Carnegie Mellon University
Dietrich College of Humanities and Social Sciences
Dissertation

Submitted in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy

Title: Networks, Point Processes, and Networks of Point Processes

Presented by: Neil A. Spencer

Accepted by: Department of Statistics and Machine Learning Department

Readers:

Professor Robert Kass, Advisor

Professor Cosma Rohilla Shalizi, Advisor

Professor Brian Junker

Professor Jared S. Murray

Approved by the Committee on Graduate Degrees:

Richard Scheines, Dean Date

Carnegie Mellon University

Networks, Point Processes, and Networks of Point Processes

A Dissertation Submitted to the Graduate School
in Partial Fulfillment of the Requirements for the degree

Doctor of Philosophy

in

Statistics and Machine Learning

by

Neil A. Spencer

Department of Statistics and Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Carnegie Mellon University

August 2020

To my family

Acknowledgements

This dissertation would not have been possible without the support of many friends, family, and colleagues.

First of all, I would like to thank the members of my dissertation committee: Rob Kass, Cosma Shalizi, Brian Junker, and Jared Murray. My PhD followed a bit of a non-standard route in that I worked closely with each of you on four different projects to form my dissertation. While this set-up was challenging for me to balance at times, it was worth it for the chance to receive mentorship and guidance from each of you. Your unique styles complemented each other well—it was a pleasure to get to work and learn so much from all of you. The fact that we could make it work is a testament to your patience and kindness as advisors and colleagues. Thank you so much. Along the same vein, I also owe a great deal of thanks to Jay Kadane for his support and guidance during our work together along with the Allegheny County Office of the Medical Examiner. I learned a lot about collaboration through our work.

I am thankful for the support and encouragement from the members of three groups I participated on campus: Networkshop, Neurostats, and CSAFE. It was a pleasure to get such insightful, critical, and honest feedback on a regular basis, as well as get to hear about all of your interesting work. It helped shape my growth as a researcher and presenter, as well as steer me toward promising research directions. Most importantly, it was also a lot of fun!

Thank you to the fellow members of my cohort: Alden, Daren, Ilmun, Jaehyeok, Kayla, Octavio, Xiao Hui, YJ, and Zongge. Your support, camaraderie, and shared meals over the past five years helped make the process of working on a PhD to be manageable and enjoyable experience. Moreover, I would also like to thank all of my fellow PhD students, as well as the faculty and staff in the statistics department and machine learning department, for helping to foster a fun and comfortable environment in which to work and learn. Many of the IM sports games, parties, and game nights were highlights of my past five years. Special shoutout to Brendan and Xiao Hui for being such great conference travel buddies multiple times.

During this global pandemic, I feel like I can't write an acknowledgment section without expressing extreme gratitude to my lockdown bubble of Ben, Ciaran, and Kayla. You all—along with spike ball and Twisters—helped keep me sane over the last several months. I'll miss it!

It is also important that I acknowledge my undergraduate advisors—Franklin and Pritam—for your continued support you've provided over the years. I am eternally grateful for your introducing me to research and your ongoing encouragement. Also, to Sean Jewell and Creagh Briercliffe, it has been a pleasure to continue to keep in touch after UBC. Continuing to share advice about our various PhD experiences helped me to navigate both my PhD and what to do afterwards.

Finally, special thanks to both my family and Emily. Much of what I have accomplished I owe to your unwavering love, support, and encouragement throughout this entire process. I love you all.

The work in this dissertation was completed with financial support from a doctoral fellowship from the Natural Science and Engineering Research Council of Canada, as well as grant support from IES (US Dept of ED) Award 305D150045, the National Institute of Mental Health grant R01MH064537, and the Center for Statistics and Applications in Forensic Evidence (CSAFE) through Cooperative Agreement 70NANB15H176 between NIST and Iowa State University, which includes activities carried out at Carnegie Mellon University, University of California Irvine, and University of Virginia.

Abstract

I consider two classes of statistical models: networks and point processes. These random structures are often used in situations where traditional statistical assumptions do not hold (e.g. the data are not independent or identically distributed), meaning extra care must be taken to develop and extend statistical theory and tools to these settings. This dissertation consists of four separate projects (Chapters 2–5) each tackling a different problem pertaining to the point processes, network models, or their interface.

Chapter 2 develops a new hierarchical Bayesian point process model for an application in forensic science. Chapter 3 develops a new point process-based framework for latent position network models—as well as supporting theory—to fill an important gap in the existing sparse network literature. Chapter 4 develops an efficient Monte Carlo algorithm for Bayesian inference of large latent position network models, and Chapter 5 develops an efficient Monte Carlo algorithm for Bayesian inference of high-dimensional point process models that are useful for analyzing networks of neural spike trains.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

For the past five years, I have been working on problems related to two classes of statistical models: networks and point processes. These random structures are often used in situations where traditional statistical assumptions do not hold (e.g. the data are not independent or identically distributed), meaning extra care must be taken to develop and extend statistical theory and tools to these settings. This dissertation consists of four separate projects (Chapters 2–5) each tackling a different problem pertaining to the point processes, network models, or their interface.

Chapter 2 develops a new hierarchical Bayesian point process model for an application in forensic science. Chapter 3 develops a new point process-based framework for latent position network models—as well as supporting theory—to fill an important gap in the existing sparse network literature. Chapter 4 develops an efficient Monte Carlo algorithm for Bayesian inference of large latent position network models, and Chapter 5 develops an efficient Monte Carlo algorithm for Bayesian inference of high-dimensional point process models that are useful for analyzing networks of neural spike trains.

Each chapter in this dissertation represents a self-contained project that has been submitted for publication or is in preparation for submission. As such, I have made an effort to include all relevant notation, terminology, and background information within-chapter for the benefit of the reader—each chapter can be read independently without relying on other chapters. Unfortunately, this independence comes at the cost of a few minor notation clashes across chapters. However, confusion can be minimized by focusing on each chapter individually.

Although I intend for the chapters are intended to be standalone, the following short background section may be useful to a reader looking to orient themselves on some basics of network models and point processes prior to tackling a specific chapter. Moreover, Section 1.1.2 contains some additional background regarding marked temporal Hawkes process—the model underlying the spike train GLM—that is not included in Chapter 5.

1.1 Background

1.1.1 Latent Variable Network Models and Graphons

Networks are a tool for modeling relational information between entities, such as ties between friends. Usually these take the form of binary matrix A with entry A_{ij} indicating the presence of a link between entities i and j ($1 \leq i \leq j \leq n$). It can be useful to mathematically formalize networks as stochastic graphs, treating the ties A_{ij} as random edges between n nodes.

Chapters 3 and 4 of this dissertation (and to a lesser degree, Chapter 5) focus on a specific family of stochastic graph models called latent variable network models. These models explain heterogeneities in connection patterns between nodes using node-level latent variables. I primarily focus on a sub-class of this family called the latent position model, with the stochastic block model earning the occasional mention as well.

Latent position models (LPM)(Hoff et al., 2002) are characterized by each node of the network possessing a latent position Z_i in a metric space (S, ρ). The edges are modeled as drawn independently according to

$$\mathbb{P}(A_{ij} = 1 \mid Z_i, Z_j) = K(\rho(Z_i, Z_j)), \tag{1.1}$$

with $K : \mathbb{R}_+ \to [0, 1]$ known as the link probability function. Typically, K assumed to be decreasing, S is assumed to be a low-dimensional Euclidean space, and the Z_i are treated as random effects drawn independently from a multivariate Gaussian on S. The Euclidean space provides an intuitive way to visualize the network, and the decreasing K ensures connection transitivity–friends of friends are more likely to be friends because nodes due to the triangle inequality.

Similar to LPMs, stochastic block models (SBM) (Holland et al., 1983) are characterized by each node having a latent block membership $Z_i \in \{1, \ldots, M\}$, with edges drawn independently according to

$$\mathbb{P}(A_{ij} = 1 \mid Z_i, Z_j) = P_{Z_i, Z_j}, \tag{1.2}$$

with $P \in [0, 1]^{M \times M}$. The matrix P is usually assumed to be larger along the diagonal, leading to edges being more likely within communities than between between. Typically, block memberships are treated as random effects, drawn from a categorical distribution over $\{1, \ldots, M\}$. Performing inference on the block memberships is essentially model-based clustering on the graph nodes.

The similarity in how LPMs and SBMs are formulated is not purely coincidental. They both fall within a more general class of latent variable network models called graphon[*] models Diaconis and Janson (2008). Graphons characterize jointly exchangeable random graphs (graph distributions invariant to vertex index

[*]Graphon is a portmanteau of graph and function

permutations (Orbanz and Roy, 2015)), as shown by the Aldous-Hoover theorem (Aldous, 1981). The graphon set-up involves each node possessing a latent variable Z_i in S drawn iidly from a distribution f. Conditionally on Z_i, Z_j and a function $w : S^2 \to [0, 1]$ (called the graphon), each edge is drawn independently according to

$$\mathbb{P}(A_{ij} = 1 \mid Z_i, Z_j) = w(Z_i, Z_j). \tag{1.3}$$

It is clear that both the SBM and LPM correspond to certain classes of w. Many other popular latent variable network models in the literature also fall within the graphon framework, such as the mixed membership stochastic block model (Airoldi et al., 2008), the random dot product model (Athreya et al., 2017), and hierarchical random graphs (Clauset et al., 2008). Graphons provide a common lens for considering all of these models, as well as a natural nonparametric framework for assigning priors on jointly exchangeable graphs (Lloyd et al., 2012).

Some of the future work described in Chapter 5 relies on a more general statement Aldous-Hoover for exchangeable real-valued arrays W. These arrays can be viewed as weighted graphs with the $W_{ij} \in \mathbb{R}$ corresponding to edge weights. They are characterized as follows.

As before, each node possesses latent variable $Z_i \in S$ distributed according to f. Furthermore, for each pair of indices (i, j), generate a random variable $U_{ij} \in \mathbb{R}$ according to a distribution q. Now, the weighted edges W_{ij} are given by

$$W_{ij} = w(Z_i, Z_j, U_{ij}), \tag{1.4}$$

where $w : S^2 \times \mathbb{R} \to [0, 1]$. The result in (1.3) can be recovered from this more general result by choosing w and q to define the appropriate Bernoulli distribution. The SBM and LPM can be extended to this setting (Aicher et al., 2014; Sewell and Chen, 2016).

Though the exchangeability assumption and the resultant graphon framework are intuitively appealing, the framework has its drawbacks. Exchangeability has the unfortunate side effect of forcing the resultant network models to be dense— the expected number of edges in the model must grows quadratically with the number of nodes (Orbanz and Roy, 2015). Many real-world networks exhibit sparse edge scaling behaviour (the expected number of edges grows sub-quadratically) (Newman, 2010). Consequently, exchangeable models are mis-specified in these contexts, necessitating alternatives to the exchangeability. Developing and analyzing network models which are able to accommodate different levels of sparsity is an active research area (e.g. Caron and Fox (2014); Borgs et al. (2014)). I contribute to this literature in Chapter 3 by developing and analyzing a new framework which generalizes the latent position network model to accommodate for sparsity. The approach relies heavily on point processes, in particular the Poisson process.

1.1.2 Point Processes and Marked Point Processes

A point process (Daley and Vere-Jones, 2007) is characterized by a random counting measure X on a well-behaved[†] space S equipped with σ-field $\mathcal{S}$. However, it is usually sufficient to think of X as the random countable collection of points induced by the atoms in this counting measure. I usually take this view, treating a point process X as a random collection of points on S, with both the number of points and their locations randomly distributed. Occasionally, I do treat X as a random measure, using $X(A)$ to denote the value of the random measure X on $A \in \mathcal{S}$.

