equiv \{t_1(\alpha), t_2(\alpha), \ldots\}$ where $t_i(\alpha) \in T$.

Since we want to infer a tree structured taxonomy which is assumed to have generated the set of tuples, we may henceforth also use the term node when we refer to a tag. Given these prerequisites, we let

- $\Theta \equiv \{\alpha_1, \alpha_2, \ldots\}$ denote the set of observed n-tuples; we do not expect different α_i to have the same size but assume their sizes to be finite
- $O(t_i) = \{t_{i_1}, t_{i_2}, \ldots\}$ denote the set of tags which co-ocurr with tag t_i, that is the set of tags where there exists at least one $\alpha_k \in \Theta$ such that $t_i, t_{i_j} \in \alpha_k$.
- $p(t_i)$ denote the parent of node t_i
- $C(t_i)$ denote the set of children of node t_i
- $D(t_i)$ denote the set of all descendants of node t_i, i.e. the children of the children of node t_i

Based on these concepts, we define the support set of a node t_k over a node t_i as the set

$$S[t_k](t_i) = O(t_k) \cap D(t_i), \tag{133}$$

i.e. as the set of all the descendants of t_i which co-occur with t_k. The relative support is then given by

$$s[t_k](t_i) = \frac{|S[t_k](t_i)|}{|D(t_i)|} \tag{134}$$

and indicates the proportion of the descendants which co-occur with with node t_k.

Furthermore, we define a hierarchy $\mathcal{H}$ as a pair

$$\mathcal{H} \equiv \big(T, P(T)\big) \tag{135}$$

$\alpha_1 = \{B,D\}$ $\alpha_9 = \{A,E\}$ $\alpha_{17} = \{B,A,G\}$
$\alpha_2 = \{E,H\}$ $\alpha_{10} = \{B,E\}$ $\alpha_{18} = \{E,B,F\}$
$\alpha_3 = \{A,C\}$ $\alpha_{11} = \{B,E\}$ $\alpha_{19} = \{E,F\}$
$\alpha_4 = \{A,C\}$ $\alpha_{12} = \{B,A,F\}$ $\alpha_{20} = \{A,D\}$
$\alpha_5 = \{A\}$ $\alpha_{13} = \{E,A,F\}$ $\alpha_{21} = \{B,F\}$
$\alpha_6 = \{E,H\}$ $\alpha_{14} = \{A\}$ $\alpha_{22} = \{B,E\}$
$\alpha_7 = \{A,B\}$ $\alpha_{15} = \{E,G\}$ $\alpha_{23} = \{A,B,H\}$
$\alpha_8 = \{A,D\}$ $\alpha_{16} = \{A,C\}$ $\alpha_{24} = \{A,C\}$

t	2	3
A	7	{C,B,E,D,G,F,H}
B	6	{A,E,D,G,F,H}
E	5	{A,H,B,G,F}
H	3	{A,B,E}
F	3	{A,B,E}
G	3	{A,B,E}
D	2	{A,B}
C	1	{A}

Figure 28 – Data set exaxmple as obtained from a synthetic tree following the anmalous tagging procedure. Rightmost panel shows the data structure used by our taxonomy learning algorithm; it can be obtained directly from the training set.

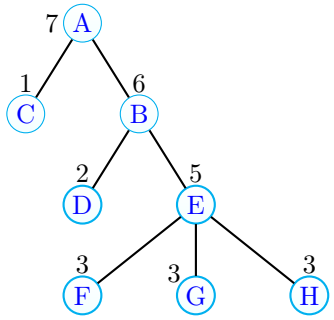


Figure 29 – Example of a taxonomy tree along with training set of n-tuples of tags resulting from a non anomalous tagging process guided by this tree. In the tree, letters represent tags and numbers indicate with how many other tags a tag co-occurs in the training set.

where $P(T) = \{p(t_1), p(t_2), \ldots\}$ denotes the set of parents of all nodes. On the following we will refer to the leafs nodes as those with no children. The ancestors of a node of the set of parents, parents all the way to the root node. Finally, we are prepared to state the following

Problem Definition: Given a set Θ of tuples of tags, find the hierarchy $\mathcal{H}$ which best reflects the joint occurrences of tags in tuples in Θ. In doing so, assume that tag co-occurrence patterns are due to a process described below.

6.5 Tagging Process Model

Our task at hand is to identify a taxonomy or tree structure of topics from observed co-occurrences of tags. This is an unsupervised learning problem where we need to learn a a model (the taxonomy) from a set of data points (the n-tuples of tags). In order for this learning task not to be ill-posed, we shall inform it with prior knowledge that is with a model of a hypothetical process which describes how users assign tags to questions.

To this end, we assume that there are universal hidden hierarchies which define relations between different tags and which are, to some extend, known to the users. For instance, these could be discipline specific taxonomies which define part-of relations among tags. On the biology stack exchange site, for example, *human anatomy* will be a part of *human biology*.

We also assume that the way users assign tags to questions is conditioned on these hierarchies or trees. One of the tags assigned to a question will reside on a tree level less or equal than all other tags. We define this tag or node as the subject node. Due to the coarse to fine inheritance of scientific categories, the question is then related to every tag or node above the given tag. If we follow the branch of all ancestor nodes of the tag up to the root node of the tree, we encounter all

the knowledge areas to which the given question pertains. To describe the content of the question, the user is thus assumed to randomly select tags from this branch. Under this model, the creation of the question is a local process; the user only knows about the branch to which the question pertains. The overall tree, on the other hand, is a global construct; the hierarchical structure we want to uncover emerges from the cumulative process of collective question answering.

6.5.1 Anomalous Tagging Behavior

There is a possibility for anomalous tagging behavior. By this we refer to tagging behavior which is not guided by latent taxonomies but instead arises from random user behavior. For example, a user might pose a question related human-biology *and* botany. Questions like these do not provide information as the latent hierarchy. To be able to argue formally, we define different uninformative tagging process:

1. Children Tagging: once the subject node is selected (conditioned on the latent tree), the user selects the upper branch as well as several children as possible tagging options; this will create co-occurrences between tags on the same level of the tree.
2. Horizontal Tagging: the user selects a subject node, its upper branch as well as random tags which exist on the same level.
3. Random Tagging: the user decides for a subject node and then randomly selects from all nodes on levels above the given one.

Although we can think of additional possible tagging strategies, these anomalous behaviors account for any possible co-occurrences of tags in our data set.

6.5.2 Taxonomy Learning Algorithm

The learning algorithm which we describe next, relies on simple principles which is a direct consequence of the assumed tagging process: if a tag t_a is a parent of t_b, then, provided that anomalous tagging is rare and that t_a has more than one child, the co-occurrence set $O(t_a)$ is larger than $O(t_b)$. This is a consequence of the the descendants of t_b also being descendants of t_a and of the co-occurrence set of t_a containing the descendants $D(t_b)$ of t_b as well as the siblings of t_b.

Given this principle, the proposed inference algorithm then works as follows:

First, we determine the co-occurrence set $O(t_i)$ for each tag t_i in our data set.

We then initialize a list $\mathcal{J}$ containing these these sets and sort it in an ascending order according to the size of its elements (see Fig. 28(c)).

Next, we realize a bottom up process to identify child parent relationships. To this end, we consider t_b is a candidate child of t_a if it co-occurs with t_a, i.e. $t_b \in O(t_a)$, We then look for the t_a that has the biggest support over t_b from the tags up in the list, i.e. t_a is the tag which co-occurs most frequently with all the descendants of t_b and is above a threshold given by the user. Formally

$$t_a = \underset{t_i}{\operatorname{argmax}}\{s[t_i](t_b)\} \text{ and } s[t_i](t_b) > \tau \tag{136}$$

This rule helps to safeguard against the anomalous tagging. Although a particular tag might co-occur with another by random, the subsequent co-occurrence with the descendant set enforces the tree relationship.

In Fig. 28, we show an example of a small taxonomy tree and different n-tuples of tags sampled from the tree. The number attached to each node of the tree indicates the size of the co-occurrence

set of the corresponding tag. According to the exemplary data, tag C will be set as a child of tag A, subsequently tags F, G and H will be set children of E. In Fig. 28(c), each row of the table corresponds to a index of the data structure $\mathcal{J}$. The bottom row corresponds to index 0. We perform one bottom up past over the list to define each relationship.

6.5.3 Validation on Synthetic Taxonomies

In order to quantitatively evaluate the taxonomy learning algorithm, we perform taxonomy inference on synthetic data. Given artificially created taxonomy trees, we sample n-tuples of tags using the above tagging process model. We then apply our algorithm to infer a taxonomy from the data and compare the results to the ground truth, i.e. to the taxonomy tree used for sampling. This allows us to test the algorithm under different anomalous tagging behaviors as well as under different taxonomy structures.

To create artificial taxonomies for testing, we consider a branching probability $P_b(|C(t_i)| = C)$ which indicates how likely a given node t_i has C children and proceed as follows: starting with the root node t_0, we iterate from upper levels down to lower levels. For each node on the current level, we sample its number of children according to P_b and create children correspondingly. In order to assign a name or artificial tag to a newly created node N, we use natural numbers such that $t_0 = 0$ and $t_N = t_{N-1} + 1$. This process continues, until a maximum number of nodes $T_{\max}$ is reached. This construction is in essence a Galton process, similar to the one presented in Section 2.1.3 for the hawkes process.

To quantify the quality of taxonomies inferred from data samples, we applied a variant of the concept of dendogram purity (Heller and Ghahramani 2005). In classification, purity is defined by taking two random leafs and calculating the proportion of leafs from the common ancestor which also belong to the real tree. Instead of class labels, we calculate for all nodes in the tree the proportion of leafs they have in common

$$\langle P \rangle = \frac{|L^{\mathcal{H}}(t_i) \cap L^{\hat{\mathcal{H}}}(t_i)|}{|L^{\mathcal{H}}(t_i)|} \tag{137}$$

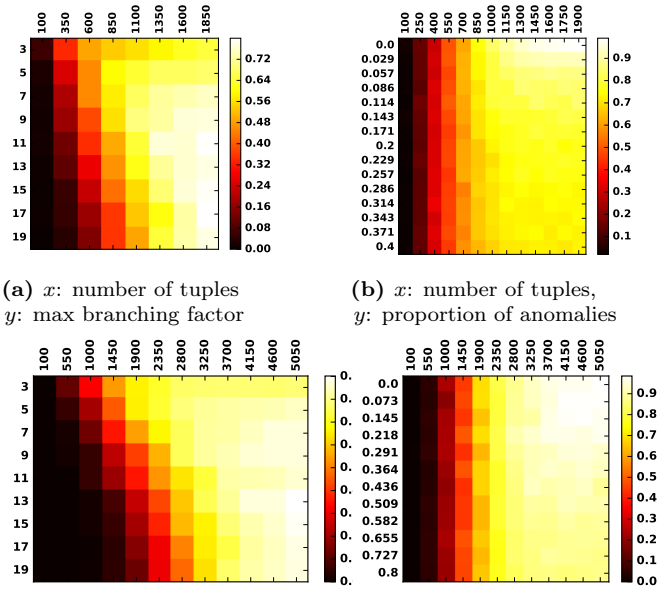
In (137), $\mathcal{H}$ refers to the ground truth tree, whereas $\hat{\mathcal{H}}$ is the tree inferred by our algorithm. Both, for different tag anomalies and different branching factors, the algorithm is observed to perform better for growing numbers of n-tuples and we were able to obtain over 0.8 purity (see Fig. 30 where colors indicate purities).

To investigate the role of taxonomy structure, we tested different branching probability distributions with different parameters. In Figure 30a, a uniform distribution over the interval $[2, u]$ was used and the upper bound u of the branching factor as well as the number of tuples for training were varied. If the number of tuples is too small (below 600), an increase in the branching factor will decrease the purity. For big enough numbers of training tuples, an increase in the branching factor will increase in purity.

In Fig. 30c shows same behavior for a Gaussian branching probability. Here, we fixed the variance $\sigma^2 = 2$ and modified the mean branching factor μ as well as the number of training tuples.

To understand the influence of noise or anomalous behavior according to our process model, we performed inference for different noise levels p, i.e. different proportions of tuples generated using the anomalous behaviors of section 6.5.1.

Figure 30b shows the influence of noise under a uniform branching distribution and Fig. 30d shows it for a Gaussian branching function. In both cases, we observe that for, small numbers of training, tuples noise will always be detrimental w.r.t. purities obtained. However, if enough samples of n-tuples are provided, lower levels of noise lead to higher purities.



(a) x: number of tuples
y: max branching factor

(b) x: number of tuples,
y: proportion of anomalies

(c) x: number of tuples, y: mean of Gaussian branching probability

(d) x: number of tuples,
y: proportion of anomalies

Figure 30 – For quantitative evaluation, of our taxonomy learning algorithm, we performed taxonomy inference on synthetic data sets of n-tuples of tags sampled from given taxonomy trees. The quality of the inferred taxonomies was quantified using the average purity in (137). The number of children per node in the taxonomy trees where sampled from uniform or Gaussian branching functions. The behavior of the algorithm was tested for different sample sizes (x-axis), different branching factors, and different amounts of anomalous samples. For large sample sizes, we consistently obtained purities above 0.8.

6.6 Stack Exchange Results

We performed taxonomy inference for all the Stack Exchange sites in our data set. Figure 4 shows excerpts of the trees we obtained for the biology site, the Physics site, and the English site. Notice that these hierarchies result from from the tagging behavior of the Stack Exchange communities. For instance, in Fig. 4b, general relativity appears as sub-field of quantum field theory. Although this would not be the case in a typical physics taxonomy, an inspection of the siblings of general relativity shows that research level is a sibling indicating that, as a research field, general relativity is a sub-field of quantum field theory. Given the current state of physics research on a unified field theory, this is in deed an acceptable taxonomic classification.