Certain popular choices for the space S have received special attention in the literature. When S is the positive real line $\mathbb{R}_+$, X is commonly referred to as a temporal point process (Brillinger et al., 2002), with the points in the process forming a well-ordered collection of arrival times (e.g. the points could represent times at which a neuron in the brain fires, or random points at which the parameters of an algorithm are updated). When $S \subseteq \mathbb{R}^d$, X is commonly referred to as a spatial point process (Baddeley et al., 2007); the points can be thought of as d-dimensional locations in space (e.g. the locations of scrapes on a shoe sole). Note that the space S need not correspond to physical time or physical space. The latent space for a LPM in Chapters 3 and 4 is an example of an abstract space for a spatial Poisson process. The zigzag process in Chapter 5 is an example of temporal Poisson process with an abstract time dimension.

In some instances, each point $x_i \in X$ is also associated with a mark m_i. Often, these marks are discrete $m \in \{1, \ldots, M\}$ indicating the membership of the point to one of M distinct groups. Data of this form are common when working with groups neural spike trains: continuous time series recording firing times of M distinct neurons in the brain. Instead of treating the group of points as M distinct point processes, all points can jointly modeled as a single process known as a marked point process (Daley and Vere-Jones, 2007, Chapter 6). Though marked point processes are seemingly distinct, they can set within the traditional point process framework by augmenting the original space S with the mark space $\{1, \ldots, M\}$ as $S^* = S \times \{1, \ldots, M\}$ with $\mathcal{S}^*$ induced through the product σ-field. For instance, a marked temporal point process is simply a point process on $S = \mathbb{R}_+ \times \{1, \ldots, M\}$.

For my dissertation, I intend to mainly focus on two families of S: spatial point processes for Chapters 2 and 3, and temporal point processes (both marked and unmarked) for Chapter 5. For spatial point processes and unmarked temporal point processes, I focus on a special class of point process called the Poisson process (Kingman, 1993). For the marked temporal point processes, I focus on mutually-exciting Hawkes processes (Hawkes, 1971; Laub et al., 2015). I describe them both here, starting with the Poisson process.

A Poisson process (PP) on $(S, \mathcal{S})$ is one of the simplest point processes on S, assigning a Poisson distributed number of points to each of the subsets of S in $\mathcal{S}$, resulting in a random configuration of points over S. A PP is parameterized by a measure Λ on S called the rate measure. When Λ can be normalized to

[†]Specifically, a locally compact second countable Hausdorff space

define a density λ over S, a realization X of $\mathrm{PP}(\Lambda)$ can be generated according to the following procedure:

$$N \sim \text{Poisson}(\Lambda(S)), \tag{1.5}$$

$$X = (x_1, \ldots, x_N) \text{ such that} \tag{1.6}$$

$$x_i \sim^{\text{iid}} \lambda. \tag{1.7}$$

There are a few properties of the PP which are especially important to note. Conditional on N, the points $(x_1, \ldots, x_N)$ are independently and identically distributed. This separates the inference of Λ into two separate components: inferring $\Lambda(S)$ from the number of points, and inferring λ from their locations. A second property is that, for $A \cap B = \emptyset$, $X(A)$ and $X(B)$ are each independent and Poisson distributed. Furthermore, the restrictions of X to A and B are each independent point processes. This property allows the above generative approach to be extended to cases in which Λ is only σ-finite by partitioning S into $S_1, S_2, \ldots$ such that each $\Lambda(S_i) < \infty$, then combining independent Poisson processes $\mathrm{PP}(\Lambda I_{S_i})$. Here, the $\mathrm{PP}(\Lambda I_{S_i})$ are generated by restricting Λ to each S_i. This σ-finite case is important, as it is required to define infinite graphs for Chapter 3.

Chapter 5 makes use of two additional convenient properties—superposition and thinning—for generating PPs. The superposition principle indicates that the sum of two independent Poisson processes over S is itself a Poisson process over S. Specifically, if $X_1 \sim \mathrm{PP}(\Lambda_1)$ and $X_2 \sim \mathrm{PP}(\Lambda_2)$ are independent over S, then $X_1 + X_2 \sim \mathrm{PP}(\Lambda_1 + \Lambda_2)$. By induction, this results extends immediately to the sum of multiple PPs.

The Poisson thinning principle states that if each point in a PP is independently subsampled, the result is still a PP with a rate that depends on the specific subsampling probabilities. Specifically, if the points $x_i \in X$ in a Poisson process $X \sim \mathrm{PP}(\Lambda_1)$ are randomly discarded such that a point $x_i \in S$ is kept with probability $\pi(x_i)$ $(\pi : S \to [0, 1])$ then the resultant thinned process X' is also a Poisson process—$X' \sim PP(\pi \times \Lambda_1)$.

The independence properties of the PP make it one of simplest and most tractable families of point processes. However, it sometimes beneficial to relax these independence assumptions. In particular, treating Λ as also being random measure allows one to model over-dispersion in the number of points observed, while also introducing global dependence between the locations of all points. The resultant marginal distribution over X is referred to as a Cox process, or a doubly-stochastic Poisson process (Cox and Isham, 1980). Cox processes are particularly useful for spatial modeling when there are replicated observations in the data, or for hierarchically modeling several dependent spatial point processes as I do in Chapter 2.

In other cases, the global dependence induced by the Cox processes is not enough to capture the dependencies in the data. In temporal modeling, one often wants the dependence to be local rather than global; the dependence between two times periods should depend on the time elapsed between them. For instance, it is known that the firing of a neuron in the brain can trigger (or inhibit) its firing again within a nearby time interval, causing strong dependence between a neuron's activity at nearby time points. A neuron may trigger or inhibition other neurons as well, enforcing local dependence between points with different marks. I consider such a problem in Chapter 5.

This local-in-time dependence between points can be directly modeled by including it in the generative process– this is the case in Hawkes process (HP) (Laub et al., 2015), in which a point at one time point can inhibit or trigger points in the following time period. In Chapter 5, I consider a special class of marked Hawkes processes referred to as mutually exciting Hawkes processes (MEHP), which capture dependencies between marks in time.

The following is the set-up of a MEHP. It is a marked temporal point process on $S = [0, T] \times \{1, \ldots, M\}$ for which $[0, T]$ is the time space and $1, \ldots, m$ are the possible values of marks. The MEHP is parametrized by two functions: the *background rate* $\lambda : S \times \{1, \ldots, M\} \to \mathbb{R}_+$ which controls the baseline rate at which each mark occurs, and the *triggering rate* $\eta : [0, T] \times \{1, \ldots, M\}^2$, which controls how the probability of a point and its mark at a current time points is influenced by the history of the process (the points $(m_i, t_i) \in X$ such that $t_i < t$). The following expression defines the rate of points with mark m at time $t \in [0, T]$:

$$f_m(t) = \phi(\lambda(t, m) + \sum_{i:t_i<t} \eta(t - t_i, m_i, m)). \tag{1.8}$$

Here, ϕ is a link function. In a standard (linear) MEHP, ϕ is defined as the rectifier $\phi(x) = \min(0, x)$ to prevent the rate from falling below zero in the presence of inhibition ($\eta < 0$). In Chapter 5, I consider models for which ϕ is defined as a non-negative nonlinear function, such as the exponential or sigmoid function (we rely on the sigmoid function in Chapter 5). This approach results in a nonlinear Hawkes process (Brémaud and Massoulié, 1996).

It is common in the literature to model the triggering function η as following a parametric form, such as $\eta(t, m_i, m_j) = \beta_{ij} g(t; \theta)$ where $\beta \in \mathbb{R}^{m \times m}$ and g is a class of smooth functions parametrized by $\theta \in \mathbb{R}^d$ (e.g. $g(t; \theta) = \exp(-\theta t)$). I make such an assumption in Chapter 5, modeling each η as a linear combination of known basis functions, therefore making estimation of η equivalent to that of estimating β.

When modeling non-linear Hawkes processes with exponential or sigmoid link functions, it is common practice to bin the observations into counts over small time increments. Inferring the parameters of the baseline $\lambda(t, m)$ and triggering β in this setting is equivalent to learning a log-linear model (Kass et al., 2011; Truccolo et al., 2005). This approach, sometimes referred to as a log-linear MEHP, is more computationally tractable than the non-discrete case; it permits out-of-the-box software implementations of generalized linear models to be used to the fit the models.

However, care must be taken when inferring β to ensure that the resultant model is stable — the excitation level should not routinely result in the number of points exploding (Chen et al., 2018). Two ways to greatly reduce the risk of estimating an unstable model is to assume sparsity in the parameters of η (requiring that most marks do not influence each other), and placing Bayesian priors on the parameter values to place little prior probability on models which are unstable. Such additional structure can complicate inference. We address this problem in Chapter 5.

Chapter 2

A Bayesian Hierarchical Model for Evaluating Forensic Footwear Evidence

2.1 Introduction

Forensic footwear analysis encompasses a suite of techniques used to analyze latent shoeprints as part of forensic investigations. A principal goal of these investigations is to link a suspect's shoe to a crime scene print, providing evidence to place the suspect at the scene of the crime. Figure 2.1a provides an example of a latent crime scene shoeprint.

As described by Bodziak (2017), the procedure for determining the source of a latent print typically consists of two stages. First, the examiner inspects the tread of the latent print to identify class characteristics (brand, model, and size) of the source shoe. This identification can be carried out manually, or automated using tread matching algorithms (e.g. Srihari and Tang (2014); Richetelli et al. (2017a); Kong et al. (2017)).

Manufacturers routinely produce thousands of shoes of the same make and model, meaning that class characteristics alone are often insufficient for determining a print's source. For this reason examiners regularly turn to a second stage of analysis: the inspection of *accidentals*. Accidentals, also known as randomly acquired characteristics, are the post-manufacturing cuts, scrapes, holes, and debris that accumulate on a shoe sole. Examiners are trained to identify accidentals on a shoe by inspecting both the shoe's sole and test impressions— high quality prints created using the shoe in a controlled laboratory setting. Figure 2.1b, Figure 2.1c, and Figure 2.1d depict a shoe sole, test impression, and accidentals locations, respectively.

These images all correspond to the same shoe obtained from the JESA database (Yekutieli et al., 2012) (we describe the JESA database in §2.2.2).

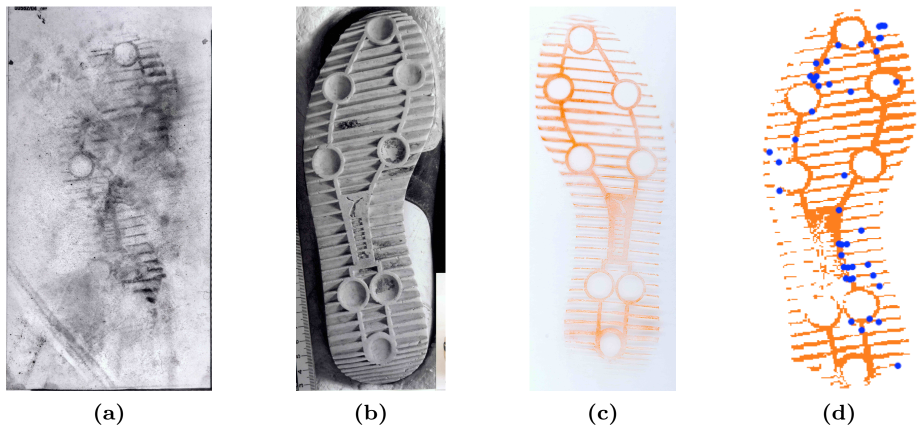


Figure 2.1: (a)-(d) represent objects pertaining to the same shoe from the JESA database. (a) is a photograph of a latent crime scene print, (b) a photo of the shoe's sole, (c) is a raw image of a test impression, and (d) is the contact surface obtained from standardizing the test impression. The superimposed blue points in (d) correspond to accidental locations.