6.6.1 Taxonomy Statistics

To further characterizes the taxonomy trees we obtained and to uncover how topics are structured, we show different dendrogram statistics in Fig. 5. One of the main advantages of our algorithm is its ability to handle unbalanced and highly heterogeneous latent trees. The existence of imbalanced knowledge folksonomies across the various Stack Exchange communities becomes apparent from looking at Fig. 5a. This distribution quantifies the depths of the trees obtained. We note that most of the tags have a preferred branch size of 6 ancestors. The physics taxonomy, however, has longer overall branch sizes. To characterize distribution of tags per level, Fig. 5b shows the number of tags per level. This gives an impression of the widths of the corresponding trees. We note that, for mathematics and statistics, the maximum number of tags is usually obtained around level 3. Physics and English have larger numbers of tags at deeper level of the trees. Together with the longer branch sizes, this indicates that specialization occurs at deeper levels of the tree. Aiming at a structural definition of the tree analogous to the degree distribution for the co-occurrence network, we plot the branching distribution in Fig. 5c which shows the distribution of the number

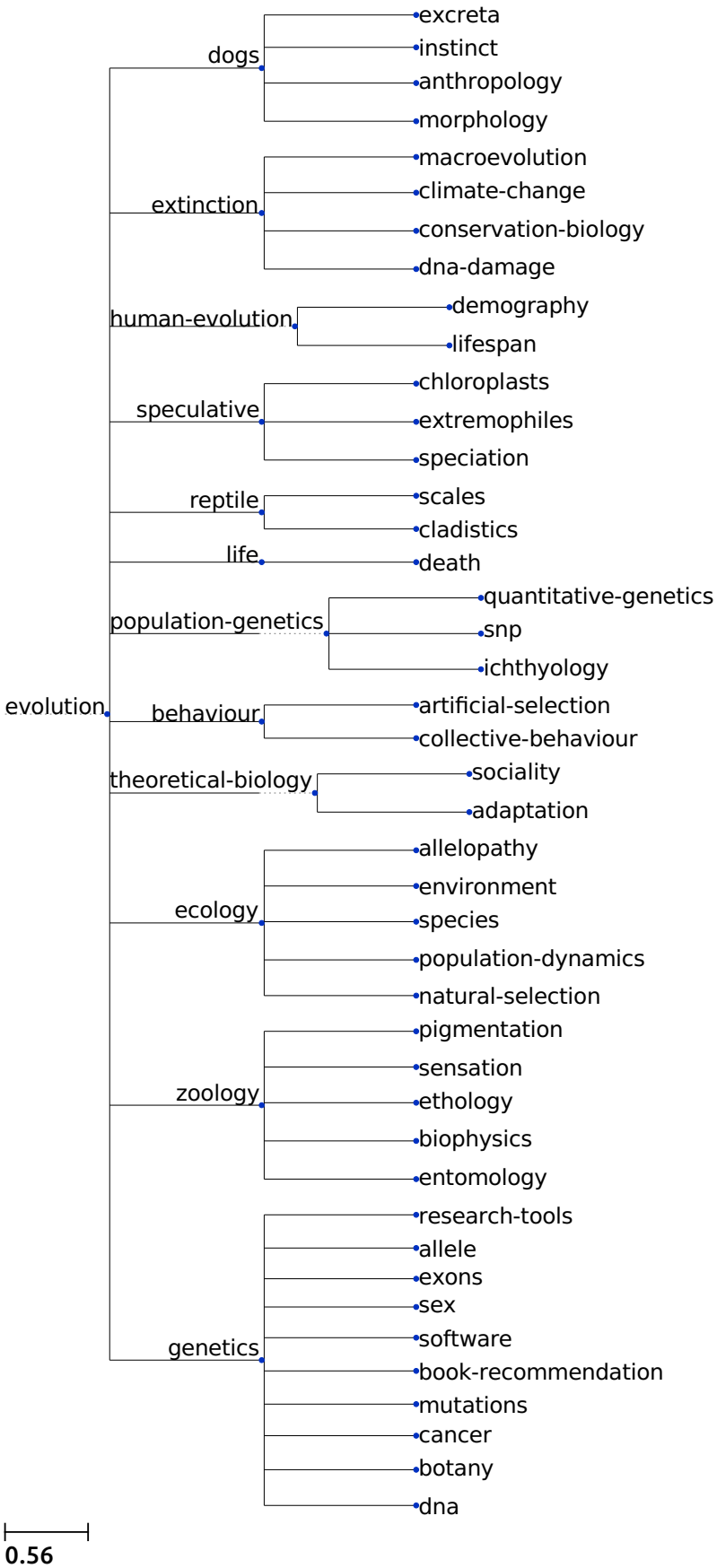


Figure 31 – Biology

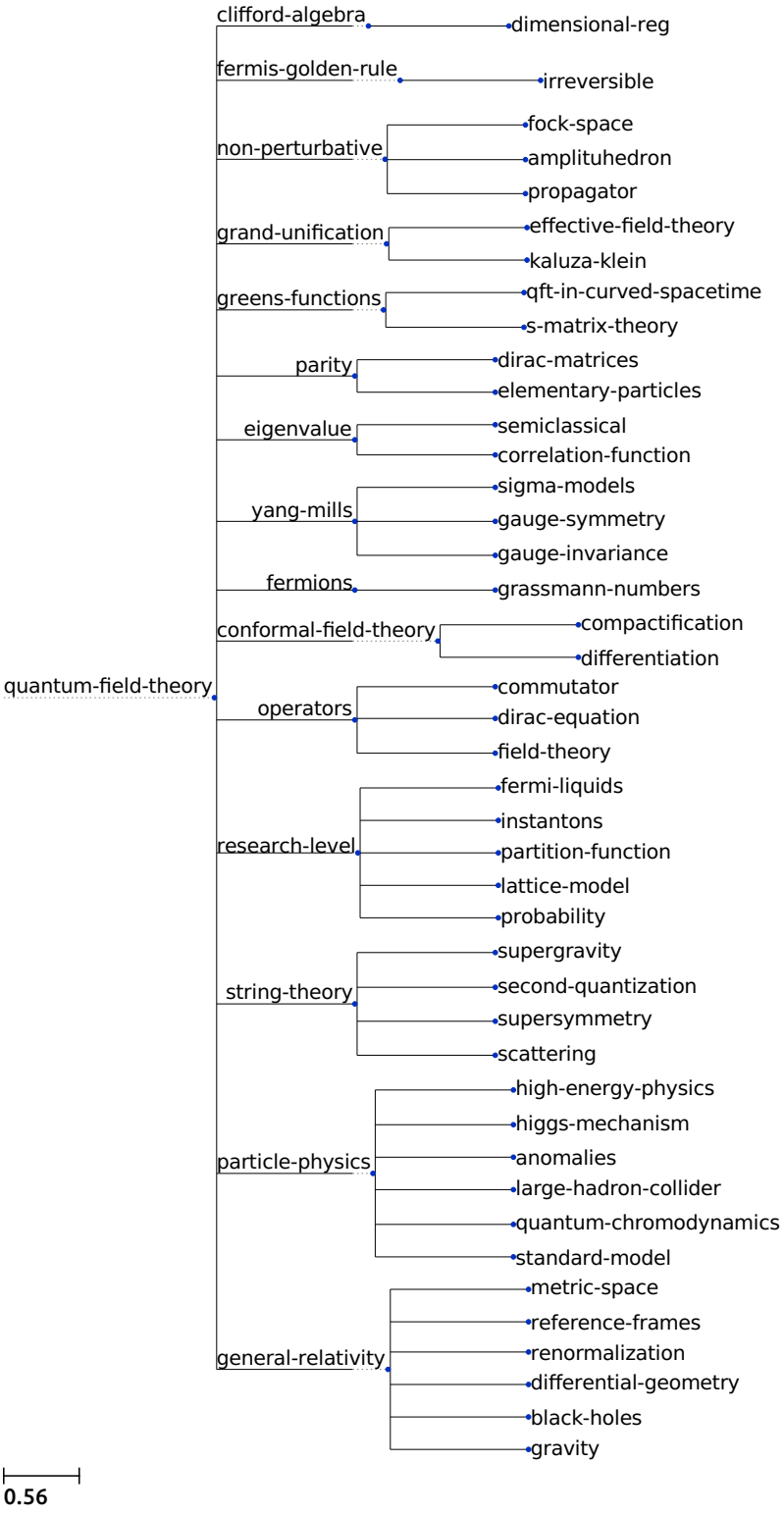


Figure 32 – Biology

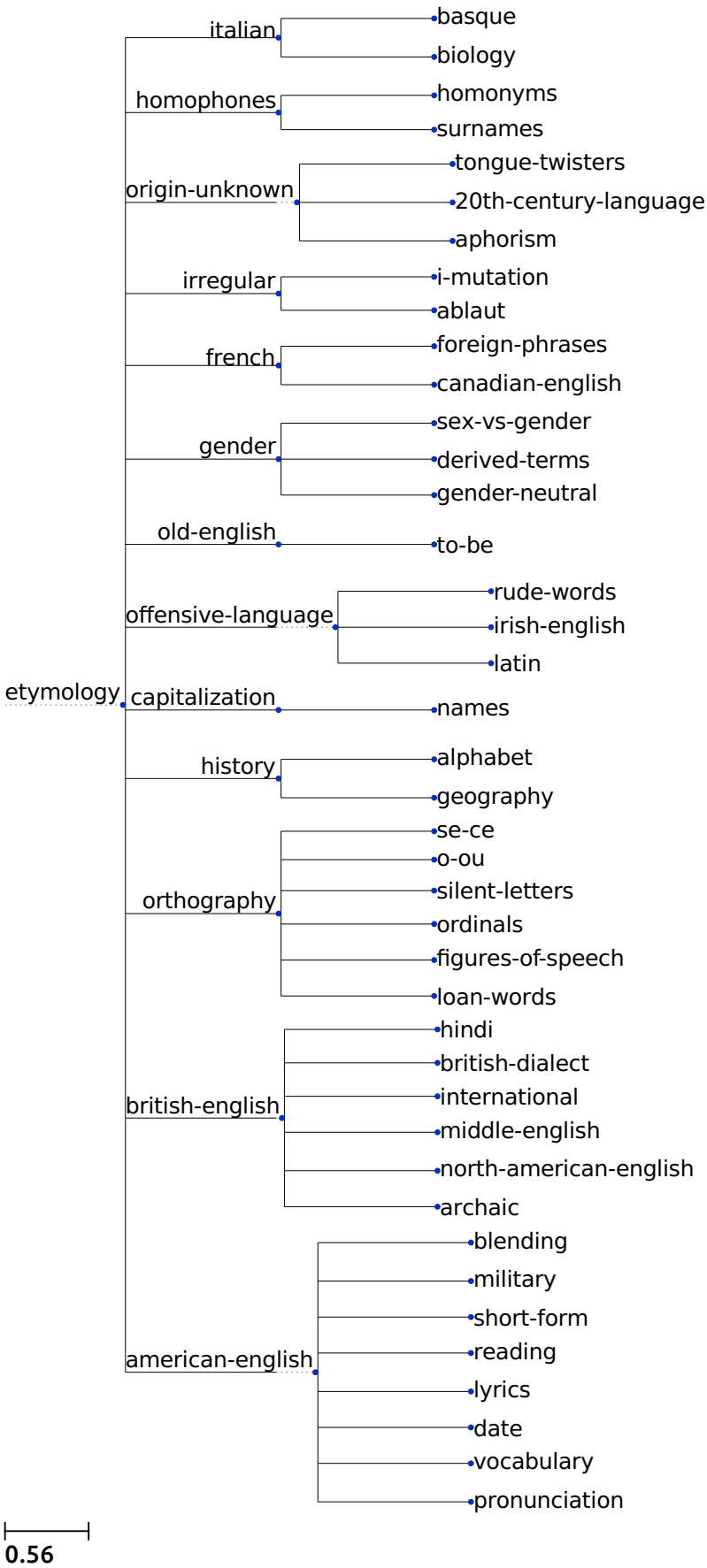
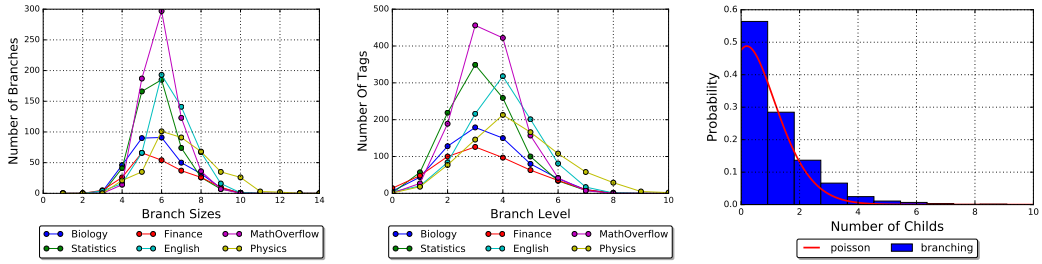


Figure 33 – Biology



(a) branch sizes for different taxonomies **(b)** tags per level **(c)** [branching distribution

Figure 34 – Statistics of the learned taxonomy trees in terms of their empirical branching factors, branch sizes (or depths of the trees), and counts of tags per level (or widths of the trees).

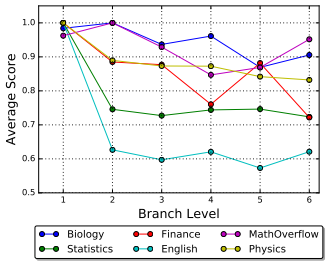


Figure 35 – Average user score per level.

of children per node. After Kolmogorov-Smirnov tests were performed, the Poisson distribution gave the best fit for the branching distribution. The Poisson distribution is known to be a limiting case of the binomial distribution and characterizes random walks on networks with many steps small transition probabilities [20]. In essence, this indicates a random process for the local extension of the taxonomies collaboratively created by the users of the Stack Exchange sites.

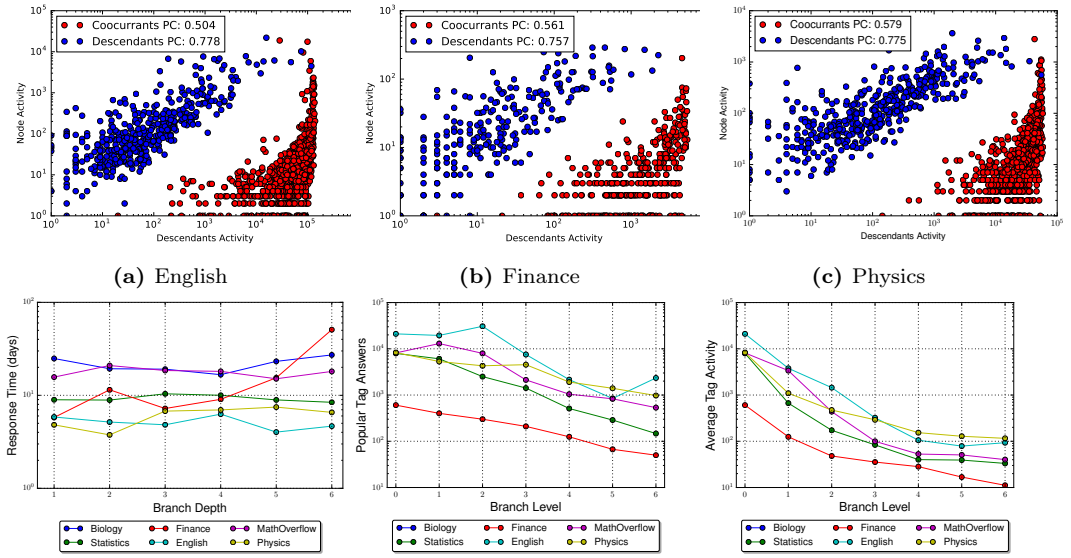
The quality of Stack Exchange answers can be studied by looking at the average score at the different taxonomic levels. Figure 35 shows the average score normalized by the maximum value per site. For all sites, it shows a slightly diminishing trend, possibly because the complexity of the subjects increases on deeper levels of the taxonomies and it may be more difficult for users to provide acceptable answers.

6.6.2 Inverse Dynamical Inheritance

Having inferred taxonomic structures, we are now positioned to study the effect of the taxonomies on the overall behavior of the stack exchange communities. That is we ask: Does the knowledge structure of a field impact the number of answers a particular tag receives?

Here, it is important to note that the taxonomies were derived solely from observed co-occurrences of tags. According to our model, there is a difference between the dynamics of the tagging process and the intrinsic hierarchy of the knowledge. That is, the probability of selecting a given node as the *subject node* is independent of the node's location in the hierarchy. Yet, this location usually depends on the current interest of the population in the corresponding topic. Nonetheless, another node might be selected trough the branch dependencies or a highly active tag might arise as a consequence of users filling knowledge gaps.

We refer to this process as **Inverse Dynamical Inheritance**. The structure of the hierarchy affects the dynamics in a backwards manner. Parent nodes get activated through activities of their descendants. Quantitative results as to this phenomenon can be seen in Fig. 36. Here, we plot the activity of a node against the activity of all its descendants. As defined above, activity is given by the number of answers to questions where a particular tag occurs. In order to remove spurious



(a) English **(b)** Finance **(c)** Physics

(d) response times per level **(e)** maximum tag activity per level **(f)** average tag activity per level

Figure 36 – Dynamical dependance between taxonomy structures and community behaviors; (a)–(c) shown via correlations of the activities related to a parent node and activities related to its descendants (blue); as a reference, activities are also shown for tags and their co-occurrence sets (red); (d)–(f) show the average response times per level, the number of answers for the most popular tag per level, and the average number of answers per level.

correlations due to using the same set of answers, we remove all the answers which where also part of the descendants. Conversely, the answers of the descendant set of tags do not contain the answers where the parent is involved.

For comparison, we also plotted tag activities against the activities of the co-occurrence sets of each tag (devoid of the descendant set $O(t_i) \setminus D(t_i)$). This allows us to study the dynamic dependance as provided trough the hierarchy and not through co-occurrence. For all inferred taxonomy trees, we obtained a Pearson correlation of 0.7 or above for the descendants activities. For the co-occurrence activities, a Pearson correlation of 0.5 was obtained.

Figure 36 also shows changes in the activities per level. Figure 36d displays the average response time per level. Interestingly, fluctuations are small and response times are almost invariant. This is unexpected given the dependance on descendants and that fewer descendants could have been thought to imply less activity since fewer users may specialize in the corresponding topics. Yet, once again, specialization seems not to impact response time. This might be attributed to the reward mechanism of the stack exchange sites. Independent of the taxonomic level, users responds as soon as possible to gain reputation in their community.

Figure 36e shows the number of answers for the most popular tag at different levels. As expected, there is a decreasing trend since deeper levels indicate higher degrees of specialization. Yet, the fluctuations in these curves indicate that tag related activity is not solely level driven. Tag in deeper levels can show more activity than their parents. Finally, average activities per tag per level are displayed in Figure 36f and show a more pronounced decreasing trend.

In this context, it is important to note that, according to our model, if not enough time is allowed for, some co-occurrences may not manifest in observed data due to low activity of certain nodes or topics. That is, nodes have to be active enough so as to be able to sample at least the node with is parent. In this sense, our work in this chapter resembles complex network research where the dynamics of different process, epidemics, random walker, or synchronization phenomena, are constrained by the structure of the network (Newman 2003). This implies that the frequency of appearance of a tag alone is not a good predictor of its position in the tree. In order to develop the

algorithm we focus on the coocurrance patterns, i.e. the tree structure has to be inferred only from the set of tuples which are present in the data set. Ignoring the frequency of the item sets or of the tags themselves.

6.7 Implications for Complex Systems Analysis

The analysis of this work was inspired by the works in the interdisciplinary field of complex systems. However, the relationships between structure and dynamics in past research is slightly different. Hierarchical structures in physics are studied in the context of random walks in ultra-metrics spaces and relaxation in glassy materials (Palmer et al. 1984; Bachas and Huberman 1986), this accounts for phenomena that occur in the hierarchies or hierarchies which arise as a result of complex behavior. In economics, studies have mapped the relationships between agents and macro economic behavior through the use of hierarchical organization as well as the emergence of hierarchical time scales (Aoki 1994; Sornette 2006; Sornette and Johansen 1998). Also, in neuroscience and psychology (Alvarez-Lacalle et al. 2006; Érdi et al. 1992; Saaty 1990), the organizational structure of the brain as well as the semantic representations is studied. In all of these cases, a detailed direct empirical study was lacking due to the difficulty of obtaining the specifics of the dynamics and structure. The results were indirect or restricted to mathematical modeling. Our work presents an empirical investigation along those lines as we are able to observe in detail the dynamical process at all levels of the tree. As a whole, the questions answering sites represent a complex system which process information as the system seeks to answers the question provided by some users. The dynamical inheritance implies that if a given site requires a particular point in the tree to increase it activities, it can do so by the activities of its descendants. The topological characteristics of the tree encoded in metrics as the branching distributions will impose constrains in the dynamics. In a sense, the tree structure give insights as to how to aggregate the behavior of the different tags which depends upon the given parent in the same way that a graph allows to propagate an epidemic through its edges. The ubiquity of the tagging paradigm indicates that this agenda can be extended to sites as diverse as Twitter or Instagram.

6.8 Discussion and Outlook

In the present chapter, we presented an algorithm to infer knowledge taxonomies from n-tuples of tags assigned to questions on the stack exchange family of question-answering sites. The algorithm was based on a probabilistic model of the tagging process, and extensive quantitative evaluations on data for which there was ground truth showed favorable performance characteristics.

We applied the proposed algorithm to data crawled from stack exchange and used our results to analyze dynamics within the stack exchange communities and verify whether these depend on the respective fields' taxonomies. We found that the automatically uncovered taxonomies and relations between tags can account for certain tags' popularity. The approach proposed in this chapter provides a way of encoding the knowledge structures of QA sites that differs from previous graph-based models in that it naturally incorporates coarse-to-fine relations. In future work, we intend to study how a user's reputation is related to their level of expertise as reflected by where in the taxonomy, their answers are located.

In the next and final chapter of the dissertation, we will work in the opposite direction presented in the current chapter. Instead of obtaining the effects of the representation structures in dynamics, but instead, we will define representations constrained in specific properties of the stochastic processes defined from them. Instead of ontologies, we will work with autoencoders, both from a variational inference and adversarial inference approach. The nature of the paper is somewhat theoretical, as proofs for the constructions are presented.

Chapter 7

Auto Encoding Explanatory Examples

A considerable drawback of the deep classification paradigm is its inability to provide explanations as to why a particular model arrives at a decision. This black-box nature of deep systems is one of the main reasons why practitioners often hesitate to incorporate deep learning solutions in application areas, where legal or regulatory requirements demand decision-making processes to be transparent. A state-of-the-art approach to explain misclassification is saliency maps, which can reveal a classifier's sensitivity to its inputs. Recent work (Adebayo et al. 2018), however, indicates that such methods can be misleading since their results are at times independent of the model, and therefore do not provide explanations for its decisions. The failure to correctly explain some of these methods lies in their sensibility to feature space changes, i.e. saliency maps do not leverage higher semantic representations of the data. This motivates us to provide explanations that exploit the semantic content of the data and its relationship with the classifier. Thus we are concerned with the question: *can one find semantic differences which characterize a classifier's decision?*

In this work, we propose a formalism that differs from saliency maps. Instead of characterizing particular data points, we aim to generate a set of examples that highlight differences in the decision of a black-box model. Let us consider the task of image classification and assume a misclassification has taken place. For example, imagine that a female individual was mistakenly classified as male, or a smiling face was classified as not smiling. Our main idea is to articulate explanations for such misclassifications through sets of semantically connected examples that link the misclassified image with a correctly classified one. Starting with the misclassified point, we suitably change its features until we arrive at the correctly classified image. Tracking the black-box output probability while changing these features can help articulate why the misclassification happened in the first place. Now, how does one generate such a set of semantically-connected examples? Here we propose a solution based on a variational auto-encoder framework.

We use interpolations in latent space to generate a set of examples in feature space connecting the misclassified and the correctly classified points. We then condition the resulting feature-space paths on the black-box classifier's decisions via a user-defined functional. Optimizing the latter over the space of paths allows us to find paths that highlight classification differences, e.g. paths along which the classifier's decision changes only once and as fast as possible. A basic outline of our approach is given in Fig. 37.

Note that providing explanations is a rather subjective task. Here I can mention the different approaches, and explanin that one thig is to explain a aparticular desicion and another is to explain a classifiers, and the limits of non linears manifolds.

Within our framework, this implies that the types of changes that a particular user expects to see along an example path are likely data and application dependent. We thus propose a general formalism that allows users to specify the nature of the example path providing the explanation

that better suits their needs. We achieve this by introducing and formalizing the notion of *stochastic semantic paths*: stochastic processes on feature space created by latent code interpolations. Expected changes of a data point are characterized in terms of stochastic functionals along the path, which leads to the notion of a semantic Lagrangian. To train, say, a Variational Auto-Encoder, one must define a new training cost by solving the variational problem that minimizes the functional along the paths.

7.0.1 Summary of the Chapter

In what follows, we introduce and formalize the notion of stochastic semantic paths — stochastic processes on feature (data) space created by decoding latent code interpolations. We formulate the corresponding path integral formalism, which allows for a Lagrangian formulation of the problem, viz. how to condition stochastic semantic paths on the output probabilities of black-box models, and introduce an example Lagrangian which tracks the classifier's decision along the paths. We show the explanatory power of our approach on the MNIST and FashionMNIST datasets.

7.1 Related Work

The bulk of the explanation literature for deep/ black box models relies in input dependent methodologies. Gradient Based approaches (Simonyan et al. 2013a; Erhan et al. 2009) derive a sensibility score for a given input example and class label by computing gradient of the classifier with respect to each input dimension. Generalization of this methodology address gradient saturation by incorporating gradients values in the saliency map (Shrikumar et al. 2016) or integrating scaled versions of the input (Sundararajan et al. 2017). Ad hoc modifications of the gradient explanation via selection of if required value (Springenberg et al. 2014a) and (Zeiler and Fergus 2014) as well as direct studies of final layers of the convolutions units of the classifiers (Selvaraju et al. 2016a) are also provided.

Different from gradient based approaches, a different categories of explanatory models rely on *reference based approaches* which modify certain inputs with uninformative reference values (Shrikumar et al. 2017). Bayesian approaches, treat inputs as hidden variables and marginalize over the distribution to obtain the saliency of the input (Zintgraf et al. 2017). More recent generalizations exploit a variational Bernoulli distribution over the pixels values (Chang et al. 2018).

Other successfully methodologies include substitution of black box model with locally interpretable linear classifiers. This is further extended to select examples from the data points in such a way that the latter reflect the most informative components in the linear explanations. (Ribeiro et al. 2016).

Studies of Auto encoder interpolations seek to guarantee reconstruction quality. In (Arvanitidis et al. 2017) authors characterize latent space distortions compared to the input space through an stochastic Riemannian metric. Other solutions include adversarial cost on the interpolations such as to improve interpolation quality compare to the reconstructions (Berthelot et al. 2018). Examples which are able to fool classifier decisions have been widely studied in the framework of adversarial examples (Goodfellow et al. 2014b). This methodologies however do not provide interpretable explanations or highlight any semantic differences which lead to classifier decisions.

7.2 Explanations

We are concerned with the problem of explaining a particular decision of a black-box model. In broad terms, to explain we mean (Ribeiro et al. 2016) *to provide textual or visual artifacts that provide qualitative understanding of the relationship between the data points and the model prediction.* Recent attempts to clarify such a broad notion of explanation c.f. (Doshi-Velez et al. 2017) require the answers to questions such as: (1) *What were the main factors in a decision?*, as well as (2) *Would changing a certain factor have changed the decision?*. To provide an answer to such questions, one must be able to define a clear notion of *factors*. One can think of factors as the minimal set of coordinates that allows us to describe the data points. A good description is then given by information which allows us to reconstruct a data point. As such, this definition mirrors the behavior of the autoencoder code (defined below). By training an autoencoder one can find a code which describes a particular data point. Our role here is to provide a connection between these codes and the classifier's decision. Changes on the code should change the classification decision in a user-defined way. Defining such a code will allow us to formalize the framework required to provide an answer to question (1) and (2). Following (Ribeiro et al. 2016) we require explanations to be **agnostic**, i.e independent of the model, **interpretable**, and expressing **local fidelity**.

7.3 Explaining Through Examples

We define a defendant black-box model $\mathcal{B}(l|x)$ as a classifier which provides classification decisions l over the data x. This model was trained on a dataset $\mathcal{D} = \{(x_i, l_i)\}$, where $x \in \mathcal{X}$ is the data in feature space. A plaintiff is defined as the tuple (l_0, x_0). A litigation case is defined as the 6-tuple $\mathcal{C}_L = (x_{-T}, x_T, l_0, x_0, l_t, \mathcal{B}(l|x))$, wherein a plaintiff with a given data x_0 presents a complain over a particular classification decision l_0 of model $\mathcal{B}$, and provides a desired solution l_t. x_{-T} is a representative data point for label l_0, whereas x_T is a representative data point for the label l_t. These representatives should provide examples for which the classifier correctly performs the classification, as expected by the plaintiff, the defendant (if agreed) or the institution upon which the complain or litigation case is presented (say the court). For a given generative model $P_\theta(X)$ of the data — possible defined by an autoencoder: an explanation is an example set $\mathcal{E} = \{x^{\mathcal{E}}_{-E}, \ldots, x^{\mathcal{E}}_0, \ldots, x^{\mathcal{E}}_E\}$ where $x^{\mathcal{E}}_k \sim P_\theta(X)$. The index k runs over semantic changes that highlight classification decisions. This example set constitutes the context revealing how factor changes impact the classification decision (see Section 7.2). In this chapter we provide explanations for particular representative pairs. However, the inherent multimodality of the data distribution for each label might require explanations for several representative examples. One could then perform the procedure proposed here multiple times for different representative pairs.

7.4 Semantics and Example Generation: Auto-Encoders

Following the discussion above, we use the autoencoder formalism to introduce a notion of semantics useful to qualitatively explain the decisions of a black-box classifier. This formalism will allow us to both infer a code from the data point which inputs the classifier, and define a generative model of examples associated to that code. Let us denote the data (feature) space by $\mathcal{X}$ and the latent space of codes (describing the data) by $\mathcal{Z}$, where usually $\dim(\mathcal{Z}) < \dim(\mathcal{X})$. We generically think of an autoencoder as a tuple of (either, stochastic or deterministic) maps: an encoding map from $\mathcal{X}$ to $\mathcal{Z}$, defined by $z = Q_\phi(z|x)$ parametrized by ϕ, and a decoding map from $\mathcal{Z}$ to $\mathcal{X}$, given by $\tilde{x} = P_\theta(\tilde{x}|z)$, with θ its parameter set and $\tilde{x} \in \mathcal{X}$ an approximate reconstruction of the input x. The autoencoder formalism allows to efficiently infer a latent variable code z associated to a data point x by training both P_θ and Q_ϕ to minimize some reconstruction error $c(x, \tilde{x})$ between x and its reconstruction $\tilde{x}$. Once the model is trained one can think of the inferred code z as containing some high-level description of the input data x.

Within the VAE framework (Kingma and Welling 2013a) the latent code becomes a random variable Z distributed according to some fixed prior distribution $P(Z)$. Both encoder and decoder maps become stochastic within this picture, and are constrained to have a computable and differentiable probability density. That is, the encoder and decoder are now given by the distributions $Q_\phi(Z|X)$ and $P_\theta(X|Z)$, with X, Z random variables taking values in $\mathcal{X}, \mathcal{Z}$, respectively[1]. The encoder thus provides an approximation to the true posterior distribution $P(Z|X)$ over the latent code, whereas the decoder's density, which we denote by $p_\theta(x|z)$, yields the likelihood function of the data given the code. Following the exposition presented in Eq. 5. VAE are then trained by minimizing:

$$L_{\text{VAE}} = -\mathbb{E}_{P_D(X)}\mathbb{E}_{Q_\phi(Z|X)}\left[\log p_\theta(x|z)\right] + D_{\text{KL}}\left(Q_\phi(Z|X), P(Z)\right), \tag{138}$$

where $P_D(X)$ corresponds to the input data distribution and D_{KL} denotes the Kullback-Leibler divergence between the prior and the approximate posterior distribution. In contrast to VAE, within the Wasserstein Autoencoder framework (WAE) (Tolstikhin et al. 2018) one only needs to be able to sample from $Q_\phi(Z|X)$ and $P_\theta(X|Z)$ — i.e. their density is not needed. Similar to the adversarial training presented in Section 4.3.2. WAE is trained by minimizing a (penalized) optimal transport divergence (Bousquet et al. 2017) — the Wasserstein distance, between the input data

[1] In the literature both $Q_\phi(Z|X)$ and $P_\theta(X|Z)$ are usually chosen to be Normal distributions parametrized by neural networks.

distribution $P_D(X)$ and the implicit latent variable model $P_\theta(X)$. As in VAE, the latter is defined by first sampling Z from $P(Z)$ and then mapping Z to X through the decoder $P_\theta(X|Z)$. The loss function of WAE is given by

$$L_{\text{WAE}} = \mathbb{E}_{P_D(X)}\mathbb{E}_{Q_\phi(Z|X)}\left[c\left(X, P_\theta(X|Z)\right)\right] + \lambda D_Z\left(Q_\phi(Z), P(Z)\right), \tag{139}$$

where c is a distance function and D_Z is an arbitraty divergence between the prior $P(Z)$ and the agregate posterior $Q_\phi(Z) = \mathbb{E}_{P_D(X)}\left[Q_\phi(Z|X)\right]$, weighted by a positive hyperparameter λ. Minimizing Eq. (139) corresponds to minimizing the Wasserstein distance if the decoder is deterministic (i.e. $P_\theta(X|Z=z) = \delta_{g_\theta(z)} \forall z \in \mathcal{Z}$, with the map $g_\theta : \mathcal{Z} \to \mathcal{X}$) and the distance term is optimized. If the decoder is stochastic Eq. (139) yields an upper bound on the Wasserstein distance (Bousquet et al. 2017). It is essential to notice the fundamental difference from that which was presented in Section 4.3.2. Here we are interested in obtaining a representation Z, whereas in Section 4.3.2, only the ability to sample from the data distribution was required.