In theory, if both the class characteristics and the accidentals of a suspect's shoe coincide exactly with those detected from the crime scene print, then the suspect's shoe is almost certainly the source of the print. In practice, the comparison is less clear-cut. Latent crime scene prints are typically of low quality, making it difficult to pick out all of the individual accidentals. Furthermore, accidental locations are known to vary slightly from test print to test print due to variability in the impression-taking process (Shor et al., 2017), so there is some uncertainty on their exact locations on the source shoe. As a result, accidental comparisons typically involve comparing a subset of approximate accidental locations on the test impression to those detected on the crime scene print. This uncertainty leaves the possibility of a false positive due to chance, especially for partial prints and tread patterns on which accidentals are very likely to occur in certain regions.

To account for the possibility of a false positive, shoeprint analysts are encouraged to provide a measure of the uncertainty of the match when testifying in court (Edwards and Gotsonis, 2009). One popular summary for communicating this uncertainty is the random match probability (RMP) (Thompson and Newman, 2015). The RMP is the probability that a randomly sampled shoe would produce a print matching the observed features at the crime scene. For instance, if 15 out of 10000 relevant shoes were consistent with the crime scene print, the RMP would be 0.0015.

The standard approach for evaluating RMPs decomposes into three terms: the evidence given by the class characteristics, the evidence based on general wear, and accidental-based evidence (Evett et al., 1998; Skerrett et al., 2011). In this work, we focus on accidental-based evidence, inspired by the recent report on forensic science by The President's Council for Advisors on Science (PCAST, 2016) that criticized existing work in the area.

We address the concerns of PCAST (2016) by developing and estimating the parameters of a model for the distribution of accidental configurations on a shoe. Specifically, we model the spatial distribution of accidentals on a shoe sole as a point process, treating the sole's tread pattern as a covariate. We fit and evaluate our model using the *JESA database* (Yekutieli et al., 2012), a ground truth dataset of 386 accidental-annotated shoeprints compiled by the Israeli Police Department's Division of Forensic Science. The JESA database is one of the largest existing databases of its kind (Speir et al., 2016), consisting of shoes with a variety of tread patterns.

We define our model within a hierarchical Bayesian framework, pooling information across JESA to infer general trends spanning a broad variety of shoes. Our model is a finite resolution version of the normalized compound random measure framework of Griffin and Leisen (2017), modified to incorporate spatial covariates and dependency of the intensity across space. We develop the computational tools to fit our model, evaluate it, and demonstrate that it outperforms existing approaches by a wide margin.

The remainder of this chapter is organized as follows. In Section 2.2, we review the literature related to random match probabilities, formalize the link between evaluating random match probabilities and modeling spatial distributions of accidentals, describe the JESA database of annotated shoeprints collected by Yekutieli et al. (2012), and review the relevant literature pertaining to vectors of dependent probability measures. In Section 2.3, we provide the details of our hierarchical Bayesian model for spatial configurations of accidentals. In Section 2.4, we propose a Markov chain Monte Carlo algorithm for inferring the parameters of the model, and an importance sampling algorithm for evaluating marginal likelihoods. In Section 2.5, we showcase the results of fitting our model to the JESA dataset and compare its performance to other candidate models. Section 2.6 contains some concluding remarks.

2.2 Preliminaries

2.2.1 Random Match Probabilities

A theory to evaluate RMPs for footwear evidence was laid out in Evett et al. (1998) in the context of evaluating likelihood ratios. The framework is equally applicable to evaluating raw RMPs. Let y denote a crime scene print and $\mathbb{A}$ denote the relevant population of plausible sources of the crime scene print. For instance, $\mathbb{A}$ could be all shoes belonging to residents of a particular city or town. As per Evett et al. (1998), the random match probability for footwear evidence is given by

$$\text{RMP} = p(y \equiv s \mid s \sim \mathbb{A}) \tag{2.1}$$

where $y \equiv s$ indicates that shoe s exhibits features consistent with those of the print y, and $s \sim \mathbb{A}$ is shorthand for s being chosen uniformly at random from all shoes in the set $\mathbb{A}$. Further discussion of random match probabilities with examples from forensic science is available in Srihari and Su (2011).

Following the classical two step process of forensic footwear analysis, Evett et al. (1998) suggested that the RMP be calculated using the factorization RMP = $\text{rmp}_M \text{rmp}_U$. Here, rmp_M denotes the probability that a randomly chosen shoe in $\mathbb{A}$ has class characteristics matching the latent crime scene print, and rmp_U denotes the probability that the shoe is also consistent with the wear patterns and accidentals, given that it matches on the class characteristics. Skerrett et al. (2011) refined this representation by further decomposing rmp_U into rmp_W and rmp_V, corresponding to separate conditional probabilities of matching on general wear and accidentals, respectively.

Let y_M, y_W, and y_V denote the class characteristics, general wear, and accidentals observed on the latent print y with s_M, s_W, s_V denoting the same features as observed on a shoe $s \in \mathbb{A}$. The factorization proposed by Skerrett et al. (2011) can be formally expressed as

$$\text{RMP} = \text{rmp}_M \cdot \text{rmp}_W \cdot \text{rmp}_V, \tag{2.2}$$

$$\text{rmp}_M = p(y_M \equiv s_M \mid s \sim \mathbb{A}), \tag{2.3}$$

$$\text{rmp}_W = p(y_W \equiv s_W \mid s \sim \{s' \in \mathbb{A} : y_M \equiv s'_M\}), \tag{2.4}$$

$$\text{rmp}_V = p(y_V \equiv s_V \mid s \sim \{s' \in \mathbb{A} : y_M \equiv s'_M, y_W \equiv s'_W\}), \tag{2.5}$$

where $y_M \equiv s_M$ denotes the class characteristics of s being consistent with those of y, and $y_W \equiv s_W$ and $y_V \equiv s_V$ defined similarly. Implicit in this decomposition is the assumption that $y \equiv s$ is characterized by $y_M \equiv s_M$, $y_W \equiv s_W$, and $y_V \equiv s_V$, a reasonable choice given that these features form the basis of forensic footwear analysis (Bodziak, 2017). Strategies for evaluating rmp_M and rmp_W based on relevant databases (e.g. Evett et al. (1998); Champod et al. (2004) for rmp_M and Fruchtenicht et al. (2002); Facey et al. (1992); Bodziak et al. (2012) for rmp_W) were discussed in Skerrett et al. (2011). However, evaluating the accidental-based component rmp_V was left as a subject for future work. In this work, we focus on the remaining accidental-based component. We begin by making two simplifying assumptions.

First, we follow Petraco et al. (2010) in assuming that the evidence present in a configuration of accidentals on a crime scene print y_V is characterized by the locations (e.g. the blue points shown in Figure 2.1d). We omit secondary characteristics such as shape or size of the accidental as they are difficult to reliably glean from latent prints. We use x^s to denote the accidental locations on shoe s and x^y to denote the locations detected on print y. Employing a standardized coordinate system (details provided in §2.2.2), we have $x^s \in ([0, 100] \times [0, 200])^{N_s}$, $x^y \in ([0, 100] \times [0, 200])^{N_y}$ where N_s denotes the number of accidentals on shoe s and N_y denotes the number detectable on print y. We use $x^s_n = (x^s_{n,1}, x^s_{n,2})$ to denote the nth row of x^s.

Because examiners are adept at recovering y_M and y_W from a shoeprint y, our second assumption is that a shoe's class characteristics and wear are characterized by its *contact surface*. A shoe's contact surface refers to the portion of its sole that typically touches the ground when worn — the part responsible for leaving prints. An example contact surface is provided in Figure 2.1d. We provide a more detailed definition of contact surface in §2.2.2. Letting $\mathcal{C}^s$ denote the contact surface of shoe s, this assumption can be formalized as $s, s' \in \mathbb{A}$, $\mathcal{C}^s = \mathcal{C}^{s'}$ if and only if $s_M \equiv s'_M$ and $s_W \equiv s'_W$.

After characterizing y_V using accidental locations and y_W, w_M using the contact surface, we can now re-express the accidental-based random match probability in (2.5) in a form that is more tractable for statistical inference. The relation $y_V \equiv s_V$ reduces to a comparison of the point clouds x^y and x^s (denoted $x^y \equiv x^s$). The set $\{s' \in \mathbb{A} : y_M \equiv s'_M, y_W \equiv s'_W\}$ reduces to the set of relevant shoes with the given contact surface (i.e. $\mathbb{A}_{\mathcal{C}^y} = \{s' \in \mathbb{A} : \mathcal{C}^{s'} = \mathcal{C}^y\}$, where $\mathcal{C}^y$ denotes the contact surface as determined from y). Thus, the accidental-based random match probability given in (2.5) reduces to

$$\text{rmp}_V = p(x^y \equiv x^s \mid s \sim \mathbb{A}_{\mathcal{C}^y}). \tag{2.6}$$

In theory, computing rmp_V using (2.6) is straightforward. One would simply inspect all shoes in $\mathbb{A}$ with contact surface $\mathcal{C}^y$ to determine the ratio that also have accidentals consistent with x^y. Even if $\mathbb{A}$ were not completely accessible, a large random sample would suffice to provide a sufficiently accurate approximation. Figure 2.2 illustrates this strategy for a small example.

In practice, the computation of rmp_V is complicated by two issues:

1. In many cases, no shoes in $\mathbb{A}_{\mathcal{C}^y}$ (other than the suspect's shoe) are accessible by the examiner. Examiners are left to rely on previous experience and limited data (e.g. a small convenience sample from $\mathbb{A}$ or a related database) to make inferences regarding the conditional distribution of $x^s | s \sim \mathbb{A}_{\mathcal{C}^y}$. Historically, these inferences have been based on heuristics that lack empirical support (PCAST, 2016).

2. Determining if $x^y \equiv x^s$ is complicated by three phenomena: (i) a shoe's detected accidental locations are known to vary slightly each time it is printed (Shor et al., 2017), meaning that the locations in x^y may only approximate those in x^s, (ii) some accidentals do not reliably show up on crime scene prints (Richetelli et al., 2017b), meaning that the accidentals in x^y could be a thinned version of x^s, and (iii) test impressions may not be obtained until long after the crime was committed, leaving the opportunity for new accidentals to arise (Wyatt et al., 2005) or existing accidentals to change (Sheets et al., 2013) in the meantime.