7.5 Stochastic Semantic Processes and Corresponding Paths

In this section we first formalize the notion of semantic change by introducing the concept of (stochastic) semantic interpolations in feature space $\mathcal{X}$. This will allow us to generate examples which provide *local fidelity*, as the examples are smooth modifications of the latent code associated to the plaintiff data point x_0. We then define a collection of probability measures over semantic paths in $\mathcal{X}$. These measures will be used later in Section 7.6 to constrain the paths to be explanatory with respect to the classifier's decision.

7.5.1 Semantic Interpolations

One of the main motivations behind the VAE formalism is the ability of the inferred latent code z to provide semantic high-level information over the data set. If one is to generate examples which have characteristics common to two different data points, say x_0 and x_T from the litigation case, one can perform interpolations between the latent codes of these points, that is z_0 and z_T, and then decode the points along the interpolation. A main observation is that these interpolations in latent space can be used to **induce certain interpolating stochastic processes on feature space**[2] $\mathcal{X}$. We refer to these as *stochastic semantic processes*. In what follows, we first focus on *linear* latent interpolations, i.e.

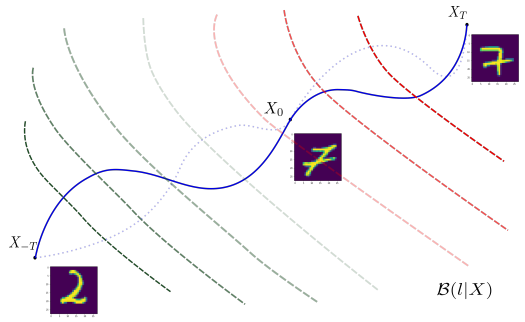
$$z(t) := t\, z_0 + (1-t) z_T, \tag{140}$$

and construct an interpolating stochastic semantic process X_t on $\mathcal{X}$ by using the decoder distribution $P_\theta(X|Z=z(t))$. In practice, the generation process of such stochastic interpolations consists then of three steps: (i) sample $Q_\phi(Z|X)$ at the end points x_0 and x_T using the reparametrization trick (Kingma and Welling 2013a), (ii) choose a set of points z_t along the line connecting z_0 and z_T and (iii) decode the z_t by sampling $P_\theta(X|Z=z_t)$. A formal description of this procedure is given below, in subsection 7.5.2, and an impression of the stochastic process thus constructed is presented in Fig. 40c.

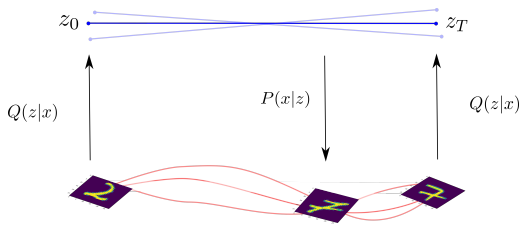
7.5.2 An Approach via Explicit Family of Measures

We observe that for every sequence of points $\{t_i\}_{i=0}^n$ there is a natural measure on piecewise linear paths starting at $x_0 \in \mathcal{X}$ and terminating at $x_T \in \mathcal{X}$. More precisely, we define the probability of a

[2] Moreover, under appropriate assumptions on the auto-encoder mappings (P_θ, Q_ϕ) the proposed induced stochastic processes could posses additional properties (e.g. trajectory regularity, controlled moments, etc).



(a) Paths in feature space. Lines with slahes, corresponds to level lines for the classifier values.



(b) Propagator. Upper lines corresponds to the interpolation of the z variables, while lower line corresponds to samples in data space.

Figure 37 – Stochastic Semantic Paths

piecewise linear path $x(t)$ with nodes $x_1, x_2 \ldots, x_n \in \mathcal{X}$ as

$$dP_{t_0,\ldots,t_n}(x(t)) := \int_{\mathcal{Z}} \int_{\mathcal{Z}} \left(\prod_{i=1}^{n} p_\theta(x_i|z(t_i)) \right) \times q_\phi(z_0|x_0) q_\phi(z_T|x_T)\, dz_0\, dz_T, \tag{141}$$

where q_ϕ, p_θ label the densities of Q_ϕ, P_θ, respectively, and where $z(t)$ is defined by eq. (140) [3]. In other words, for every pair of points x_0 and x_T in feature space, and its corresponding code samples $z_0 \sim Q_\phi(Z|X = x_0)$ and $z_T \sim Q_\phi(Z|X = x_T)$, the decoder $P_\theta(X|Z)$ induces a measure over the space of paths $\{x(t)|x(0) = x_0, x(T) = x_T\}$. Formally speaking, the collection of measures $d\mathcal{P}_{t_0,\ldots,t_n}$ given by different choices of points $\{t_i\}_{i=0}^n$ in (141) defines a family of consistent measures (cf. Definition in the Appendix, Subsection Consistent-Measures). This implies that these different measures are assembled into a stochastic process on feature space $\mathcal{X}$ over the continuous interval $[0, T]$:

Proposition 1 *The collection of measures prescribed by (141) induces a corresponding continuous-time stochastic process. Moreover, under appropriate reconstruction assumptions on the auto-encoder mappings P_θ, Q_ϕ, the sample paths are interpolations, that is, start and terminate respectively at x_0, x_T almost surely.*

The statement goes along the lines of classical results on existence of product measures. For the sake of completeness we provide all the necessary technical details in the Appendix. Another important remark is that the stochastic semantic process construction in Proposition 1 is just one way to define such a process — there are other natural options, e.g. in terms of explicit transition kernels or Itô processes.

7.6 Principle of Least Semantic Action

Having described a procedure to sample stochastic semantic processes in $\mathcal{X}$, we need to discover auto-encoding mappings (P_θ, Q_ϕ) that give rise to reasonable and interesting stochastic paths. Specifically, to generate examples which are able to explain the defendant black-box model $b(l, x)$ in the current litigation case (Section 7.3), one needs to ensure that semantic paths between the data points x_0 and x_T *highlight classification differences*, i.e. classifications of the model along this path are far apart in the plaintiff pair of labels. Thus, to design auto-encoding mappings P_θ, Q_ϕ accordingly, we propose an optimization problem of the form

$$\min_{\theta,\phi} S_{P_\theta,Q_\phi}[X_t], \tag{142}$$

where X_t is a stochastic semantic process and S_{P_θ,Q_ϕ} is an appropriately selected functional that extracts certain features of the black-box model $b(l, x)$. A couple of remarks are in place:

[3] We remark that the integral (eq. 141) is, moreover, finite, if, for example, the densities p_θ are bounded with respect to z.

- The functional $\mathcal{S}_{P,Q}$ is tailored along the problem at hand - in our example above $\mathcal{S}_{P,Q}$ should be defined as to extract certain features of the black box model $\mathcal{B}_\theta(l|x)$,
- An important remark related to the the variational problem (142) is the following: one could develop plenty of meaningful functionals $\mathcal{S}_{P,Q}$ that involve taking velocities or higher derivatives - to this end, one is supposed to work over spaces of curves with certain regularity assumptions. However, as stated above we are working over stochastic paths X_t whose regularity is, in general, difficult to guarantee (we refer to the Appendix for further remarks on Hölder regularity and diffusion Itô processes). A straightforward way to alleviate this issue is to consider a "smooth" version of the curve X_t - e.g. by sampling X_t through a decoder with controllable or negligible variance or by means of an appropriate convolution.

  Furthermore, one could also approach such stochastic variational analysis via Malliavin calculus - however, in the present work we do not pursue this direction (cf. Section 7.9).

The minimization problem (142) can be seen in the context of Lagrangian mechanics. In mechanics, the optimization given by suitable Lagrangians delivers physically meaningful paths, e.g. following equations of motion ((Landau and Lifshitz 2013)). In our case, a guiding intuition is that the semantic Lagrangian should reflect how the black-box $\mathcal{B}$ was taking decisions along the path X_t, starting at x_0 and ending at x_T - e.g. whether the classifications along X_t were certain (from the point of view of $\mathcal{B}$); whether the decisions were taken in a gradually changing fashion; etc. In this direction, minimization of the semantic action (i.e. finding minimizing paths X_t) should make these classification aspects prominent. In our case, the paths are sampled from the measure induced by the autoencoder pair. To find semantic paths translates into finding auto encoders pairs.

For a given stochastic semantic process X_t, and given initial and final feature "states" x_0 and x_T, we introduce the following function, named the *model-b semantic Lagrangian*

$$\mathcal{L} : [0,1] \times \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (t, x_0, x_T) \mapsto \mathcal{L}[X_t, x_0, x_T], \tag{143}$$

which gives rise to the *semantic model action*:

$$S[X_t] := \int_0^T \mathcal{L}[X_t, x_0, x_T]dt. \tag{144}$$

Recalling that the semantic stochastic processes are constructed through the encoder/decoder mechanism, we come to the heart of our discussion: by optimizing and further tuning the autoencoder pair P, Q we aim to generate and sample stochastic processes X_t that minimize the semantic action $\mathcal{S}$. In other words, we attempt to find autoencoder pairs that would allow us to generate meaningful explanations in the form of example collections given by semantic stochastic paths. Specifically, minimization of (144) plays a similar role to that of the well-known variational lower bound in the variational autoencoding framework - one can discretize the integral and perform the reparametrization trick at each time step in order to average over the semantic stochastic paths. In our case, the paths are sampled from the measure induced by the autoencoder pair. To find semantic paths translates into finding auto encoders pairs. Our problem, viz. to find encoding mappings P_θ, Q_ϕ which yield explainable semantic paths with respect to a black-box model, is then a constrain optimization problem whose total objective function we write as

$$L(\theta, \phi) := L_{\text{VAE}}(\theta, \phi) + \lambda \, \mathbb{E}_{dP[x(t)]} S[x(t)], \tag{145}$$

where L_{VAE} is given by eq. (138), $S[x(t)]$ corresponds to the Lagrangian action and λ is an hyper parameter controlling the action' scale. The average over the paths (Majumdar 2007; Feynman and Mechanics 1965) is taken with respect to the stochastic paths and the corresponding measure $dP[x(t)]$ from Proposition 1, that is, the path integral

$$\mathbb{E}_{dP[x(t)]} S[(x(t))] = \int \mathcal{L}[x(t), x_0, x_T] dP[x(t)] \tag{146}$$

$$\approx \frac{1}{nK} \sum_k^K \sum_t^n \mathcal{L}[x_t^k, x_0, x_T], \tag{147}$$

where x_t^k labels the tth point along the the kth path, sampled as described in Section 7.5, n is the number of points on each path, K is the total number of paths, and the estimator on the right hand side corresponds to an explicit average over paths[4]. In practice, both L_{VAE} and the action

Algorithm 5: PATH Auto-Encoder

Data: Dataset $\mathcal{D} = \{(x_i, l_i)\}$ Litigation case $(x_{-T}, x_T, l_0, x_0, l_t, \mathcal{B}(l|x))$
Encoder $P_\theta(x|z)$, Decoder $Q_\phi(z|x)$

while *ϕ and θ not converged* **do**
- Draw $\{x_1, ..., x_n\}$ from the training set
- Calculate Auto-Encoder Loss $L_{\text{VAE}}(\theta, \phi)$
- Sample Litigation Codes
- $z_{-T} \sim Q_\phi(Z|x_{-T})$, $z_0 \sim Q_\phi(Z|x_0)$, $z_T \sim Q_\phi(Z|x_T)$
- Generate Latent Interpolations
- $t_j^k \sim$ Sort(Uniform(0,1))
- $z_j^k = z_{-T} \times t_j^k + z_0 \times (1 - t_j^k)$
- Sample k Paths in Feature Space
- $x_t^k \sim P_\theta(X|z_j^k)$
- Evaluate Semantic Action for each path k
- and average over k
- $L_{\mathcal{S}} = \mathbb{E}_{d\mathcal{P}[x(t)]}[\mathcal{S}(x(t))]$
- Update P_θ and Q_ϕ by descending: $L_{\text{VAE}}(\theta, \phi) + L_{\mathcal{S}}(\theta, \phi)$

end
return P_θ, Q_ϕ

term are optimized simultaneously. Note that the VAE loss function L_{VAE} is trained on the entire data set on which the black-box performs. The action term, in contrast, only sees the x_0 and x_T points. This can be seen explicitly in Algorithm 5, which shows an overview of the auto-encoder pair training algorithm. Let us finally note that, drawing analogies with the adversarial formalism (Goodfellow et al. 2014b), the defendant black-box model plays the role of a fixed discriminator, not guiding the example generation, but the interpolations among these examples.

7.6.1 The Choice of Lagrangians

There are plenty of options for Lagrangian functionals that provide reasonable (stochastic) example-paths — roughly speaking, we attempt to define an objective value for a certain subjective notion of explanations. In what follows we illustrate three different Lagrangians.

Minimum Hesitant Path

We want to find an example path such that the classifier's decisions along it changes as quickly as possible, as to highly certain regions in $\mathcal{X}$. In other words, the path is forced to stay in regions where the black-box produces decisions with maximum/minimum probability. An intuitive way to enforce this is via the simple Lagrangian

$$\mathcal{L}_1(x(t), x_0, x_T) := -\left(b(l_T, x(t)) - b(l_0, x(t))\right)^2, \tag{148}$$

where l_0, l_T are the labels of the litigation case in question. Roughly speaking, given the appropriate initial conditions, the paths that minimize the action associated to $\mathcal{L}_1$ are paths that attempt to keep $\mathcal{L}_1$ close to 1 over almost the entire interpolation interval.

[4]Note that, as mention in Sec. 7.5, one must resort to the reparametrization trick to sample from Q_ϕ and efficiently evaluate the gradients of the action term. Note also that Proposition 1 tells us that different choices of the discrete (approximation) grids in the t integration are qualitatively related to the same underlying stochastic interpolation process.

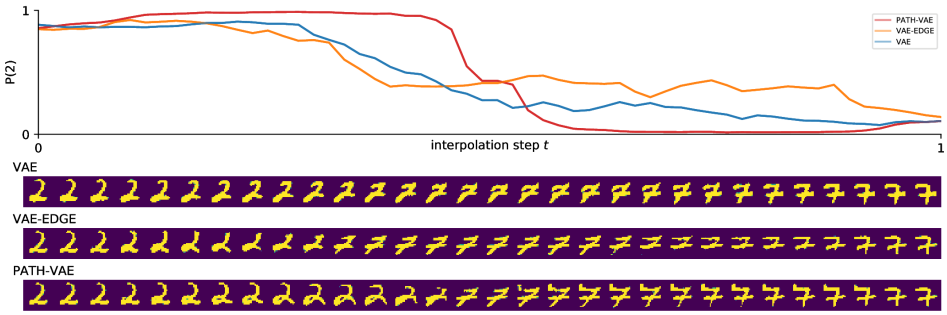


Figure 38 – Probability Paths for the litigation case $l_0 = 2, l_T = 7$. Y axis corresponds to classification probability and x axis corresponds to interpolation index. Interpolation images for a specific paths are presented below the x axis.

Minimum Transformation Path

Another meaningful Lagrangian construction is given by following the geometry of $\mathcal{B}$ itself and attempting to find paths that are close to being gradient-descent lines. This can be embodied by defining

$$\mathcal{S}_3(x(t), x_0, x_t) := \int_0^T \|\nabla \mathcal{B}(l_T|x(t)) - \alpha \dot{x}(t)\|^2 dt := \int_0^T \mathcal{L}_3(x(t), x_0, x_t), \tag{149}$$

where α is a suitably chosen positive constant describing the extent to which the stochastic path should follow the geometry of $\mathcal{B}$.

Minimum Deformation Path

In addition to following the geometry of the black box $\mathcal{B}$, one could also impose a natural condition that the stochastic paths minimize distances on the manifold in feature space that the auto-encoder pair induces. We recall from basic differential geometry that the image of the decoder as a subset of the feature space is a submanifold with a Riemannian metric g induced by the ambient Euclidean metric in the standard way (for background we refer to (do Carmo 1976)). In the simple case of a deterministic auto-encoder, one can think of g as the matrix $J^T J$ where J denotes the Jacobian of the decoder - thus g gives rise to scalar product $g(X, Y) := XJ^T JY$. In the stochastic case, one can use suitable approximations to obtain g in a similar manner - e.g. in (Arvanitidis et al. 2017) the authors decompose the decoder into a deterministic and a stochastic part, whose Jacobians J_1, J_2 are summed as $J_1^T J_1 + J_2^T J_2$ to obtain the matrix g. Now, having Riemannian structure (i.e. the notion of a distance) on the data submanifold, geodesic curves naturally arise as minimizers of a suitable distance functional, namely:

$$\mathcal{S}_4(x(t), x_0, x_t) := \int_0^T \|\dot{x}(t)\|_g dt, \tag{150}$$

where the norm $\|\cdot\|_g$ is computed with respect to the Riemannian metric g, that is $\sqrt{g(\cdot, \cdot)}$. We note that the utilization of geodesics for suitable latent space interpolations was thoroughly discussed in (Arvanitidis et al. 2017).

Other regularizers

Additionally we require $b(l_T, x(t))$ to be a monotonous function along the interpolating path $x(t)$. Furthermore, in accordance with Proposition 1 we require certain level of reconstruction at the end points. To enforce these conditions we introduce the regularizers r_m, r_e. The first, corresponds to

an criteria tha enforces that the interpolations do not interfere with the reconstruction at the ends, whereas r_e stands for explanatory examples. An explanatory path is a mapping $x : [0, T] \to \mathcal{X}$ with $x(0) = x_0, x(T) = x_T$ so that $\mathcal{B}(l_t|x(t))$ is a monotonous function. Explanatory paths are preferable in the sense that they provide examples following a particular trend along the disputed labels. The classifier must change decision in a monotonous fashion, as the interpolation proceeds the probability of classification of the l_0 decreases whereas the classification probability for l_T increases. These paths can be enforced in a straightforward way by introducing the constraint:

$$r_e = \frac{d}{dt}\mathcal{B}(l_T|x(t)) < 0, \quad \forall t \in [0, T]. \tag{151}$$

Note that this constraint requires differentiability of $x(t)$ - in contrast, the notion of explanatory path is not relying on such. We approximate the differential with finite differences. Our final loss reads:

$$L(\theta, \phi) := L_{\text{VAE}}(\theta, \phi) + \lambda \, \mathbb{E}_{dP[x(t)]} S_1[x(t)] + \lambda_m r_m + \lambda_e r_e, \tag{152}$$

where $\lambda, \lambda_m, \lambda_e$ are hyper-parameters and S_1 is the action associated to the minimum hesitant Lagrangian $\mathcal{L}_1$ in eq. (148).

7.7 Comparison to other models

Although the aim of our methodology is to provide representations which are embedded with explanatory power, traditional approaches rely on different philosophy to provide explanations. A saliency maps is a another image $S \in \mathbb{R}^{P \times P}$ where P is the number of pixels, which highlight important areas of the images that the black box classifier show in classification. We will use the interpolations as obtained by the auto encoders trained with our procedure to define a saliency maps. We will achieve this by creating weighed sum over the images in a given interpolation. Since the aim is to know the relevant characteristics that change the classification decision, it is natural to use the change in the probabilities as the corresponding weigth. Formally the Interpolation saliency Map is defined as:

$$S(x_0) = 1/T \int \delta B(x|x_0) \delta x dP[x(t)] = 1/T \int \{B(l_T|x(t)) - B(l_0|x(t))\} (x(t) - x_0) dP[x(t)]. \tag{153}$$

We obtained approximations o this integral by using a discrete approximation as performed in the calculation of the action Eq. 146

7.7.1 Evaluation

In order to quantitatively evaluate the effectiveness of our methodology, we use a masking procedure. For a given image x and its corresponding saliency map s, the masking is accomplished by changing the pixels of x which have a saliency value bigger than the τ percentile set of values of the map s itself. We then quantify the change in the odds probability, per number of pixel changed (in percentage values)

$$\log P(c|x) = \log P(c|x) - \log(1 - P(c|x)). \tag{154}$$

In short, a good saliency map, will achieve the biggest change in the log odds, with the least amount of pixel changed. It is to be noted that, we have to define the fill in new pixel procedure. We utilized for tree options. The minimum and higher value of a pixel (for a given image set), as well as a random uniform pixel value along this values. We show an example of the saliency maps obtained for one classification in Fig. 39. In order to obtain statistics of the procedure, we obtained the relevance values for 100 different black box missclassifications and provide the average in table 7.1. Here we compare against traditional saliency maps as well as variations of our own methodology for different Lagrangian's combinations. We find that the minimum transformation action applied

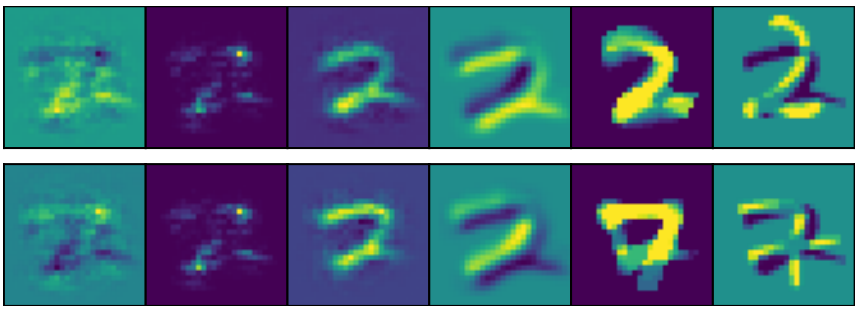


Figure 39 – Saliency Maps Comparison: Vanilla Gradients (Simonyan et al. 2013b), Smooth Gradients (Smilkov et al. 2017), Guided BackProp (Springenberg et al. 2014b), Grad CAMP (Selvaraju et al. 2016b), Interpolations, Difference with Representative. Upper row corresponds to $l_T = 2$, lower row to $l_T = 7$.

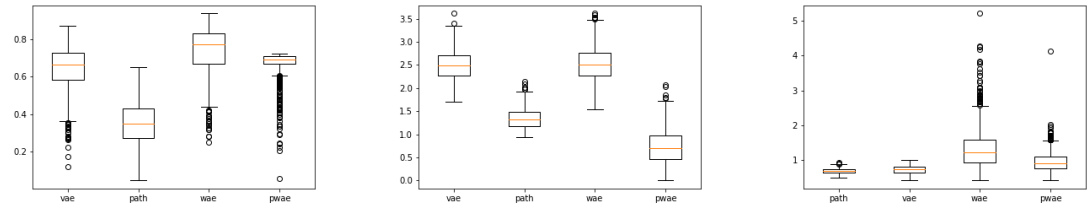
index	max	min	mean	random
VAE mhp (1.0) mdp (5.0)	26.890537	13.248348	13.780094	24.259376
VAE mdp (5.0)	17.528036	12.430488	13.249453	16.323000
VAE mtl (5.0)	22.425894	1.593850	17.865745	18.878264
VAE mhp (1.0) mtl (5.0)	41.968799	3.516279	25.474598	41.076384
WAE mdp (5.0)	1.348628	7.626165	4.758138	2.598853
WAE mhp (1.0) mdp (5.0)	28.650618	19.274864	13.260943	24.217626
WAE mtl (5.0)	33.308588	1.721710	10.395801	27.469413
WAE mhp (1.0) mtl (5.0)	21.113389	25.378131	6.343344	16.944631
vanilla	18.799356	12.129845	12.124648	18.617377
smooth	2.626274	16.802856	10.184854	3.966701
guided	25.264783	4.653241	2.523255	15.527908
mask	4.276590	0.248701	3.414551	3.244211

Table 7.1 – Relevance Statistics for Different Models and Comparison Saliency Maps. Here Minimum hesitant path (mhp), minimum transformation path (mtp), minimum deformation path (mdp). The numbers in parentesis indicate the value of the corresponding λ hyperparameters. Here Vanilla (Simonyan et al. 2013b), Smooth (Smilkov et al. 2017), guided (Springenberg et al. 2014b), mask (Selvaraju et al. 2016b)

to the Wasserstein autoeoncoder achieves better performance, as might be explained by its longest change along the path as opposed by the minimum hesitant.

7.8 Experimental results

We evaluate our method in two real-world data sets: MNIST, consisting of 70k Handwriting digits, (LeCun 1998) and the CelebA dataset (Liu et al. 2015) with roughly 203k images of celebrities faces. We use a vanilla version of the VAE (Kingma and Welling 2013a) with Euclidean latent spaces $\mathcal{Z} = \mathbb{R}^{d_z}$ and an isotropic Gaussian as a prior distribution $P(Z) = \mathcal{N}(Z|0, I_{d_z})$. We used Gaussian encoders, i.e. $Q_\phi(Z|X) = \mathcal{N}(Z|\mu_\phi(X), \Sigma_\phi(X))$, where μ_ϕ, σ_ϕ are approximated with neural networks of parameters ϕ, and Bernoulli decoders $P_\theta(X|Z)$. We compare the standard VAE, VAE-EDGE (VAE augmented with the edge loss r_e) and PATH-VAE (our full model, eq. (152)). The black-box classifier $b(l, x)$ is defined as a deep network with convolutional layers and a final soft-max output layer for the labels. Details of the specifics of the architectures as well as training procedure are left to the Appendix. For MNIST we studied a litigation case wherein $l_{-T}, l_T = 2, 7$ and $l_0 = 2$, whereas its true label (i.e. that of x_0) is $l_t = 7$ (see Section 7.3). The results are presented in Fig. 38. VAE delivers interpolations which provide uninformative examples, i.e. the changes in the output probability $b(l_0, x)$ *cannot* be associated with changes in feature space. In stark contrast, PATH-VAE causes the output probability to change abruptly. This fact, together with the corresponding generated examples, allows us to propose explanations of the form: what makes the black-box model classify an image in the path as *two* or *seven*, is the shifting up of the lower stroke in the digit *two* as to coincide with the middle bar of the digit *seven*. Similarly,



(a) Minimum Hesitant **(b)** Minimum Transformation **(c)** Fixed Lenght Scales

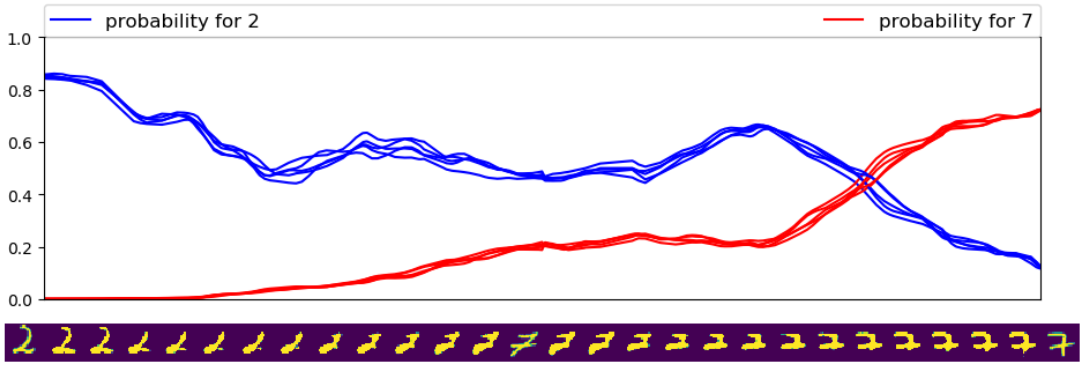
Figure 40 – Box Plot Statistics for different action values vs its un regularized auto encoder variance for different lagrangians

the upper bar of the digit *seven* (especially the upper left part) has a significant decision weight. Further comparison are presented in more detail in Fig. 41. In order to provide a more quantitative analysis we demonstrate the capability of our methodology to control the path action while retaining the reconstruction capacity. Hereby, we use not only the VAE as the underlying generative model, but also Wasserstein Auto-Encoder (WAE) (Bousquet et al. 2017). The theoretical details and corresponding architectures are presented in the Appendix. In order to validate our hability to control the action values during the training procedure, we present, in Fig. 40, the action values defined over random litigation end pairs (x_{-T}, x_T). The PATH version of the model indeed yields correspondign changes in the action values. Furthermore, these models tend to reduce the variance within the different paths. This is expected since there is one path that minimizes the action, hence, the distribution will try to arrive at this specific path for all samples.

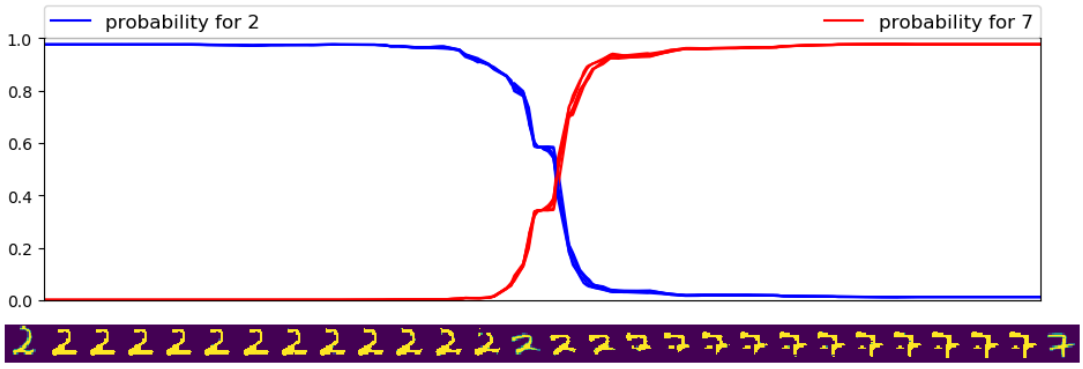
For the CelebA dataset we use a black box classifier base on the ResNet18 architecture (He et al. 2016). We investigate two specific misclassifications. In the first case, a smile was not detected (Fig. 42 a). Here we only interpolate between the misclassified image (left) and correctly classified one of the same person (right). Interpolations obtained by the VAE model are not informative: Specific changes in feature space corresponding to changes in the probability can not be detected since the latter changes rather slowly over the example path. This observation also holds for the VAE-EDGE model, except that the examples are sharper. Finally, our PATH-VAE model yields a sharp change in the probability along with a change of the visible teeth (compare the third and fifth picture in the example path), revealing that this feature constitutes one decisive factor in the probability of detecting a smile. The second case deals with a woman who was wrongly classified as a man (Fig. 42 b). We observe the same pattern as above, i.e., the VAE and the VAE-EDGE model do not reveal decisive features, since the probability along the path changes too slowly. In contrast, the probability in the PATH-VAE model features a sudden jump. Comparing the corresponding images (the eighth and the ninth image) suggests that the eyebrows might have a strong effect on the decision. It is important to note that these observations represent one of many possible path changes that could change the classifier decision; the current realization and representative endpoints constrain this. The important result is that our methods can shape the classifier's behavior along the path.