We concentrate on issue 1 in this work, developing a more principled approach to inferring the distribution $x^s | s \sim \mathbb{A}_{\mathcal{C}^y}$ using the JESA database. Issue 2 is beyond the scope of this work, as determining an appropriate definition of $x^y \equiv x^s$ would require much richer data than is currently available in the literature. However,

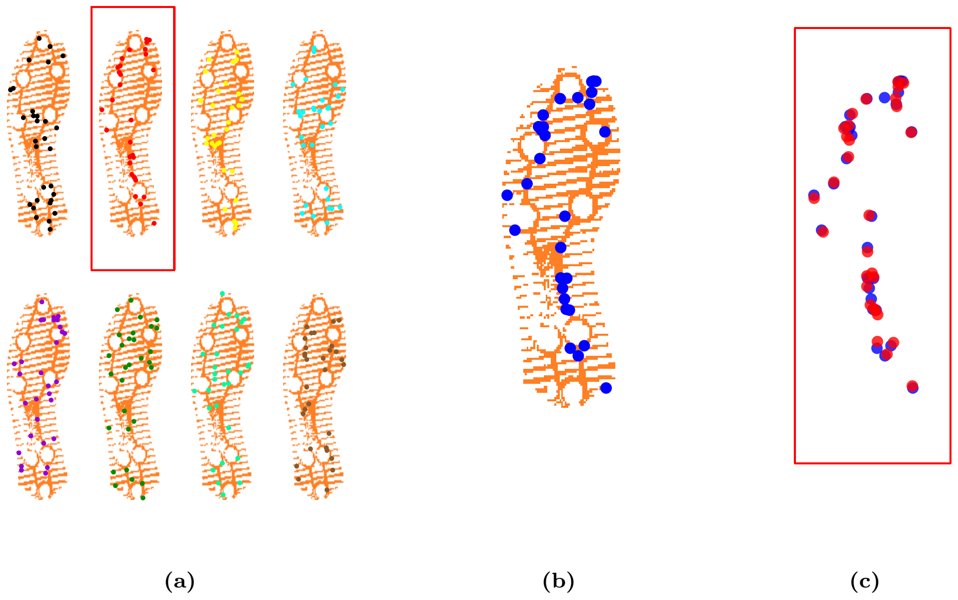


Figure 2.2: (a) depicts the accidental locations (blue) and contact surface (orange) for eight synthetic draws from the population $\mathbb{A}_{\mathcal{C}^y}$ corresponding to the crime scene print y shown in Figure 2.1a. (b) depicts the contact surface $\mathcal{C}^y$ (orange) and accidental locations x^y (blue). (c) illustrates the close correspondence between x^y (blue) and x^s (red) given by the accidental locations from the rectangle enclosed shoe in (a).

given a definition of $x^y \equiv x^s$, our model can compute the RMP via Monte Carlo. Figure 2.2 demonstrates this process with Figure 2.2a depicting the samples drawn from the distribution $x^s|s \sim \mathbb{A}_{\mathcal{C}^y}$.

2.2.2 JESA

The Jerusalem Shoeprint Accidentals Database (JESA) is one of a series of datasets created by the Israel Police Department's Division of Forensic Science. It pertains to 386 men's shoes collected as evidence through casework. A full description of the database is available in Yekutieli et al. (2012). For each shoe, there are two data structures relevant to our work – the *standardized shoeprint image* (contact surface) and the *accidentals*.

Standardized Shoeprint Image

Test impressions for each shoe were obtained by applying orange powder to their soles, pressing them onto clear films, then digitally photographing the residual orange impressions on the films. An example impression image is shown as Figure 2.1c.

For consistency across shoes, each image was standardized onto a 200 by 100 grid. Standardization involved translating, aligning, and scaling the images so the prints were centered, pointed upwards, and of the same length. The axes for the alignment were designated through point-and-click software by trained examiners. All left shoes were mirrored to appear as right shoes. Alignment of the images facilitates the pooling of information across shoes, even if they differ in size or chirality (i.e. left shoe or right shoe).

After standardization, the images were smoothed and de-noised to isolate the *contact surface*— the areas of the shoe sole that typically touch the ground. The smoothing was performed to preserve the shoe's tread pattern and general wear while filtering out small breaks due to accidentals or imperfections in the impression. These contact surfaces take the form of 200 by 100 binary arrays, with each bit defining contact or non-contact of a region of the shoe. Figure 2.1d illustrates the positive values of contact surface for the shoe in Figure 2.1b. The superimposed points are the locations of accidentals.

Additional example contact surfaces are shown in Figures 2.3a, 2.3b, and 2.3c, demonstrating the variety of tread patterns in the JESA database. No two contact surfaces in the JESA database are exactly alike, although those that correspond to the same brand of shoe are similar (differences in wear patterns, as well as variation in test impressions, account for the differences).

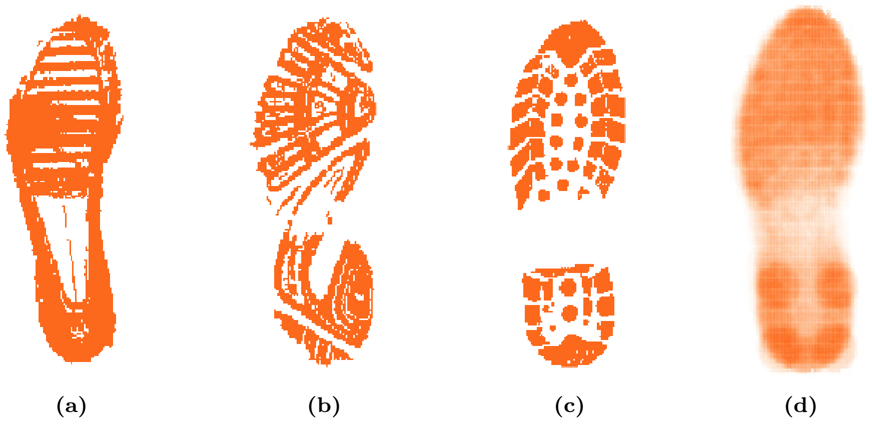


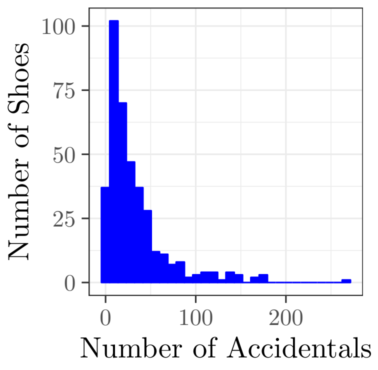
Figure 2.3: (a),(b),(c) are standardized test impressions from the JESA database, (d) is the mean test impression across the entire JESA database.

Figure 2.3d depicts the average contact surface across the entire database. It shows that it is far more common for regions of the shoe corresponding the heel and toes to be part of the contact surface than regions corresponding to the shoe arch. This discrepancy drives home the importance of conditioning on contact surface when evaluating accidental-based RMPs; shoes with arches that do make contact with the ground (the minority) would likely have different accidental distributions than those that do not. We use $\mathbb{C} = \{0,1\}^{100\times 200}$ to denote the space of values that a contact surface can take, and $\mathcal{C}^s \in \mathbb{C}$ to denote the contact surface of shoe s.

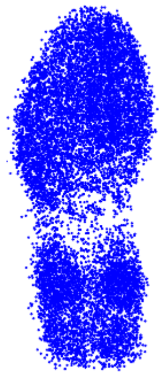
Accidentals

For each shoe, examiners identified accidentals by inspecting the shoeprint image and the shoe sole itself. The locations of the centroids of the accidentals were recorded using a computerized system. These locations were stored as real numbers in $[0, 100] \times [0, 200]$ corresponding to the standardized space of the contact surface. The region $[0, 1] \times [0, 1]$ corresponds to the bottom left hand corner of the standardized grid, and $[99, 100] \times [199, 200]$ corresponds to the top right. Figure 2.1d gives an example of the locations accidentals as points on the shoeprint image.

The number of accidentals, and their locations, varies from shoe to shoe. Figure 2.4a provides a histogram of the number of accidentals on each shoe. The distribution is heavily skewed to the right– the median number of accidentals is 20, whereas the mean is 33, and the maximum is 268.



(a)



(b)

Figure 2.4: (a) is a histogram summarizing the number of accidentals on each shoe in JESA. (b) illustrates the locations of these accidentals with points.

Figure 2.4b aggregates the coordinates of all accidentals recorded in the JESA database. Its similarity to that of Figure 2.3d is consistent with the intuition that accidentals should appear more frequently in areas of the shoe which are part of the contact surface. However, not all accidental locations fall on the sole with contact surface. Of the accidentals in JESA, about 12 percent of them occur in grid points without contact. Therefore, a robust model should be able to assign probability to situations in which accidentals do not occur directly on the contact surface. Examples of shoes in JESA for which accidentals occur away from the contact surface are available as Figures 2.13a and 2.13b. Following Damary et al. (2018), we exclude rift-type accidentals from our analysis because they occur only on specific type of shoe tread, making their spatial distribution markedly different than the more frequently occurring types of accidentals (e.g. hole or scratch).

2.2.3 Existing Models for the Distribution of Accidentals

Going forward, we use the shorthand $x^s|\mathcal{C}^s$ to refer the distribution of accidental locations x^s on a shoe s with contact surface $\mathcal{C}^s$, with $\mathcal{C}^s = \mathcal{C}^y$ referring to the distribution required to compute the RMP (2.6). For easy comparison of existing models in the literature with the approach we develop, we use a unified notation.

We begin by treating each $x^s|\mathcal{C}^s$ as a draw from a 2-dimensional spatial point process (Daley and Vere-Jones, 2007) over the standardized space $[0, 100] \times [0, 200]$. For our model, we make three additional assumptions regarding the structure of these point processes: (1) the individual accidentals $(x^s_n)_{n=1 \ldots N_s}$ are exchangeable, (2) the marginal distribution of each x^s_n is independent of the total number of accidentals N_s, and (3) the distribution of x^s depends on s only through the contact surface $\mathcal{C}^s$. The first two assumptions are common in the literature, whereas the third is unique to our model because we are first to incorporate the contact surface.

Following assumptions (1) and (2), $(x^s_n)_{n=1 \ldots N_s}$ can be treated as independent draws from a random probability measure Λ_s on $[0, 100] \times [0, 200]$. The literature has mostly focused on universal models for Λ_s, assuming a fixed Λ that is common to all shoes $s \in \mathbb{A}$. Stone (2006) proposed a uniform model for Λ, i.e. $\Lambda \propto 1$. This assumption has been criticized for its lack of empirical support, as noted by PCAST (2016). Yekutieli et al. (2012) instead inferred Λ using a kernel density estimator on the accidentals in JESA (Section 2.2.2). Speir et al. (2016) applied a similar histogram estimator to a different database, yielding comparable results.

Because estimating a single Λ does not allow for conditioning on class characteristics or wear, these approaches implicitly assume that a shoe's accidental locations are independent of its contact surface. Evidence against this assumption was provided by Damary et al. (2018); their analysis of multiple replicates of three different tread patterns appearing in the JESA database revealed that different tread patterns tend to yield different accidental distributions. Therefore, having distinct Λ_s that depend on $\mathcal{C}^s$ seems more appropriate, serving as the motivation for our assumption (3) above.

We encode assumption (3) in our model by explicitly treating each Λ_s as a draw from a distribution $G_{\mathcal{C}^s}$. As the notation suggests, $G_{\mathcal{C}^s} = G_{\mathcal{C}^{s'}}$ if $\mathcal{C}^s = \mathcal{C}^{s'}$, but the distributions of Λ_s and $\Lambda_{s'}$ can differ otherwise. Other works have followed a similar line of thought by restricting analysis to a single type of shoe at a time (Adair et al., 2007; Petraco et al., 2010; Wilson, 2012). In each of those studies, several replicates of the exact same pair of shoes were worn independently for a period of time, after which their accidental locations were annotated, analyzed, and compared. This allowed for the identification of common trends for one specific type of shoe. Though such data is ideal for modeling $G_{\mathcal{C}^s}$, the approach cannot be practically scaled to all types of shoes. Collecting multiple annotated observations for all given tread patterns is prohibitively expensive. In addition, the project would have to continue in perpetuity, continually updating the database to account for the ever-growing list of footwear styles and brands.

For this reason, we propose a more general and scalable approach in our modeling of Λ_s. Instead of developing independent models $G_{\mathcal{C}^s}$ for each unique contact surface, we propose a Bayesian hierarchical model to pool information across many contact surfaces at once. Let $\mathbb{C}$ denote the space of possible contact surfaces. Our goal is to infer the entire family of distributions $G = (G_{\mathcal{C}})_{\mathcal{C} \in \mathbb{C}}$ as a single model, treating a shoe's contact surface $\mathcal{C}$ as a covariate. This joint modeling approach helps to leverage the information available in heterogeneous databases — in our case the JESA database — to identify the relationship between the contact surface and accidental locations and to capture commonalities that span across many shoe types.