7.9 Discussion and Outlook

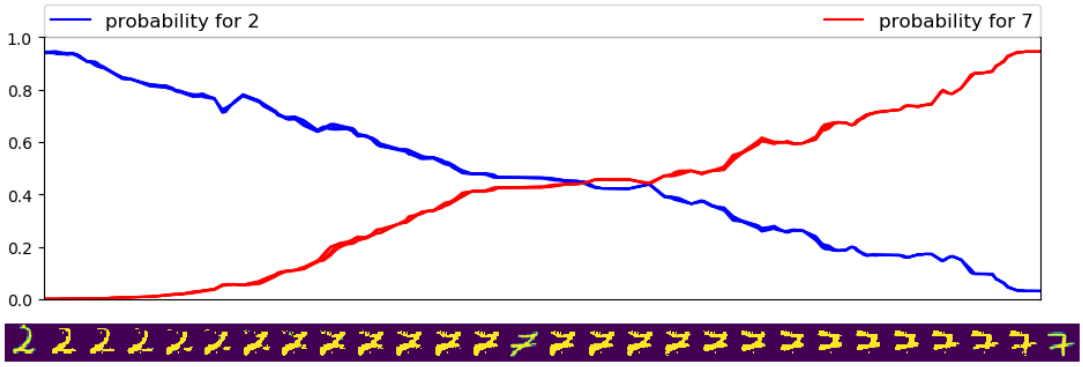
It is important to understand that our methodology is limited by the autoencoders' ability to provide proper samples in the data manifold. Although recent developments in the autoencoder community have vastly improved this capacity, one needs to remember that the datasets in which autoencoders are applicable are usually those already segmented within one *object class*. So autoencoders work correctly in datasets as MNSIT of CelebA, which are only comprised of digits or faces. Traditional methods of image explanations then work in larger datasets that do not provide enough regularity and are abundant in classes with low representation e.g. ImageNet (Deng et al. 2009). This does not stop the autoencoders for providing meaningful representations for say, classification, and variations



(a) Regular VAE



(b) Minimum Hesitant $\mathcal{L}_1$

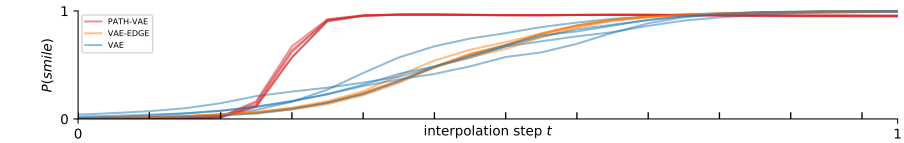


(c) Minimum Deformation $\mathcal{L}_3$

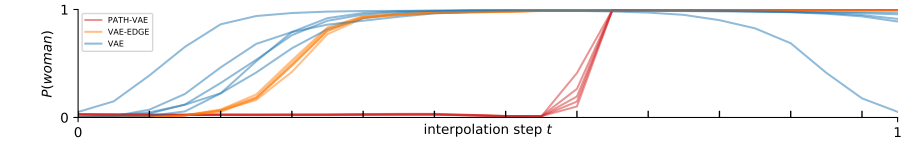
Figure 41 – Probability Paths for a litigation case ($l_0 = 2$,$l_T = 7$). Y axis corresponds to classification probability for the different labels and x axis corresponds to interpolation index. Interpolation images for a specific paths are presented below the x axis. The final and initial image corresponds to the plaintiff class representatives X_{-T} and X_T. The center image corresponds to the plaintiff x_0

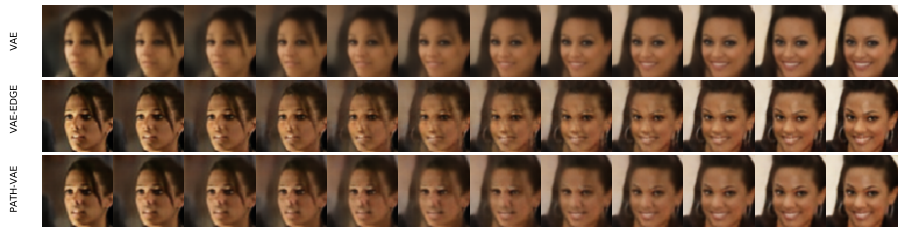
of our methodology using statistics of classifications on the interpolated code can also be attained.

In summary, in the present chapter, we provide a novel framework to explain black box classifiers through examples obtained from deep generative models. To summarize, our formalism extends the autoencoder framework by focusing on the interpolation paths in feature space. We train the autoencoder, not only by guaranteeing reconstruction quality but by imposing conditions on its interpolations. These conditions are such that information about the model's classification decisions $\mathcal{B}$ is encoded in the example paths. Beyond the specific problem of generating explanatory examples, our work formalizes the notion of stochastic process induced in feature space by latent code interpolations. It provides quantitative characterization of the interpolation through the



(a) Smile





(b) Sex

Figure 42 – Probability Paths for the case of detecting a smile (a) or the sex (b) in images of celebrities. Y axis corresponds to classification probability and x axis corresponds to interpolation index. Interpolation images for a specific paths are presented below the x axis. The images are vertically aligned with a corresponding tick in the x-axis determining the interpolation index of the image

semantic Lagrangian's and actions. Our methodology is not constrained to a specific Auto Encoder framework provided that mild regularity conditions are guaranteed for the autoencoder and decoder.

Part IV

Appendix

Chapter 8

Proof Concerning Regularity of Paths

8.1 Stochastic Semantic Processes: Proof of Proposition 1

Briefly put, the construction we utilize makes use of the well-known notion of consistent measures, which are finite-dimensional projections that enjoy certain restriction compatibility; afterwards, we show existence by employing the central extension result of Kolmogorov-Daniell.

8.1.1 Collections of Consistent Measures

We start with a couple of notational remarks.

Definition 1 *Let S, F be two arbitrary sets. We denote*

$$S^F := \{f : F \to S\}, \tag{155}$$

that is, the set of all maps $F \to S$.

Definition 2 *Let $(S, \mathcal{B})$ be a measurable space and let $G \subseteq F \subseteq [0, T]$ for some positive number T. We define the restriction projections $\pi_{F,G}$ by*

$$\pi_{F,G} : S^F \to S^G, \quad f \in S^F \mapsto f|_G \in S^G. \tag{156}$$

Moreover, for each $F \subseteq [0, T]$ the restriction projections induce the σ-algebra $\mathcal{B}^F$ which is the smallest σ-algebra on S^F so that all projections

$$\pi_{F,\{t\}} : S^F \to S^{\{t\}} \cong S, \quad \forall t \in F, \tag{157}$$

are measurable. In particular, the projections $\pi_{F,G}$ are measurable with respect to $\mathcal{B}^F, \mathcal{B}^G$.

Definition 3 *Let us denote by $\mathrm{Fin}([0, T])$ the set of all finite-element subsets of $[0, T]$. A collection of finite measures $\{(\mu_F, \mathcal{B}^F), F \in \mathrm{Fin}([0, T])\}$ is called consistent if it is push-forward compatible with respect to the restriction projection mappings, i.e.*

$$(\pi_{F,G})_* \mu_F = \mu_G, \quad \forall F, G \in \mathrm{Fin}([0, T]), G \subseteq F. \tag{158}$$

Here

$$(\pi_{F,G})_* \mu_F(A) := \mu_F(\pi_{F,G}^{-1}(A)), \quad \forall A \in \mathcal{B}^G. \tag{159}$$

Proposition 2 *Let $F = \{0 \leq t_1 < t_2 < \cdots < t_n \leq T\} \in \mathrm{Fin}([0, T])$ be an arbitrary finite set. The mapping*

$$\mu_F(A) := \int \chi_A(x_1, x_2, \ldots, x_n) \left(\prod_{i=1}^{n} p_\theta(x_i|z_i) \right) q_\phi(z_0|x_0) q_\phi(z_T|x_T) dz_0 dz_T dx_1 \ldots dx_n \tag{160}$$

defines a consistent collection of finite measures.

Let us fix

$$F_1 := \{0 \leq t_1 < t_2 < \cdots < t_n \leq T\} \in \mathrm{Fin}([0,T]), \tag{161}$$

$$F_2 := \{0 \leq t_1 < t_* < t_2 < \cdots < t_n \leq T\} \in \mathrm{Fin}([0,T]), \tag{162}$$

Without loss of generality, it suffices to check consistency for the pair (F_1, F_2). We have

$$(\pi_{F_1,F_2})_* \mu_{F_2}(A) = \mu_{F_2}\left(\pi^{-1}_{F_1,F_2}(A)\right) \tag{163}$$

$$= \int \chi_{\pi^{-1}_{F_1,F_2}(A)}(x_1, s, x_2, \ldots, x_n) \left(\prod_{i=1}^{n} p_\theta(x_i|z_i)\right) \tag{164}$$

$$\times\, p_\theta(s|z_{t_*}) q_\phi(z_0|x_0) q_\phi(z_T|x_T) ds dz_0 dz_T dx_1 dx_2 \ldots dx_n \tag{165}$$

$$= \int \chi_A(x_1, x_2, \ldots, x_n) \left(\prod_{i=1}^{n} p_\theta(x_i|z_i)\right) q_\phi(z_0|x_0) q_\phi(z_T|x_T) dz_0 dz_T dx_1 \ldots dx_n \tag{166}$$

$$= \mu_{F_1}(A), \tag{167}$$

where we have used L^1-finiteness and integrated out the s variable via Fubini's theorem. Note also, that by the definitions above

$$\chi_{\pi^{-1}_{F_1,F_2}(A)}(x_1, s, x_2, \ldots, x_n) = \chi_A(x_1, x_2, \ldots, x_n). \tag{168}$$

for any fixed $s \in \mathcal{X}$.

We briefly recall the following classical result due to Kolmogorov and Daniell:

Theorem 1 (Theorem 2.11, (Bär and Pfäffle 2012)) *Let $(S, \mathcal{B}(S))$ be a measurable space with S being compact and metrizable and let I be an index set. Assume that for each $J \in \mathrm{Fin}(I)$ there exists a measure μ^J on $S^J, \mathcal{B}^J$, such that the following compatibility conditions hold:*

$$\mu^{J_1} = \mu^{J_2} \circ \pi^{-1}_{J_1}, \quad \forall J_1 \subseteq J_2 \in \mathrm{Fin}(I). \tag{169}$$

Here $\pi_{J_1} : S^{J_2} \to S^{J_1}$ denotes the canonical projection (obtained by restriction).

Then, there exists a unique measure μ on $(S^I, \mathcal{B}^I)$ such that for all $J \in \mathrm{Fin}(I)$ one has

$$\mu \circ \pi_J^{-1} = \mu^J. \tag{170}$$

We recall that a well-known way to construct the classical Wiener measure and Brownian motion is precisely via the aid of Theorem 1 ((Taylor 2011)). We are now in a position to construct the following stochastic process.

Proposition 3 *There exists a continuous-time stochastic process $X_t : [0,T] \to \mathbb{R}^D$ satisfying*

$$\mathbb{P}((X_{t_1}, X_{t_2}, \ldots, X_{t_n}) \in A) = \int \chi_A(x_1, x_2, \ldots, x_n) \tag{171}$$

$$\times \left(\prod_{i=1}^{n} p_\theta(x_i|z_i)\right) q_\phi(z_0|x_0) q_\phi(z_T|x_T) dx_1 \ldots dx_n. \tag{172}$$

$$\tag{173}$$

Moreover, for small positive numbers ϵ, δ we have $X_0 \in B_\delta(x_0)$ with probability at least $(1-\epsilon)$, provided the reconstruction error of encoding/decoding process is sufficiently small. In particular, if x_0 stays fixed after the application of encoder followed by decoder, then $X_0 = x_0$ almost surely. A similar statement holds also for the terminal point X_t and x_T respectively.

By applying Theorem 1 to the collection of consistent finite measures prescribed by Proposition 2

we obtain a measure μ on the measurable space $(S^{[0,T]}, \mathcal{B}^{[0,T]})$. Considering the probability space $(S^{[0,T]}, \mathcal{B}^{[0,T]}, \mu)$ we define stochastic process

$$X_t := \pi_{[0,T],\{t\}} : S^{[0,T]} \to S. \tag{174}$$

It follows from the construction and the Theorem of Kolmogorov-Daniell that $\mathbb{P}\left((X_{t_1}, X_{t_2}, \ldots, X_{t_n}) \in A\right)$ is expressed in the required way. This shows the first claim of the statement.

Now, considering a small ball $B_\delta(x_0)$ we have

$$\mathbb{P}(X_0 \in B_\delta(x_0)) = \int \chi_{B_\delta(x_0)}(x) p_\theta(x|z_0) q_\phi(z_0|x_0) q_\phi(z_T|x_T) dx dz_0 dz_T \tag{175}$$

$$= \int \chi_{B_\delta(x_0)}(x) p_\theta(x|z_0) q_\phi(z_0|x_0) dx dz_0 \tag{176}$$

$$:= R(x_0, \chi_{B_\delta(x_0)}). \tag{177}$$

Here, the function $R(x^*, U)$ measures the probability that the input x^* is decoded in the set U. Thus, if the reconstruction error gets smaller, R converges to 1. This implies the second statement.

Finally, if we assume that the auto-encoder fixes x_0 in the sense above, we similarly get

$$\mathbb{P}(X_0 = x_0) = \int \chi_{\{x_0\}}(x) p_\theta(x|z_0) q_\phi(z_0|x_0) q_\phi(z_T|x_T) dx dz_0 dz_T \tag{178}$$

$$= \int \chi_{\{x_0\}}(x) p_\theta(x|z_0) q_\phi(z_0|x_0) dx dz_0 \tag{179}$$

$$= \delta_{x_0}(\chi_{\{x_0\}}) \tag{180}$$

$$= 1. \tag{181}$$

8.1.2 Concerning the Regularity of Sample Paths

An important remark related to the the variational problem (142) is the following: one could develop plenty of meaningful functionals $\mathcal{S}_{P_\theta, Q_\phi}$ that involve taking velocities or higher derivatives - thus one is supposed to work over spaces of curves with certain regularity assumptions. However, as stated above we are working over stochastic paths X_t whose regularity is, in general, difficult to guarantee. A straightforward way to alleviate this issue is to consider a "smooth" version of the curve X_t - e.g. by sampling X_t through a decoder with controllable or negligible variance or by means of an appropriate smoothing. Furthermore, one could also approach such stochastic variational analysis via Malliavin calculus - however, we do not pursue this direction in the present work.

We now briefly discuss a few remarks about the regularity of the stochastic semantic process from Proposition 1. First, we state a well-known result of Kolmogorov and Chentsov:

Theorem 2 (Theorem 2.17, (Bär and Pfäffle 2012)) *Let (M, ρ) be a metric measure space and let $X_t, t \in [0,T]$ be a stochastic process. Suppose that there exists positive numbers a, b, C, ϵ with the property*

$$\mathbb{E}\left[\rho(X_s, X_t)^a\right] \leq C|t-s|^{(1+b)}, \quad \forall s, t, |s-t| < \epsilon \tag{182}$$

Then, there exists a version $Y_t, t \in [0,T]$ of the stochastic process X_t whose paths are α-Hölder continuous for any $\alpha \in (0, b/a)$.