Let $\mathbb{J}$ denote a set of available shoes (e.g. JESA) used to infer G. Then $(\Lambda_s)_{s \in \mathbb{J}}$ is a vector of dependent random probability measures, with the dependence between them induced by a hierarchical model on G. We now review existing approaches for modeling vectors of dependent probability measures, limiting our discussion to that which is most relevant to our model. We defer discussion of additional related work to Section 2.7.2.

2.2.4 Random Vectors of Dependent Probability Measures

Over the years, there has been a broad interest in modeling dependent probability measures, especially via nonparametric Bayes (Hjort et al., 2010; Foti and Williamson, 2015). The approach we use to model $(\Lambda_s)_{s \in \mathbb{J}}$ in this work is not fully nonparametric, but it is a finite-resolution approximation of one. Thus, it is natural to frame our review within the nonparametric Bayesian literature.

The canonical Bayesian nonparametric approach to modeling a measure μ on a space Ω is to treat it as a random draw from some subclass of measures on Ω. Completely random measures (Kingman, 1967) are an especially tractable subclass of random measures that are composed of a (possibly countably infinite) collection of weighted atoms in Ω. We use $(\theta_i)_{i=1,\ldots,\infty} \in \Omega^\infty$ to denote the locations of the atoms of the completely random measure μ, and $(w_i)_{i=1,\ldots,\infty} \in \mathbb{R}_+^\infty$ to denote the corresponding (non-negative) atom weights. The defining feature of a completely random measure is that, for any disjoint subsets $\Omega_1, \Omega_2 \subset \Omega$, $\mu(\Omega_1)$ is independent of $\mu(\Omega_2)$ (complete randomness). An accessible review of completely random measures as they pertain to statistical modeling is available in Jordan (2010).

For our purposes, we are interested in atomic measures that do not necessarily satisfy the complete randomness assumption. In particular, we are interested in atomic random *probability* measures — random measures μ consisting of atoms such that $\mu(\Omega) = 1$. Any finite atomic random measure can be converted to a probability measure via normalization. For instance, a normalized completely random measure takes the form

$$\bar{\mu}(\cdot) = \frac{\sum_{i=1}^{\infty} w_i \delta_{\theta_i}(\cdot)}{\sum_{i=1}^{\infty} w_i}. \tag{2.7}$$

where w_i, θ_i are defined analogously to above. The strength of atomic probability measures is that they can be convolved with probability kernels to define mixture models for densities (e.g. Escobar and West (1995), Rasmussen (2000)). Each atom acts as its own mixture component, providing a framework that is flexible and computationally tractable.

Rather than a single normalized random measure, we are concerned with a vector of dependent random probability measures $(\Lambda_s)_{s\in\mathbb{J}}$ that can capture commonalities across all shoes in JESA. Particularly relevant to our work is the recently proposed normalized compound random measure framework (NCoRM) of Griffin and Leisen (2017), which formulates the vector of random probability measures $\mu_1, \ldots, \mu_K$ on Ω as

$$\mu_k(\cdot) = \frac{\sum_{i=1}^{\infty} m_i^k w_i \delta_{\theta_i}(\cdot)}{\sum_{i=1}^{\infty} m_i^k w_i} \tag{2.8}$$

where $(\theta_i, w_i)_{i=1,\ldots,\infty}$ are drawn as in a single completely random measure and $(m_i^k)_{i=1,\ldots,\infty}$ are iid random "score" variables for $k = 1, \ldots, K$, following a distribution ρ, that up-weight or down-weight the shared set of atoms defined by the (θ_i, w_i) for each of the μ_k's. The distribution of the scores controls the strength of the dependence, with much of the exposition in Griffin and Leisen (2017) devoted to gamma distributions due to their computational tractability. We use the idea of scoring in normalized atomic random measures to develop our model. However, modifications must be made.

The NCoRM approach as described in Griffin and Leisen (2017) was developed for exchangeable vectors of random probability measures. However, exchangeability does not hold when each measure has an associated covariate (as we have in the contact surfaces $\mathcal{C}^s$). For this reason, we generalize the idea of "scoring" from NCoRMs to the non-exchangeable setting, allowing us to incorporate covariate information. It is worth noting that Griffin et al. (2018) also generalizes the NCoRM framework to a non-exchangeable regression framework, but differently than we do here.

2.3 Model

Recall that for a given shoe $s \in \mathbb{J}$, we have assumed each accidental location x_n^s is drawn independently from a probability measure Λ_s on $[0, 100] \times [0, 200]$ where Λ_s itself is randomly drawn from a distribution $G_{\mathcal{C}^s}$ that depends on the contact surface $\mathcal{C}^s \in \mathbb{C}$. Because it is impractical to independently model $G_{\mathcal{C}}$ for all possible $\mathcal{C} \in \mathbb{C}$, we develop a hierarchical model to jointly infer all entries of G, treating each $\mathcal{C} \in \mathbb{C}$ as a high-dimensional spatial covariate.

Before specifying how we model the family of distributions G, it is useful to first address the limited precision of the data. As per §2.2.2, the contact surface variables $\mathcal{C} \in \mathbb{C}$ are defined on a discrete 200 by 100 equally-spaced grid over $[0, 100] \times [0, 200]$. We use $\mathcal{A}$ to denote the set of entries in this grid:

$$\mathcal{A} = \{(a_1, a_2) : a_1 \in \{1, \dots, 100\}, a_2 \in \{1, \dots, 200\}\} \tag{2.9}$$

with gridpoint $(a_1, a_2) \in \mathcal{A}$ corresponding to the area $(a_1 - 1, a_1] \times (a_2 - 1, a_2]$ in $[0, 100] \times [0, 200]$. We restrict our model for Λ_s to have the same resolution as $\mathcal{A}$ by discretizing Λ_s to be a piece-wise constant over each gridpoint in $\mathcal{A}$. This reduced resolution provides computational advantages, simplifies interpretation, and guards against overfitting. Further discussion of the discretization is available in Section 2.7.1.

After discretization, each Λ_s can be characterized by the values it takes at the grid points in $\mathcal{A}$, and each $G_{\mathcal{C}} \in G$ can be characterized by the multivariate distribution it assigns to those grid points. This provides a natural representation for parametrizing our model — we view G as a family of distributions over the 20000-dimensional simplex indexed by $\mathbb{C}$, with each $G_{\mathcal{C}^s}$ characterized by the joint distribution it defines over the vector of values in the probability measure $\Lambda_s|\mathcal{C}^s$. It is most straightforward to describe G in terms of the generative process it assigns to a generic $\Lambda_s|\mathcal{C}^s$, as we do below.

2.3.1 Parameterization of Λ_s

We model each measure $\Lambda_s \sim G_{\mathcal{C}^s}$ as the convolution of a normalized random atomic measure μ_s with a two dimensional piece-wise constant probability kernel k. We define μ_s to consist of 20000 atoms at fixed locations — one for each gridpoint in $\mathcal{A}$. To model the weights of each of these atoms, we generalize the NCoRM scoring technique of Griffin and Leisen (2017) to incorporate the covariate information in $\mathcal{C}^s$, and to allow for spatial dependence between atom weights.

For each $a \in \mathcal{A}$, we define the distribution of $\mu_s|\mathcal{C}^s$ as

$$\mu_s(a) = \frac{w_a m_a^s}{\sum_{b \in \mathcal{A}} w_b m_b^s} = \frac{w_a \epsilon_a^s \phi_a^s}{\sum_{b \in \mathcal{A}} w_b \epsilon_b^s \phi_b}. \tag{2.10}$$

Here, $(w_a)_{a \in \mathcal{A}}$ are parameters common to all G, and $(m_a^s)_{a \in \mathcal{A}}$ are random shoe-specific location-specific scores applied to the weights of the atoms. The scores further decompose into two components: $m_a^s = \epsilon_a^s \phi_a^s$, with ϵ_a representing "traditional" scores as in NCoRM (assumed to be independent for all shoes and all locations), and ϕ_a^s representing contact-dependent scores — variables that depend on the nearby configuration of $\mathcal{C}^s$. We model the traditional scores as independent draws from $\rho_q = \text{Gamma}(q,\ 1)$. The contact-dependent scores ϕ_b^s are treated as parameters, defined as follows.

Let $\phi \in [0,1]^{32}$. For all $a \in \mathcal{A}$, $s \in \mathbb{J}$ define

$$\phi_a^s = \phi_{r_a^s} \text{ where} \tag{2.11}$$

$$r_a^s = 1 + \sum_{i=-1}^{1} \sum_{j=-1}^{1} 2^{3+i+2j} \mathcal{C}_{a+(i,j)}^s I(||(i,j)||^2 \leq 1). \tag{2.12}$$

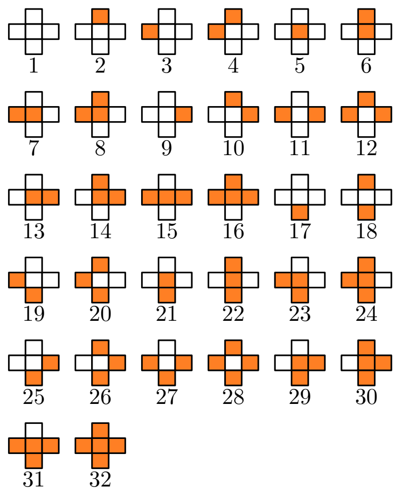
By this formulation, ϕ_a^s takes one of $2^5 = 32$ values depending on the value of the contact surface at the gridpoints surrounding a. For instance, if a is completely surrounded by contact surface, i.e.

$$\mathcal{C}_{a+(-1,0)} = \mathcal{C}_{a+(0,-1)} = \mathcal{C}_a = \mathcal{C}_{a+(1,0)} = \mathcal{C}_{a+(0,1)} = 1, \tag{2.13}$$

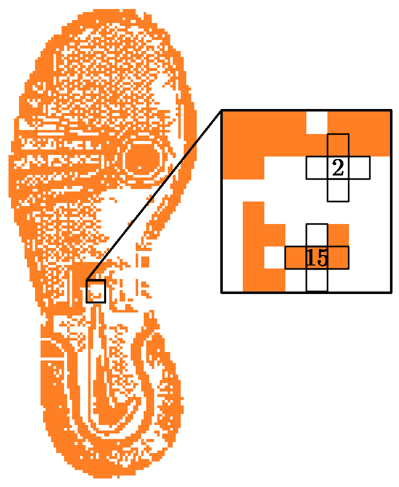
then $\phi_a^s = \phi_{32}$. Similarly, if a is in an area devoid of contact surface, i.e.

$$\mathcal{C}_{a+(-1,0)} = \mathcal{C}_{a+(0,-1)} = \mathcal{C}_a = \mathcal{C}_{a+(1,0)} = \mathcal{C}_{a+(0,1)} = 0, \tag{2.14}$$

then $\phi_a^s = \phi_1$. A demonstration of the possible configurations is provided in Figure 2.5a along with an depiction of r_a^s for two $a \in \mathcal{A}$ in Figure 2.5b.



(a)



(b)

Figure 2.5: (a) provides a list of the possible shapes the contact surface can take around an atom, accompanied by the index in $\phi \in [0,1]^{32}$ to which it corresponds. (b) zooms in on an example shoe's contact surface (zoomed region outlined in black) to demonstrate the ϕ_a^s value of two example locations.

Before specifying the functional form for the kernel k (which smooths the atom weights), let us first interpret of the various components that define the atoms weights for μ_s in the context of the shoe sole and accidentals.