Thus, roughly speaking, an estimate on $\mathbb{E}\left[\rho(X_s, X_t)^a\right]$ can be regarded as a measure of the extent to which Theorem 2 fails. To give an intuitive perspective, let us consider the stochastic process given by Proposition 1 and, considering only the points $X_s, X_{s+\delta}$ for a small positive number δ, let

us write the expectation in (182) as:

$$\int\int\int\int \|x_{s+\delta} - x_s\|\, p_\theta(x_{s+\delta}|z_{s+\delta}) p_\theta(x_s|z_s) q_\phi(z_0|x_0) q_\phi(z_T|x_T) dx_s dx_t dz_0 dz_T, \tag{183}$$

where we have used the standard Euclidean distance. To estimate the integral further, let us for simplicity assume that the encoder is deterministic and the decoder is defined via a Gaussian Ansatz of the type $\mu(z) + \sigma(z) \otimes \epsilon$ for a normal Gaussian variable ϵ. Thus the last integral can be written as:

$$\int\int \frac{\|x_{s+\delta} - x_s\|}{(2\pi)^n \sqrt{|\Sigma_{s+\delta}||\Sigma_s|}} \exp\Bigg(-\frac{1}{2}[(x_{s+\delta} - \mu(z_{s+\delta}))^T \Sigma_{s+\delta}^{-1}(x_{s+\delta} - \mu(z_{s+\delta})) \tag{184}$$

$$+ (x_s - \mu(z_s))^T \Sigma_s^{-1}(x_s - \mu(z_s))] \Bigg) dx_s dx_t, \tag{185}$$

where we denote the covariance matrix at time s by Σ_s. Now, if $\Sigma_{s+\delta}$ becomes sufficiently small as δ converges to 0, then the exponential factor will dominate and thus (182) holds. In other words, Hölder regularity of the process is verified provided that $p_\theta(x|z)$ becomes localized in x and converges to a Dirac measure (similarly to the case of the heat kernel propagator and Brownian motion). From this point of view, the variance of the decoder can be considered as an indicator of how far the stochastic process is from being Hölder continuous.

Below we discuss two other stochastic process constructions, one of which is built upon Itô diffusion processes and enjoys further path-regularity properties.

8.1.3 Stochastic Semantic Processes: Further Constructions

In this Subsection we briefly provide a couple of additional methods as to how one can construct various types of stochastic semantic processes - in an upcoming work we investigate these in further detail.

As a first suggestion, we recall that a stochastic process can be induced via a specific transition kernel $\kappa_t(x, y)$ which, roughly speaking, prescribes the probability that at time t the process will jump from x to y (e.g. Brownian motion is induced by the heat kernel $h_t(x, y)$). Now, using the auto-encoder pair one can come up with various transition kernels. A natural suggestion is:

$$\kappa_t(x_0, x) := \int\int p_\theta\left(x|tz_0 + (1-t)z\right) q_\phi(z|x) q_\phi(z_0|x_0) dz_0 dz. \tag{186}$$

Aside from transition kernels, one can also propose a construction in the spirit of Brownian bridges. To set some notation, we recall that the decoding map is often decomposed ((Kingma and Welling 2013a)) by means of the following Ansatz:

$$P(z) = \mu_\theta(z) + \sigma_\theta(z) \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbb{I}_D), \tag{187}$$

where the symbol $\cdot$ denotes element-wise multiplication. Roughly, the decoder is decomposed into a deterministic $\mu_\theta(z)$ and a stochastic $\sigma_\theta(z)$ part. We introduce the following Itô process:

$$dX_t = D\mu_\theta(\dot{z}(t))dt + \eta(\sigma_\theta(z(t)))dB_t, \tag{188}$$

where $D\mu_\theta$ denotes the differential of μ_θ; $\dot{z}(t)$ is a smooth semantic interpolation; $\eta : X \to \mathbb{R}$ is a smooth function estimating the effect of the variance and B_t denotes the standard Wiener process. For instance, reasonable choices of η include powers of the Euclidean distance. Under appropriate assumptions on the auto-encoding mappings P, Q (in particular, $\mu_\theta, \sigma_\theta$), the Itô process (188) defines a stochastic process X_t starting and terminating respectively at x_0, x_T and possessing regular sample paths. Moreover, the mean of X_t is given by $\mu_\theta(z(t))$ and the variance is estimated

in terms of $\eta(\sigma_\theta(z))$.

Chapter 9

Deep Neural Networks Architectures

Long Short Term Memory Network

One of the most common variants of recurrent neural networks, aimed at solving the vanishing gradient problem, is the long short-term memory network (Hochreiter and Schmidhuber 1997). Let us define $\hat{\mathbf{w}}$ as the observed data[1]. The LSTM network recursively processes each element $\hat{\mathbf{w}}_i^j$ in review j while updating its hidden state $\mathbf{s}^j$ as follows

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i^1\,\hat{\mathbf{w}}_t + \mathbf{W}_i^2\,\mathbf{s}_{t-1} + \mathbf{b}_i),\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o^1\,\hat{\mathbf{w}}_t + \mathbf{W}_o^2\,\mathbf{s}_{t-1} + \mathbf{b}_o),\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f^1\,\hat{\mathbf{w}}_t + \mathbf{W}_f^2\,\mathbf{s}_{t-1} + \mathbf{b}_f),\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + i_t \odot \tanh(\mathbf{W}_c^1\,\hat{\mathbf{w}}_t + \mathbf{W}_c^2\,\mathbf{s}_{t-1} + \mathbf{b}_c),\\
\mathbf{s}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned} \tag{189}$$

Here $\mathbf{i}_t$, $\mathbf{o}_t$ $\mathbf{f}_t$, $\mathbf{c}_t$, and $\mathbf{s}_t$ are the input, output, forget, memory and hidden states of the LSTM, respectively; "t" runs from one to L, the number of words in the jth review (note we have omited j above); σ labels the ReLU nonlinearity and $\odot$ labels element-wise multiplication.

9.1 Models Training Details

9.1.1 A simple VAE model: MNIST

There was no preprocessing on the 28x28 MNIST images. The models were trained with up to 25 epochs with mini-batches of size 1024. Our choice of optimizer is Adam with learning rate $\alpha = 10^{-3}$. The weight of the KL term of the VAE is $\lambda_{kl} = 1$, the *path* loss weight is $\lambda_p = 10^{-3}$ and the *edge* loss weight is $\lambda_e = 10^{-3}$. We estimate the *path* and *edge* loss during training by sampling 100 paths, each of those has 20 steps.

Encoder Architecture

$$\begin{aligned}
x \in \mathbb{R}^{28\times 28\times 1} &\to \mathrm{FC}_{400} \to \mathrm{ReLU}\\
&\hookrightarrow \mathrm{FC}_{20} \to \mu \in \mathbb{R}^{20}\\
&\hookrightarrow \mathrm{FC}_{20} \to \sigma \in \mathbb{R}^{20}\\
\mu \in \mathbb{R}^{20}, \sigma \in \mathbb{R}^{20} &\to z \sim \mathcal{N}(z; \mu, \sigma)
\end{aligned}$$

[1] traditionally LSTMs are used for the pourpose of text analysis, and w corresponds to one word in the context of natural language processing

Decoder Architecture

$$z \in \mathbb{R}^{20} \to \text{FC}_{400} \to \text{ReLU} \\ \to \text{FC}_{28\times28\times1} \to \text{Sigmoid}$$

Here FC_k stands for the fully connected layer to $\mathbb{R}^k$ and ReLU for the rectified linear units. The output of the first the rectifier linear unit is passed to two independent fully connected layers. The latent code z is sampled from a normal distribution.

9.1.2 A Gaussian CNN Encoder and CNN Decoder: MNIST and CelebA

Similarly, there was no preprocessing on the 28x28 MNIST images. The models were trained with up to 100 epochs with mini-batches of size 32. Our choice of optimizer is Adam with learning rate $\alpha = 10^{-3}$. The weight of the KL term of the VAE is $\lambda_{kl} = 1$, the *path* loss weight is $\lambda_p = 10^3$ and the *edge* loss weight is $\lambda_e = 10^{-1}$. We estimate the *path* and *edge* loss during training by sampling 5 paths, each of those has 20 steps.

Encoder Architecture

$$x \in \mathbb{R}^{28\times28\times3} \to \text{Conv}_{64} \to \text{BN} \to \text{ReLU} \\ \to \text{Conv}_{128} \to \text{BN} \to \text{ReLU} \\ \to \text{Conv}_{256} \to \text{BN} \to \text{ReLU} \to \text{FC}_8$$

Decoder Architecture

$$z \in \mathbb{R}^8 \to \text{FC}_{4\times4\times512} \\ \to \text{FSConv}_{256} \to \text{BN} \to \text{ReLU} \\ \to \text{FSConv}_{128} \to \text{BN} \to \text{ReLU} \\ \to \text{FSConv}_{64} \to \text{Sigmoid}$$

Both the encoder and decoder used fully convolutional architectures with 3x3 convolutional filters with stride 2. Conv_k denotes the convolution with k filters, FSConv_k the fractional strides convolution with k filters (the first two of them doubling the resolution, the third one keeping it constant), BN denotes batch normalization (Ioffe and Szegedy 2015), and as above ReLU the rectified linear units, FC_k the fully connected layer to $\mathbb{R}^k$.

The pre-processing of the CelebA images was done by first taking a 140x140 center crop and then resizing the image to 64x64. The models are trained with up to 100 epochs and with mini-batches of size 128. Our choice of optimizer is Adam with learning rate $\alpha = 10^{-3}$. The weight of the KL term of the VAE is $\lambda_{kl} = 0.5$, the *path* loss weight is $\lambda_p = 0.5$ and the *edge* loss weight is $\lambda_e = 10^-3$. We estimate the *path* and *edge* loss during training by sampling 10 paths, each of those has 10 steps.

Encoder Architecture

$$x \in \mathbb{R}^{64\times64\times3} \to \text{Conv}_{64} \to \text{BN} \to \text{ReLU} \\ \to \text{Conv}_{128} \to \text{BN} \to \text{ReLU} \\ \to \text{Conv}_{256} \to \text{BN} \to \text{ReLU} \\ \to \text{Conv}_{512} \to \text{BN} \to \text{ReLU} \to \text{FC}_{500}$$

Decoder Architecture

$$
\begin{aligned}
z \in \mathbb{R}^{500} &\rightarrow \mathrm{FC}_{4\times4\times512} \\
&\rightarrow \mathrm{FSConv}_{256} \rightarrow \mathrm{BN} \rightarrow \mathrm{ReLU} \\
&\rightarrow \mathrm{FSConv}_{128} \rightarrow \mathrm{BN} \rightarrow \mathrm{ReLU} \\
&\rightarrow \mathrm{FSConv}_{64} \rightarrow \mathrm{Sigmoid}
\end{aligned}
$$

Both the encoder and decoder used fully convolutional architectures with 3x3 convolutional filters with stride 2. Conv_k denotes the convolution with k filters, FSConv_k the fractional strides convolution with k filters (the first two of them doubling the resolution, the third one keeping it constant), BN denotes batch normalization, and as above ReLU the rectified linear units, FC_k the fully connected layer to $\mathbb{R}^k$.

Chapter 10

Relations to formal analysis of queuing systems

A common practice to study queuing systems is through parametric analysis – assuming the systems obeys certain distribution laws one usually attempts to explicitly estimate moments of the quantities of interest. Our construction assumes no parametric forms and thus does not allow for standard theoretical treatment of moments, etc. However, analyzing the model itself in terms of the neural networks involved could lead to interesting relations with several theoretical results in the subject. We illustrate a straight-forward estimate along the lines of the classical Little's law (Little 1961) specifying the average number of clients in the system.

First, we define the counting process

$$\{N(t)|t \geq 0, N(t) = \max\{i : a_i \leq t\}\},$$

the number of arrivals during $(0, t]$. Let $\{N^d(t)|t \geq 0\}$ denote the counting process for the departure times $\{d_i\}$, with $N^d(t)$ the number of customers who have departed by time t. Note that $N^d(t) \leq N(t), t \geq 0$. Finally, we define $L(t)$, the total number of customers in the system at time t.

Definition 4 *Let us assume that the following limits exist:*

$$\lambda = \lim_{t\to\infty} \frac{N(t)}{t}, \quad r = \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} r_j, \quad l = \lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)ds, \tag{190}$$

which we refer to as the system's arrival rate, the average sojourn time and the average number of customers, respectively.

The classical form of the celebrated Little's law states that

Theorem 3 *Assume that λ, r exist and are finite. Then l exists and the following relationship holds:*

$$l = \lambda r. \tag{191}$$

Recall that in our approach we model the service times by using a neural network, i.e. by setting:

$$\tilde{s}_i := \theta^k \circ \sigma^k \circ \theta^{k-1} \ldots \sigma^2 \circ \theta^1(\mathbf{h}_i), \tag{192}$$

where $\mathbf{h}_i := \text{concat}(\mathbf{h}_i^a, \boldsymbol{\epsilon})$, with $\mathbf{h}_i^a$ the hidden representation of the RPP model (see main text), and $\boldsymbol{\epsilon} \in \mathbb{R}^h$ with its elements sampled from P_ϵ, an isotropic Gaussian; σ^j denotes a non-linearity and θ^j denotes the application of the weights $\widehat{\theta}^j$ (and bias b^j) of the jth layer of the neural network.

Lemma 2 *Let us assume that $\sigma^j = \text{ReLU}$ for each layer j and let us assume that the operator norms $\|\hat{\theta}^j\|, \|b^j\|$ are bounded by a positive constant M for each j. Then*

$$|\widetilde{s}_i| \leq M^k \|\mathbf{h}_i\| + \frac{M^{k+1} - 1}{M - 1}. \tag{193}$$

The estimate follows from a straightforward application of the triangle inequality and induction.

The estimate (193) can be made sharper provided subtler assumptions on the linear layers $\widehat{\theta^j}$ and the bias vectors b^j are assumed. It could, moreover, be replaced by an estimate on the Lipschitz constant of the neural network - for further applications to spectral bias and Fourier transforms, cf. (Rahaman et al. 2019).

Lemma 3 *Suppose that the service time model above fits the data within a mean average error of ε. Then the average service time is bounded above as follows*

$$\langle \widetilde{s}_i \rangle \leq M^k \limsup_i \|\mathbf{h}_i\| + \frac{M^{k+1}-1}{M-1} + \varepsilon. \tag{194}$$

We easily obtain

$$\langle \widetilde{s}_i \rangle = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} s_i = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} |s_i - \widetilde{s}_i + \widetilde{s}_i| \tag{195}$$

$$\leq \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} |s_i - \widetilde{s}_i| + \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \widetilde{s}_i \tag{196}$$

$$\leq \varepsilon + M^k \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{h}_i\| + \frac{M^{k+1}-1}{M-1} \tag{197}$$

$$\leq \varepsilon + M^k \limsup_i \|\mathbf{h}_i\| + \frac{M^{k+1}-1}{M-1}, \tag{198}$$

where we have used the estimate (193).

Corollary The average number of clients/customers in the system, l, is bounded above as follows

$$l \leq \lambda \left(M^k \limsup_i \|\mathbf{h}_i\| + \frac{M^{k+1}-1}{M-1} + \varepsilon \right), \tag{199}$$

where $\lambda, M, \mathbf{h}_i$ and ε are as above.

The statement follows directly by combining Little's law and the bound (194).

We remark that one could also estimate λ from above by a similar procedure in terms of the neural network that models the arrival process, that is, our RPP model.

Bibliography

D. P. Kingma and M. Welling, arXiv preprint arXiv:1312.6114 (2013a).

Y. Ogata, Journal of the American Statistical association **83**, 9 (1988).

Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, in *Proc. KDD* (2015a).

S. W. Linderman and R. P. Adams, arXiv preprint arXiv:1507.03228 (2015).

A. G. Hawkes and D. Oakes, Journal of Applied Probability **11**, 493 (1974).

Y. Lee, K. W. Lim, and C. S. Ong, in *International Conference on Machine Learning* (2016), pp. 79–88.

Y.-L. K. Samo and S. Roberts, in *International Conference on Machine Learning* (2015a), pp. 2227–2236.

C. Lloyd, T. Gunter, M. Osborne, and S. Roberts, in *International Conference on Machine Learning* (2015), pp. 1814–1822.

C. Archambeau, D. Cornford, M. Opper, and J. Shawe-Taylor, in *Gaussian Processes in Practice* (2007), pp. 1–16.

C. E. Rasmussen and C. K. Williams, *Gaussian process for machine learning* (MIT press, 2006).

N. G. Polson, J. G. Scott, and J. Windle, Journal of the American statistical Association **108**, 1339 (2013).

S. Linderman, M. Johnson, and R. P. Adams, in *Advances in Neural Information Processing Systems* (2015), pp. 3456–3464.

R. G. Gallager, *Discrete stochastic processes*, vol. 321 (Springer Science & Business Media, 2012).

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

J. F. C. Kingman, *Poisson processes* (Wiley Online Library, 1993).

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

J. G. Rasmussen, Methodology and Computing in Applied Probability **15**, 623 (2013).

H. W. Watson and F. Galton, The Journal of the Anthropological Institute of Great Britain and Ireland **4**, 138 (1875).

J. Møller and J. G. Rasmussen, Advances in applied probability **37**, 629 (2005).

R. P. Adams, I. Murray, and D. J. MacKay, in *Proceedings of the 26th Annual International Conference on Machine Learning* (ACM, 2009), pp. 9–16.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, Machine learning **37**, 183 (1999).

C. M. Bishop, Machine Learning (2006).

L. Csató, Ph.D. thesis, Aston University (2002).

M. Titsias, in *Artificial Intelligence and Statistics* (2009), pp. 567–574.

S. Geman and D. Geman, IEEE Transactions on pattern analysis and machine intelligence pp. 721–741 (1984).

S. Nakamoto (2008).

O. Stetter, D. Battaglia, J. Soriano, and T. Geisel, PLoS computational biology **8**, e1002653 (2012).

N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016a), pp. 1555–1564.

X. Xuan and K. Murphy, in *Proceedings of the 24th international conference on Machine learning* (ACM, 2007), pp. 1055–1062.

S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert, International Journal of Computer Vision **77**, 103 (2008).

Y. Saatçi, R. D. Turner, and C. E. Rasmussen, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Citeseer, 2010), pp. 927–934.

S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, in *Artificial Intelligence and Statistics* (2017), pp. 914–922.

E. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, in *Advances in Neural Information Processing Systems* (2009), pp. 457–464.

F. Stimberg, A. Ruttor, and M. Opper, in *Artificial Intelligence and Statistics* (2012), pp. 1117–1124.

D. P. Kingma and M. Welling, arXiv preprint arXiv:1312.6114 (2013b).

J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015), pp. 2980–2988.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio, in *AAAI* (2017), pp. 3295–3301.

P. Becker-Ehmck, J. Peters, and P. Van Der Smagt, in *Proceedings of the 36th International Conference on Machine Learning*, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, Long Beach, California, USA, 2019), vol. 97 of *Proceedings of Machine Learning Research*, pp. 553–562.

G. Ackerson and K.-S. Fu, IEEE Transactionson Automatic Control **15**, 10 (1970).

S. Hochreiter and J. Schmidhuber, Neural computation **9**, 1735 (1997).

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2016), pp. 1480–1489.

D. Bahdanau, K. Cho, and Y. Bengio, arXiv preprint arXiv:1409.0473 (2014).

P. Schwab, D. Miladinovic, and W. Karlen, arXiv preprint arXiv:1802.02195 (2018).

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, ICLR-2017 (2017).

E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, arXiv preprintarXiv:1511.06297 (2015).

S. Linderman, https://github.com/slinderman/recurrent-slds.

D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).

E. N. Lorenz, Journal of Atmospheric Sciences **20**, 130 (1963).

M. Liwicki and H. Bunke, in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)* (IEEE, 2005), pp. 956–961.

S. Bai, J. Z. Kolter, and V. Koltun, arXiv preprint arXiv:1803.01271 (2018).

C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, in *Proceedings of the tenth ACM international conference on web search and data mining* (ACM, 2017), pp. 495–503.

H. Jing and A. J. Smola, in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (ACM, 2017), pp. 515–524.

T. Zhou, H. Qian, Z. Shen, C. Zhang, C. Wang, S. Liu, and W. Ou, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (AAAI Press, 2018), pp. 3704–3710.

H. Mei and J. M. Eisner, in *Advances in Neural Information Processing Systems* (2017a), pp. 6754–6764.

N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016b), pp. 1555–1564.

S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha, in *Advances in Neural Information Processing Systems* (2017), pp. 3247–3257.

A. Daw and J. Pender, Stochastic Systems **8**, 192 (2018).

S. Asmussen, *Applied probability and queues*, vol. 51 (Springer Science & Business Media, 2008).

C. Sutton and M. I. Jordan, The Annals of Applied Statistics pp. 254–282 (2011).

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, in *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Curran Associates, Inc., 2014a), pp. 2672–2680.

R. J. Williams, Annual Review of Statistics and Its Application **3**, 323 (2016).

D. Bertsimas, Operations Research **38**, 139 (1990).

O. Boxma, O. Kella, and M. Mandjes, Queueing Systems **92**, 233 (2019).

P. Chapfuwa, C. Tao, C. Li, C. Page, B. Goldstein, L. Carin, and R. Henao, in *ICML* (2018).

M. Chambers and C. Mount-Campbell, International Journal of Production Economics **79**, 93 (2002).

D. G. Kendall, Ann. Math. Statist. **24**, 338 (1953).

L. Zhu, Journal of Applied Probability **50**, 760 (2013).

H. Mei and J. M. Eisner, in *Advances in Neural Information Processing Systems* (2017b), pp. 6738–6748.

N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016c), pp. 1555–1564.

A. Graves, arXiv preprint arXiv:1308.0850 (2013).

M. Arjovsky, S. Chintala, and L. Bottou, in *Proceedings of the 34th International Conference on Machine Learning*, edited by D. Precup and Y. W. Teh (PMLR, International Convention Centre, Sydney, Australia, 2017), vol. 70 of *Proceedings of Machine Learning Research*, pp. 214–223.

C. Villani, *Optimal Transport: Old and New* (Springer, 2009).

H. Petzka, A. Fischer, and D. Lukovnikov, in *International Conference on Learning Representations* (2018).

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, in *NIPS* (2017).

O. Mogren, CoRR **abs/1611.09904** (2016).

S. Hyland, C. Esteban, and G. Rätsch, *Real-valued (medical) time series generation with recurrent conditional GANs* (2018).

A. ElBahrawy, L. Alessandretti, A. Kandler, R. Pastor-Satorras, and A. Baronchelli, Royal Society open science **4**, 170623 (2017).

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, arXiv preprint arXiv:1409.1259 (2014).

S. Petrovic, M. Osborne, and V. Lavrenko, Proc. ICWSM (2011).

S. Gao, J. Ma, and Z. Chen, in *Proc. WSDM* (2015).

Z. Dezsö, E. Almaas, A. Lukács, B. Rácz, I. Szakadát, and A.-L. Barabási, Physical Review E **73** (2006).

F. Wu and B. A. Huberman, PNAS **104** (2007).

J. Yang and J. Leskovec, in *Proc. WSDM* (2011).

E. Cunha, G. Magno, G. Comarela, V. Almeida, M. A. Gonçalves, and F. Benevenuto, in *Proc. Workshop on Languages in Social Media* (ACL, 2011).

J. Lehmann, B. Goncalves, J. J. Ramasco, and C. Cattuto, in *Proc. WWW* (2012).

L. Weng, A. Flammini, A. Vespignani, and F. Menczer, Scientific Reports **2** (2012).

C. Bauckhage, K. Kersting, and F. Hadiji, in *Proc. ICWSM* (2013).

K. Radinsky, K. Svore, S. Dumais, J. Teevan, A. Bocharov, and E. Horvitz, in *Proc. WWW* (2012).

R. Kobayashi and R. Lambiotte, in *Proc. ICWSM* (2016).

Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (2015b), pp. 1513–1522.

Y. Ogata and K. Katsura, Annals of the Institute of Statistical Mathematics **40** (1988).

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, ACM Transactions on Mathematical Software **23** (1997).

S.-E. Benkabou, K. Benabdeslem, and C. Bruno, in *Proc. ICML Workshop on Anomaly Detection* (2016).

E. T. Jaynes, Physical Review **106** (1957).

Y.-L. K. Samo and S. Roberts, in *Proc. ICML* (2015b).

H. Akaike, in *Selected Papers of Hirotugu Akaike*, edited by E. Parzen, K. Tanabe, and G. Kitagawa (Springer, 1998), pp. 199–213.

M. Müller, in *Information Retrieval for Music and Motion* (Springer, 2007), pp. 69–84.

A.-L. Barabasi, Nature **435** (2005).

R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. Amaral, PNAS **105** (2008).

J. Yang, L. Adamic, and M. Ackerman, in *Proc. ICWSM* (AAAI, 2008).

L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman, in *Proceedings of the 17th international conference on World Wide Web* (ACM, 2008), pp. 665–674.

B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, in *Proceedings of the 21st international conference companion on World Wide Web* (ACM, 2012), pp. 775–782.

K. K. Nam, M. S. Ackerman, and L. A. Adamic, in *Proceedings of the SIGCHI conference on human factors in computing systems* (ACM, 2009), pp. 779–788.

G. Begelman, P. Keller, F. Smadja, et al., in *Proc. WWW'06 Workshop on Collaborative Web Tagging* (2006).

D. Koller and M. Sahami, Tech. Rep., Stanford InfoLab (1997).

P. Schmitz, in *Proc. WWW'06 Workshop on Collaborative Web Tagging* (2006).

G. Tibély, P. Pollner, T. Vicsek, and G. Palla, PloS ONE **8** (2013).

H. Halpin, V. Robu, and H. Shepherd, in *Proc. WWW* (ACM, 2007).

S. A. Golder and B. A. Huberman, J. of Information Science **32** (2006).

D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina, in *Proc. WSDM* (ACM, 2009).

D. Blei, T. Griffiths, and M. Jordan, J. of the ACM **57** (2010).

V. Bhat, A. Gokhale, R. Jadhav, J. Pudipeddi, and L. Akoglu, in *Proc. ASONAM* (2014).

K. Heller and Z. Ghahramani, in *Proc. ICML* (2005).

M. E. Newman, SIAM review **45**, 167 (2003).

R. G. Palmer, D. L. Stein, E. Abrahams, and P. W. Anderson, Physical Review Letters **53**, 958 (1984).

C. P. Bachas and B. A. Huberman, Physical review letters **57**, 1965 (1986).

M. Aoki, Journal of economic dynamics and control **18**, 865 (1994).

D. Sornette, *Critical phenomena in natural sciences: chaos, fractals, selforganization and disorder: concepts and tools* (Springer Science & Business Media, 2006).

D. Sornette and A. Johansen, Physica A: Statistical Mechanics and its Applications **261**, 581 (1998).

E. Alvarez-Lacalle, B. Dorow, J.-P. Eckmann, and E. Moses, Proceedings of the National Academy of Sciences **103**, 7956 (2006).

P. Érdi, T. Grőbler, and P. Marton, in *Nature, cognition and system II* (Springer, 1992), pp. 193–203.

T. L. Saaty, European journal of operational research **48**, 9 (1990).

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, in *Advances in Neural Information Processing Systems* (2018), pp. 9525–9536.

K. Simonyan, A. Vedaldi, and A. Zisserman, arXiv preprint arXiv:1312.6034 (2013a).

D. Erhan, Y. Bengio, A. Courville, and P. Vincent, University of Montreal **1341**, 1 (2009).

A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, arXiv preprint arXiv:1605.01713 (2016).

M. Sundararajan, A. Taly, and Q. Yan, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. org, 2017), pp. 3319–3328.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, arXiv preprint arXiv:1412.6806 (2014a).

M. D. Zeiler and R. Fergus, in *European conference on computer vision* (Springer, 2014), pp. 818–833.

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, arXiv preprint arXiv:1611.07450 (2016a).

A. Shrikumar, P. Greenside, and A. Kundaje, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. org, 2017), pp. 3145–3153.

L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, arXiv preprint arXiv:1702.04595 (2017).

C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud (2018).

M. T. Ribeiro, S. Singh, and C. Guestrin, in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (ACM, 2016), pp. 1135–1144.

G. Arvanitidis, L. K. Hansen, and S. Hauberg, arXiv preprint arXiv:1710.11379 (2017).

D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, arXiv preprint arXiv:1807.07543 (2018).

I. J. Goodfellow, J. Shlens, and C. Szegedy, arXiv preprint arXiv:1412.6572 (2014b).

F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O'Brien, S. Schieber, J. Waldo, D. Weinberger, and A. Wood, arXiv preprint arXiv:1711.01134 (2017).

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, in *International Conference on Learning Representations* (2018), URL https://openreview.net/forum?id=HkL7n1-0b.

O. Bousquet, S. Gelly, I. Tolstikhin, C. J. Simon-Gabriel, and B. Schölkopf, Tech. Rep. (2017).

L. D. Landau and E. M. Lifshitz, *Course of theoretical physics* (Elsevier, 2013).

S. N. Majumdar, in *The Legacy Of Albert Einstein: A Collection of Essays in Celebration of the Year of Physics* (World Scientific, 2007), pp. 93–129.

R. Feynman and A. H. Q. Mechanics, Quantum Mechanics (Welland (1965).

M. P. do Carmo, *Differential geometry of curves and surfaces* (Prentice Hall, 1976), ISBN 978-0-13-212589-5.

K. Simonyan, A. Vedaldi, and A. Zisserman, arXiv preprint arXiv:1312.6034 (2013b).

D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, arXiv preprint arXiv:1706.03825 (2017).

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, arXiv preprint arXiv:1412.6806 (2014b).

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, arXiv preprint arXiv:1611.07450 (2016b).

Y. LeCun, *The mnist database of handwritten digits. nec research institute* (1998).

Z. Liu, P. Luo, X. Wang, and X. Tang, in *Proceedings of the IEEE international conference on computer vision* (2015), pp. 3730–3738.

K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, in *2009 IEEE conference on computer vision and pattern recognition* (Ieee, 2009), pp. 248–255.

C. Bär and F. Pfäffle, Preprints des Instituts für Mathematik der Universität Potsdam 1 (2012).

M. Taylor, Applied Mathematical Sciences 116, Springer-Verlag New York (2011).

S. Ioffe and C. Szegedy, arXiv preprint arXiv:1502.03167 (2015).

J. D. C. Little, Operations Research **9**, 383 (1961).

N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, in *Proceedings of the 36th International Conference on Machine Learning*, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, Long Beach, California, USA, 2019), vol. 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310, URL http://proceedings.mlr.press/v97/rahaman19a.html.

Publications

César Ojeda, Ramsés J. Sánchez, Kostadin Cvejosky, Jannis Schücker, Christian Bauckhage and Bogdan Georgiev. Auto Encoding Explanatory Examples with Stochastic Paths. (Accepted in) *International Conference on Pattern Recognition*, ICPR, 2020.

César Ojeda, Ramsés J. Sánchez, Kostadin Cvejosky, Jannis Schücker, Christian Bauckhage and Bogdan Georgiev. Switching Dynamical Systems with Deep Neural Networks. (Accepted in) *International Conference on Pattern Recognition*, ICPR, 2020

César Ojeda, Kostadin Cvejosky, Ramsés J. Sánchez, Jannis Schücker, Bogdan Georgiev, Christian Bauckhage. Recurrent Adversarial Service Times. CoRR abs/1906.09808 (2019).

César Ojeda, Kostadin Cvejoski, Rafet Sifa, and Christian Bauckhage. Patterns and Outliers in Temporal Point Processes. *In Proceedings of Advances in Intelligent Systems and Computing*, IntelliSys, 2019.

César Ojeda, Kostadin Cvejoski, Rafet Sifa and Christian Bauckhage. Inverse Dynamical Inheritance in Stack Exchange Taxonomies, *The 11th International AAAI Conference on Web and Social Media*, AAAI, 2017.