The weights are the normalized product of three components:

1. ϕ, which specifies the impact of a gridpoint's surrounding contact surface on the relative likelihood of accidental occurrence,

2. w, which specifies the impact of the position of a gridpoint's spatial coordinates on the relative likelihood of accidental occurrence, and

3. ρ_q (parameterized by q), which specifies the variability across shoes of each gridpoint's relative probability of accidental occurrence, controlling for position and contact surface.

Essentially, the parameters ϕ and w control the mean of μ_s, whereas its variance depends on the ϵ_a^s scores — distributed according to ρ_q. These choices are in-line with a common belief in forensic footwear analysis — that the locations of accidentals tend to follow a spatially inhomogeneous distribution across the shoe sole (captured by w), and that some areas are more likely to be affected than others depending on their contact with the ground (captured by ϕ). We model each of ϕ, w, and q as global parameters, assuming they take the same value for all shoes JESA.

The random shoe-specific errors ϵ^s capture deviations from this common trend. The coefficient of variation of ρ_q — given by $q^{-1/2}$ — indicates the strength of the deviations. The smaller the value of q, the larger the variation of μ_s around its mean.

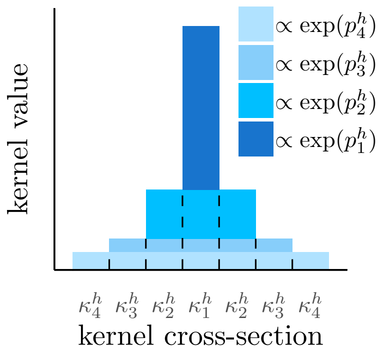
Finally, we convolve all atoms in all μ_s with a kernel k to obtain Λ_s. The kernel is parameterized to smooth the weights across nearby atoms. Recognizing that the smoothing should be local, we define the kernel k to have finite support, symmetrically redistributing the mass over a window extending three grid points from a in all four axis-aligned directions. Figure 2.6a illustrates the shape of the probability kernel. We refer to this parameterization as the *tiered cake* representation due to the resultant kernel resembling a tiered cake with p^α controlling the size of each tier.

We parameterize k as a function $k : \{-3, \dots, 3\}^2 \to [0, 1]$ such that

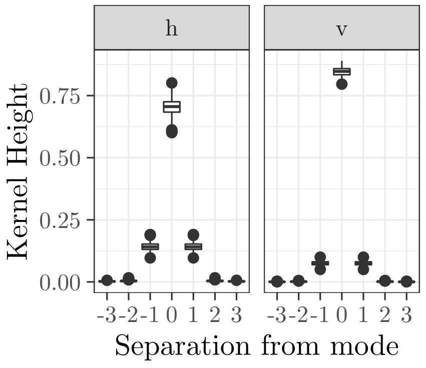
$$k(i, j) = \kappa^h_{1+|i|}\kappa^v_{1+|j|}. \tag{2.15}$$

Here, $\kappa^h, \kappa^v \in [0, 1]^4$ define independent symmetric kernels in the horizontal and vertical directions, and k is their composition. To ensure that κ^h and κ^v are unimodal probability kernels, we further re-parameterize them as

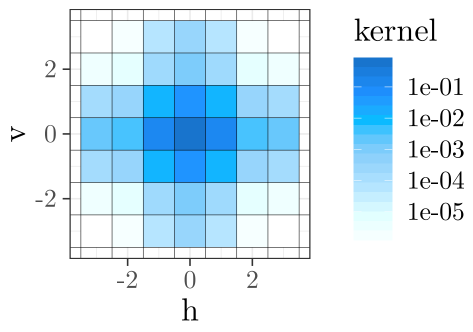
$$\kappa_i^\alpha = \frac{\sum_{j=i}^4 \exp(p_j^\alpha)/(2j-1)}{\sum_{j=1}^4 \exp(p_j^\alpha)}, \tag{2.16}$$



(a)



(b)



(c)

Figure 2.6: (a) illustrates the tiered cake parametrization of κ^h. Each uniquely colored tier is proportional to the corresponding $\exp p_i^h$, with the dotted lines depicting how the cake is sliced that form each κ_i^h. (b) demonstrates the posterior fit of κ^h and κ^v using symmetrically arranged boxplots. (c) depicts the posterior mean of $k(i,j) = \kappa_i^h \kappa_j^v$ centered $(0,0)$. The decay in the h direction controlled by κ^h, and the decay in the vertical direction (v) controlled by κ^v, with the hue changing according to a logarithmic scale.

for $i = 1, \ldots, 4$, $\alpha = v, h$, and each $p^\alpha \in \mathbb{R}^4$. Note that our fitted results (Figure 2.6b) indicate that extending the window for three grid points appears to be excessive, but parameterizing three allowed for such a discovery. Going forward, we will often suppress the p^α parameterization to make the presentation more concise, instead relying on the κ^α representation.

2.3.2 Model Summary and Prior

Having parametrized G, we now formulate the full hierarchical Bayesian model. Let Θ denote the concatenation of the global parameters ϕ, w, q, p^h, and p^v. Our prior distribution on Θ is the composition of independent priors on its entries. Letting $\text{MVN}(0, 4I_4)$ denote 4-dimensional isotropic Gaussian distribution with variance 4, and $\text{MVLN}(0, \Sigma)$ denote a multivariate log normal distribution with mean parameter 0 and precision matrix Σ, the following provides a bird's eye view of the model via the generative process of the

JESA data given $(\mathcal{C}_s, N_s)_{s\in\mathbb{J}}$:

Step 1: Generate global parameters:

$$q \sim \text{Gamma}(2,2), \qquad w_E \sim \text{MVLN}(0,\Sigma),$$
$$\phi \sim \text{unif}([0,1]^{32}), \qquad p^h, p^v \sim \text{MVN}(0, 4I_4).$$

Step 2: Generate the densities $\Lambda_s \sim G_{\mathcal{C}^s}$ for $s \in \mathbb{J}$:

$$\text{For } a \in \mathcal{A}:$$
$$\epsilon_a^j \sim \text{Gamma}(q,1),$$
$$\Lambda_s(a) = \sum_{-3\le i,j\le 3} \kappa^h_{1+|i|}\kappa^v_{1+|j|} \frac{w_{a+(i,j)}\epsilon^s_{a+(i,j)}\phi^s_{a+(i,j)}}{\sum_{a'\in\mathcal{A}'} w_{a'}\epsilon^s_{a'}\phi^s_{a'}}.$$

Step 3: Generate the accidental Locations x^s for $s \in \mathbb{J}$.

$$\text{For } n = 1,\ldots,N_s: \text{ generate } x^s_n \sim \Lambda_s.$$

Our prior on w in Step 1 uses a coarsened representation, the details of which are provided in Section 2.7.3. The vector w_E denotes the subvector of unique values after the coarsening. The scales for all of the priors were chosen based on the range of possible behaviors we expected in the model. For example, the variances of 4 for p^h and p^v were chosen to strike a balance: too small of a variance concentrates the kernel around its geometric decaying mean, and too large of a variance places most of the prior density kernels that are essentially step functions. For $\phi \in [0,1]^{32}$, the upper bound on the uniforms is arbitrary — the likelihood in (2.18) is invariant to scalings of ϕ due to the normalization of μ_s. The rate of ρ_q is fixed at 1 for the same reason.

2.4 Computation

There are two key computational challenges associated with our model.

1. How do we efficiently compute the posterior of Θ?
2. How do we efficiently compute the density of an observed set of accidentals x^s given $\mathcal{C}^s$?

Task 1 (addressed in §2.4.1) arises when fitting our model to the JESA data, and task 2 (addressed in §2.4.2) arises when evaluating models. Before describing our strategies for addressing these tasks, we develop a trick to compute the likelihood of $x^s \in ([0,100]\times[0,200])^{N_s}$ given $\mathcal{C}^s$ for a given Θ.

The raw likelihood takes the form

$$p(x^s|\mathcal{C}^s;\Theta) = \int \prod_{n=1}^{N_s} \Lambda(x_n^s) G_{\mathcal{C}^s}(\mathrm{d}\Lambda) \tag{2.17}$$

$$= \int \prod_{n=1}^{N_s} \sum_{-3\le i,j\le 3} \kappa^h_{1+|i|}\kappa^v_{1+|j|} \frac{w_{x_n^s+(i,j)}\epsilon^s_{x_n^s+(i,j)}\phi^s_{x_n^s+(i,j)}}{\sum_{a\in\mathcal{A}} w_a\epsilon_a^s\phi_a^s} \mathrm{d}\rho(\epsilon^s). \tag{2.18}$$

In a slight abuse of notation, we have overloaded x_n^s to also denote the atom $a \in \mathcal{A}$ to which the real-valued $x_n^s \in [0, 100] \times [0, 200]$ is associated. At first glance, the $|\mathcal{A}|$-dimensional integral over the ϵ^s variable in (2.18) appears to be computationally intractable. It has no closed form, and is too high dimensional to efficiently compute using quadrature or generic Monte Carlo algorithms. To overcome this problem, we introduce auxiliary variables.

For each accidental location x_n^s on shoe $s \in \mathbb{J}$, we define Z_n^s by

$$\mathbb{P}(Z_n^s = x_n^s + (i,j) \mid \kappa^h, \kappa^v) = k(i,j) = \kappa^h_{1+|i|}\kappa^v_{1+|j|}, \tag{2.19}$$

with κ^v, κ^h being the kernel parameters as defined in (2.15), and each $x_n^s \in \mathcal{A}$. We use the shorthand Z^s to refer to the collection $(Z_n^s)_{1\le n\le N_s}$ and use C_a^s to denote the number of times each $a \in \mathcal{A}$ occurs in Z^s. We also introduce the auxiliary variables

$$u^s \sim \mathrm{Gamma}\left(N_s, \sum_{a\in\mathcal{A}} w_a\epsilon_a^s\phi_a^s\right), \tag{2.20}$$

with Gamma(α, β) denoting a gamma distribution with shape α and rate β. We can now analytically marginalize the ϵ^s variables to obtain

$$p(x^s|\mathcal{C}^s;\Theta) = \int_0^\infty \frac{u^{N_s-1}}{\Gamma(N_s)} \mathbb{E}\left(\frac{1}{\Gamma(q)^{|\mathcal{A}|}} \prod_{a\in\mathcal{A}} \frac{\Gamma(q + C_a^s)\,(w_a\phi_a^s)^{C_a}}{(u^s w_a\phi_a^s + 1)^{q+C_a^s}}\right) \mathrm{d}u^s, \tag{2.21}$$

where $\Gamma(\cdot)$ denotes the gamma function and $\mathbb{E}$ denotes an expectation taken with respect to the distribution of Z^s as given in (2.19). By swapping (2.18) for (2.21), we have exchanged a $|\mathcal{A}|$-dimensional integral over ϵ^s for a more tractable one dimensional integral. The full derivation of moving from (2.18) to (2.21) is provided in Section 2.7.4.

This new expression for the marginal likelihood (2.21) enables us to address challenges (1) and (2) using Monte Carlo algorithms, relying on Markov chain Monte Carlo (MCMC) and importance sampling, respectively. For background information regarding MCMC and importance sampling, we refer the reader to Brooks et al. (2011) and Tokdar and Kass (2010).

2.4.1 Computing the Posterior for Θ

We consider an augmented version of the posterior that instantiates the auxiliary variables $Z = (Z^s)_{s\in\mathbb{J}}$ and $U = (u^s)_{s\in\mathbb{J}}$. We use $\mathcal{L}(\Theta, Z, U)$ to denote the augmented likelihood

$$\mathcal{L}(\Theta, Z, U) = \frac{1}{\Gamma(q)^{|\mathbb{J}||\mathcal{A}|}} \prod_{s\in\mathbb{J}} \frac{u^{N_s-1}}{\Gamma(N_s)} \prod_{a\in\mathcal{A}} \frac{\Gamma(q + C_a^s)\,(w_a\phi_a^s)^{C_a^s}}{(u^s w_a \phi_a^s + 1)^{q+C_a^s}} \prod_{n=1}^{N_s} k(\Delta_n^s), \tag{2.22}$$

where Δ_n^s and C_a^s are defined as in (2.21). Our target is the posterior distribution Θ, U, Z, with density $p(\Theta, U, Z|(x^s, \mathcal{C}^s)_{s\in\mathbb{J}})$ satisfying:

$$p(\Theta, U, Z|(x^s, \mathcal{C}^s)_{s\in\mathbb{J}}) \propto \mathcal{L}(\Theta, Z, U)\, p(\Theta). \tag{2.23}$$

Our MCMC algorithm consists of sequential updates of the parameters — akin to Metropolis within Gibbs — with most of the components being updated according to slice sampling (Neal, 2003; Murray et al., 2010). The updates are repeatedly performed in the following sequence:

- Each auxiliary variable $(u^s)_{s\in\mathbb{J}}$ is updated one-by-one using slice sampling. These updates can be performed in parallel.
- The entire vector w is updated jointly using elliptical slice sampling.
- Each entry in $(\psi_i)_{i=1,\dots,32}$ is updated one-by-one using slice sampling.
- The parameter q is updated using a slice sampler.
- Each entry in p^h then p^v is updated one-by-one using slice sampling.
- Each auxiliary variable (z_n^s) is updated one-by-one by Gibbs sampling.

The details and conditional distributions for these updates are available in the Section 2.7.5. This algorithm provides a sequence of draws of Θ from its posterior that can be used to approximate posterior expectations. Notably, we can use these to approximate the posterior marginal probability of a configuration of accidentals (Task 2) as we now detail in §2.4.2.

2.4.2 Computing Marginal Densities via Importance Sampling

A natural metric for assessing the performance of our model is to split $\mathbb{J}$ into a training set T and test set T', then evaluate the held out density of the accidental locations on each shoe in T' (given T). Doing this requires computing

$$p(x^\tau \mid \mathcal{C}^\tau, T) = \mathbb{E}_\Theta\left(p(x^\tau|\mathcal{C}^\tau, \Theta) \mid (x^s, \mathcal{C}^s)_{s\in T}\right) \tag{2.24}$$

for each $\tau \in T'$, where $p(\cdot \mid \mathcal{C}^\tau, T)$ denotes the posterior density. Here, $\mathbb{E}_\Theta(\cdot|(x^s, \mathcal{C}^s)_{s\in T})$ denotes the expected value under the posterior of Θ given the contact surfaces and accidentals in T. Note that the nested integrals in the expression in (2.24) can be separated into an outer integral and an inner integral. The outer integral is the posterior expectation over the global parameters Θ and can be approximated using MCMC draws as described above. The inner integral — computed for each posterior draw — is over the local auxiliary variables u^τ and Z^τ as shown in (2.21).

We approximate this integral using importance sampling. Specifically, given a draw of Θ, we define an importance distribution given by

$$u \mid \Theta, N_s, \mathcal{C}^\tau \sim \text{Gamma}\left(N_s, q\sum_{a\in\mathcal{A}} w_a\phi_a^\tau\right) \tag{2.25}$$

$$\mathbb{P}(Z_n = x_n^\tau + a \mid \Theta, x_n^\tau) = \frac{w_{a+x_n^\tau}\phi^\tau_{a+x_n^\tau}k(a)}{\sum_{b\in B} w_{b+x_n^\tau}\phi^\tau_{b+x_n^\tau}k(b)} \tag{2.26}$$

where $B = \{-3, \ldots, 3\}^2$ and $a \in B$ for all $n \in \{1, \ldots, N_s\}$. After drawing $M > 0$ importance samples $u_1, \ldots, u_M \in \mathbb{R}_+$ by (2.25) and $Z^1, \ldots, Z^M \in \mathcal{A}^{N_\tau}$ by (2.26), the inner integral can be approximated as

$$p\left(x^\tau|\mathcal{C}^\tau, \Theta\right) = \frac{\prod_{n=1}^{N_\tau}\left(\sum_{b\in B} w_{b+x_n^\tau}\phi^\tau_{b+x_n^\tau}k(b)\right)}{\Gamma(q)^{|\mathcal{A}|}\left(q\sum_{a\in\mathcal{A}} w_a\phi_a^\tau\right)^{N_\tau}}\sum_{m=1}^{M}\frac{\exp\left(u^m q\sum_{a\in\mathcal{A}} w_a\phi_a^\tau\right)}{\prod_{a\in\mathcal{A}}\frac{(u^m w_a\phi_a^\tau+1)^{q+C_a^m}}{\Gamma(C_a^m+q)}},$$

where C_a^M denotes the number of times $a \in \mathcal{A}$ occurs as an entry in Z^M.

Thus, using one importance sample ($M = 1$) for each MCMC draw $\Theta^\ell = (\phi^\ell, w^\ell, q^\ell, (p^h)^\ell, (p^v)^\ell)$ yields the approximation

$$p(x^\tau \mid \mathcal{C}^\tau, T) \approx \sum_{\ell=1}^{L}\frac{\prod_{n=1}^{N_\tau}\left(\sum_{b\in B} w_{b+x_n^\tau}(\phi^\ell)^\tau_{b+x_n^\tau}k^\ell(b)\right)}{L\Gamma(q^\ell)^{|\mathcal{A}|}\left(q^\ell\sum_{a\in\mathcal{A}} w_a(\phi^\ell)\tau_a\right)^{N_\tau}}\frac{\exp\left(u^\ell q^\ell\sum_{a\in\mathcal{A}} w_a^\ell(\phi^\ell)_a^\tau\right)}{\prod_{a\in\mathcal{A}}\frac{(u^\ell w_a^\ell(\phi^\ell)_a^\tau+1)^{q^\ell+C_a^\ell}}{\Gamma(C_a^\ell+q^\ell)}}, \tag{2.27}$$

where L is the total number of MCMC draws and the $(u^\ell, Z^\ell)_{1\leq\ell\leq L}$ are each drawn according to the respective importance distribution for Θ^ℓ. Detailed derivations and discussion of this strategy are available the Section 2.7.6.

2.5 Comparisons to Competitors and Summary of Fit

2.5.1 Comparison to Competitors

To demonstrate that efficacy of our model, we compare its performance to three competitor models. The first two models we consider – the uniform model of Stone (2006) and the kernel density estimator of Yekutieli et al. (2012) – rely on fitting a single fixed density Λ for all shoes. Recall from §2.2.3 that the kernel density estimator does not make use of contact surface information when estimating Λ, and that the uniform model does not rely on any data at all.

For this reason, we introduce a third competitor called the *contact model.* In the contact model, each $G_{\mathcal{C}^s}$ is defined as a point mass at $\Lambda_{\mathcal{C}^s}$ with

$$\Lambda_{\mathcal{C}^s}(a) \propto \exp(\alpha_{r_a^s}). \tag{2.28}$$

Here, $\alpha \in \mathbb{R}^{32}$ are shared amongst all of G, similar to ϕ with r_a^s following the same set-up as defined as in (2.12). The parameters α are straightforward to infer using maximum likelihood (fixing $\alpha_1 = 1$ to obtain identifiability).

We fit our model and the three competitor models to four test/train splits of the JESA data, with each training set consisting of 336 randomly selected shoes. The remaining 50 serve as the test set. For our model, the posterior was computed by running the MCMC algorithm outlined in Section 2.4.1 for 30000 full sweeps and discarding the first 10000 iterations as warm-up.

Let T denote a training set and T' denote the test set. As a metric of performance, we used our importance sampling technique to evaluate the held-out density of the accidental locations x^τ on each shoe $\tau \in T'$ given T. Figure 2.7 depicts the held-out likelihood per accidental on each held-out shoe for each of the four models fit to each of the four splits. Specifically,

$$20000 \times p(x^\tau \mid \mathcal{C}^\tau, T)^{1/N_\tau} \tag{2.29}$$

is reported for each $\tau \in T'$. The scaling by 20000 is performed for readability of the y-axis (it is equivalent to transforming $\mathcal{A}$ to the unit square) and the N_τth root is taken to facilitate comparison of average performance on shoes with different numbers of accidentals. This metric is equivalent to comparing the per-accidental average log loss of each shoe. The held-out shoes were sorted according to our model's performance for each of the four splits. Note that for the uniform model, only those atoms in $\mathcal{A}$ were given positive density, hence the constant density of 1.743 rather than 1.

It is evident from Figure 2.7 that the two models that account for contact surface (our model and the simple contact surface model (2.28)) vastly outperform the two that do not. Notably, the kernel density estimator assigns 0 density to a shoe in splits 3 and 4, showing an alarming lack of robustness. The performance of our model and the contact model tend to track together across shoes, suggesting that the incorporation of the contact surface is the major driver of both models' success.

We also checked whether the other components of the model (w, κ, and ϵ) contribute positively to the model's performance. We fit an additional five variants of our model to the training data and summarized

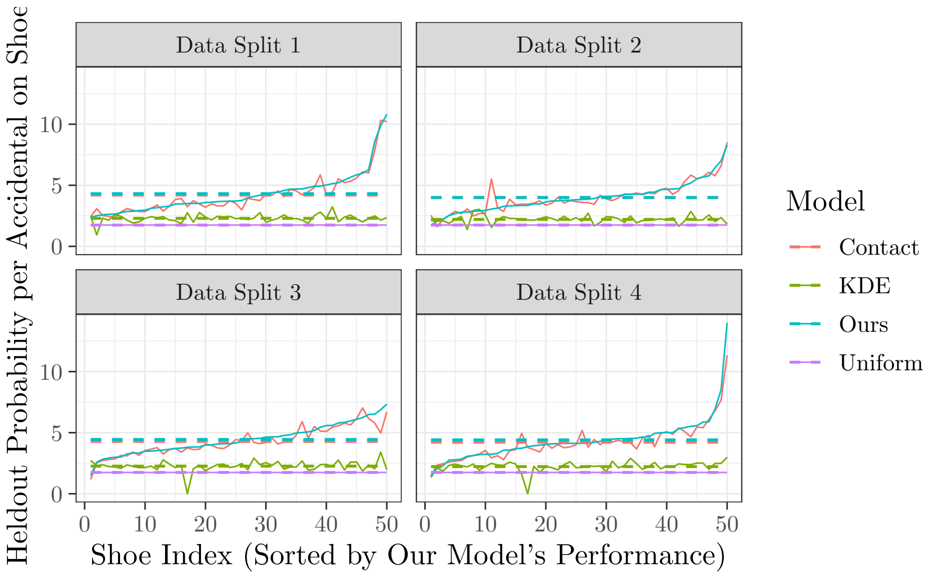


Figure 2.7: Comparison of the performance of four models: the contact model (red) the kernel density estimate (green), our model (blue) and the uniform model (purple) on 50 held out shoes across four data splits. The solid lines depict the metric given in (2.29) for each of 50 shoes (sorted by our model's performance). The dotted lines depict the mean for each model.

their results in Table 2.1, along with the performance of the four original competitors. The variant models are defined as follows. "Without scores" refers to our model with all ϵ_a^s variables are fixed at one, "without kernel" refers to our model but without k smoothing, "without scores and kernel" excludes both ϵ_a^s and k, "without w" fixes $w_E = 1$, and "without ϕ" fixes all ϕ at 1. Posterior computation for all variant models were performed using appropriate analogs of the MCMC algorithm given in §2.4.1.

For each model and test set T', Table 2.1 reports the geometric mean of (2.29) across all held-out shoes, i.e.

$$20000 \times \left(\prod_{\tau \in T'} p(x^\tau \mid \mathcal{C}^\tau, T)^{1/N_\tau} \right)^{1/|T'|}. \tag{2.30}$$

This metric is equivalent to the mean per-accidental log loss across shoes.

Table 2.1 demonstrates that our full model outperforms all competitors and variants on Splits 1, 3, and 4, being edged out only by "without w" on Split 2. Nearly all variants perform close to comparably to the full model; the notable exception is "without ϕ". It performs far worse, highlighting the importance of the contact surface. The persisting decrease in performance of the other variants across splits indicates that each component provides a small gain, and is worth keeping in the model.

Note that the superior performance of "without w" in Split 2 is explained by the presence of an atypical shoe in the test set. It possesses only two accidentals, both of which are located at the left side of the heel.

	Method	Split 1	Split 2	Split 3	Split 4
Existing Models	Uniform (Stone, 2006)	1.743	1.743	1.743	1.743
	KDE (Yekutieli et al., 2012)	2.266	2.182	0.000	0.000
Other Models	Contact	3.954	3.823	4.106	3.995
Our Model and variants	Full	**4.060**	3.832	**4.272**	**4.144**
	without scores	4.052	3.831	4.260	4.131
	without kernel	4.041	3.794	4.244	4.081
	without scores and kernel	4.039	3.791	4.238	4.072
	without w	3.981	**3.860**	4.131	4.070
	without ϕ	2.217	2.124	2.187	2.144

Table 2.1: The mean predictive performance (measured by (2.30)) of our model, five variants on our model, and three competitor models. The best performing result is bolded for each split.

As illustrated in Figure 2.9(w), w is small towards the heel, especially on the lefthand side. Consequently, including w leads to far lower predictive posterior probability for this particular shoe. Excluding this shoe from the test set 2 results in the full model regaining its spot as the top performer.

2.5.2 Summary of Inferred Model Parameters

To investigate our fitted model, we consider the posterior of Θ from Split 1 in §2.5.1. Components of the posterior distribution are summarized in Figures 2.6, 2.8, and 2.9.

Figure 2.6 summarizes the posterior fit for the kernel k. Figure 2.6a uses boxplots to demonstrate the posterior distribution of both κ^h and κ^v, arranged symmetrically to facilitate visualization of the kernel. For both h and v, the kernel's mass is mostly concentrated on its mode and immediate neighbours. The smoothing is also more diffuse in the horizontal direction that the vertical direction, suggesting that the accidental distributions are smoother in the horizontal direction that vertical direction. Figure 2.6c demonstrates the composition of the vertical and horizontal kernel into the bivariate kernel.

Figure 2.8 displays the marginal posterior distributions of each $\phi_1, \ldots, \phi_{32}$ using boxplots. Here, the larger the associated posterior value, the more likely an accidental is to occur nearby contact surface taking on the shape. There is a stark difference in accidental proclivity between gridpoints surrounded mostly by contact surface (shapes 32, 31, 30, 28, 24, 16 as depicted in Figure 2.5a) and those with little contact surface present (shapes 1, 2, 3, 5, 9, 17). This difference supports the intuition among shoeprint examiners that regions which rarely make contact with the ground are typically less likely to accumulate accidentals. Also notable is the discrepancy between different shapes containing the same amount of contact surface. For example, accidentals appear to be nearly to twice as likely to be associated with gridpoints exhibiting shape 31 than those exhibiting shape 24, even though both shapes consist of 4 of 5 possible contact components. This inference suggests the shape of the contact surface — and not just the amount of contact surface —

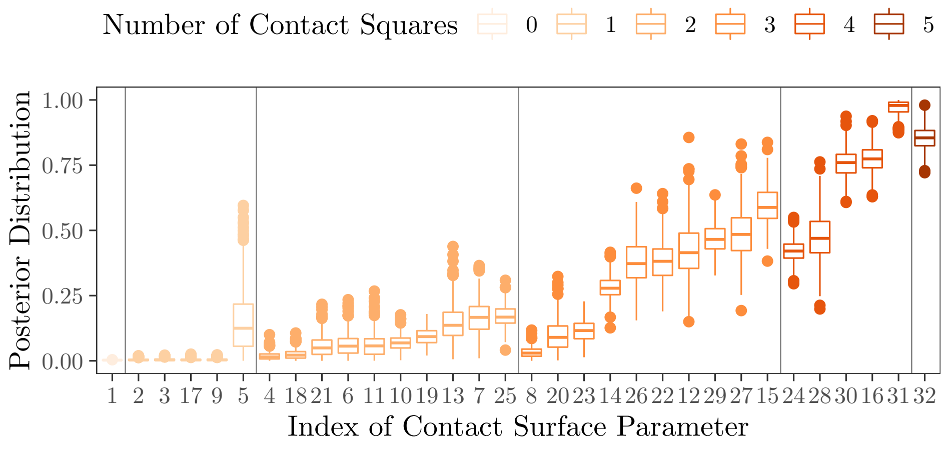


Figure 2.8: Posterior distribution boxplots of the parameters of the 32 possible shapes (listed in Figure 2.5). Boxplot color indicates with the amount of contact surface present in each, with vertical lines partitioning the levels.

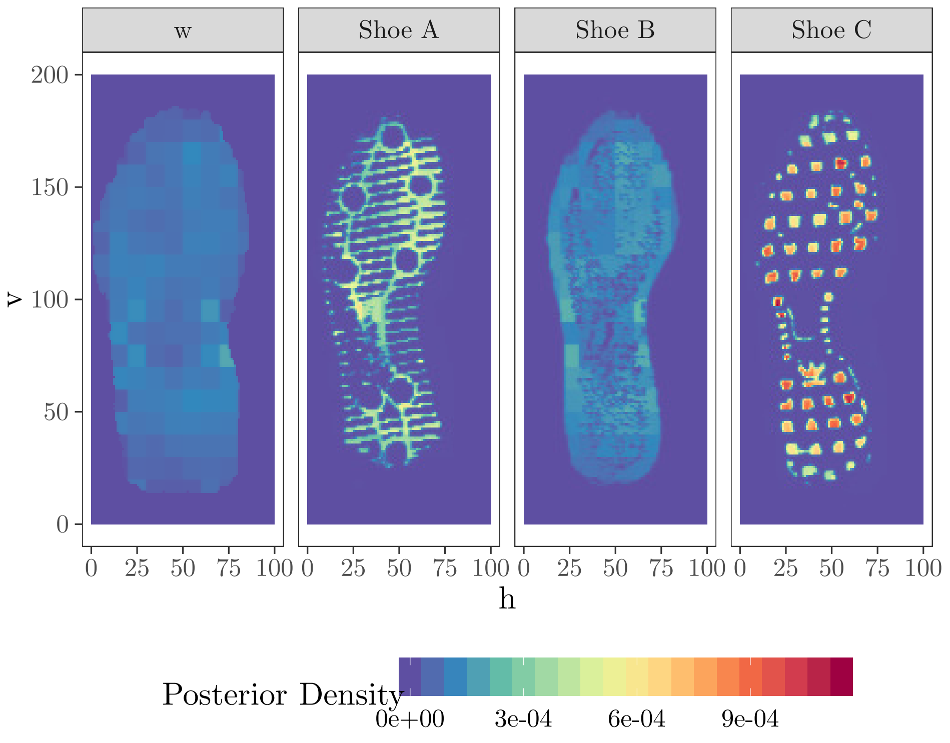
also plays a role in a region's likelihood of being marked with an accidental. However, we caution against over-interpreting such differences due to ϕ being just one component of the larger model.

Figure 2.9 illustrates the posterior predictive distribution of an accidental for four separate contact surfaces. The first panel is synthetic, consisting entirely of contact surface to demonstrate w. The inward facing side of the toe tends to exhibit more accidentals than the in outward facing portion, and the front of the heel tends to exhibit more accidentals than the rear of the heel. A depiction of the fit and uncertainty of the raw w parameter is available as Figure 2.10b.

The second through fourth panels of Figure 2.9 (Shoe A, Shoe B, Shoe C) demonstrate the posterior mean of Λ^s for three example contact surfaces in JESA. The difference in the magnitude of the density between Shoe B and Shoe C demonstrates that the density associated with a particular location is heavily contingent on the total amount of contact surface present for the shoe; because shoe C demonstrates relatively little contact surface, the density is much higher in locations where contact surface is present.

2.6 Discussion

In this work, we made progress on a problem put forth by the President's Council of Advisors in Science and Technology (PCAST, 2016). Namely, we formalized the problem of modeling accidental distributions for random match probabilities, developed a modeling framework for the spatial distribution of accidentals on shoe soles, fit our hierarchical Bayesian model to real data within a Bayesian nonparametric setting to pool information across a variety of shoes, and demonstrated that our model vastly outperforms existing models in the literature on a held-out data task.



(a)

Figure 2.9: Panels w, Shoe A, Shoe B, and Shoe C demonstrate the posterior predictive distribution of accidental locations for four contact surfaces. Panel w is synthetic (entirely contact surface). Shoe A corresponds to the shoe shown in Figure 1. Shoes B and C are other contact surfaces from JESA.

A key takeaway from this endeavor was the importance of explicitly incorporating the contact surface when modeling accidental distributions. We were the first to do so, and it resulted in a major improvement over the traditional models. We took care to develop our model hierarchically, allowing for the pooling information across shoes of different types to capture commonalities in how the contact surface influences accidental distributions. As data sources grow and new data collection efforts are undertaken (CSAFE, 2019), we anticipate the opportunity for more sophisticated models to better capture the relationship between contact surface and accidentals.

Along these lines, a natural extension of our model would be to allow the w, ϕ, and q parameters to differ across shoes according to a nonparametric mixture model. Another possibility would be to extend the model to a spatiotemporal setting, using the temporal data being collected by CSAFE (2019) to model how accidentals accumulate over time.

A possible limitation of our model stems from treating the contact surface parameter ϕ and spatial location parameter w separately. It is plausible that a shoe's intensity would involve dependence between the contact surface and the spatial location. For instance, accidentals could be more likely to occur in high contact areas when on the toe, but more likely to occur in low contact areas when on the heel. In such instances, a model including an interaction effect would outperform our current model.

Another issue we briefly touched on without addressing was the open problem of formally defining when two impressions "match " ($x^s \equiv x^y$). Given a similarity metric defining when $x^s \equiv x^y$, our model is tailored to computing the RMP. Draws from the posterior distribution of $x^s|\mathcal{C}^s$ can serve as a surrogate for sampling from $\mathbb{A}_{\mathcal{C}^y}$ in (2.6), providing a straightforward Monte Carlo strategy for evaluating the RMP. It is worth noting that although our exposition focused on RMPs, our approach is equally applicable to calculating other related summaries of uncertainty, such as likelihood ratios or Bayes factors (Evett et al., 1998).

Finally, we would like to highlight uses of our model outside of direct evaluation of random match probabilities. Recently, the National Institute for Standards in Technology has started development of a multipurpose software tool for forensic footwear examiners (Herman, 2016). One of the tools in development is ShoeGuli, a program for developing synthetic footwear impressions complete with accidentals. As our framework results in an accurate generative model, it is a natural choice for simulating accidental patterns.

2.7 Additional Details

2.7.1 Discretization and Kernel Choice

Recall from §2.2.2 that the contact surface variables are defined on a discrete 100 by 200 equally-spaced grid over $[0, 100] \times [0, 200]$. For practicality, we restrict our model for Λ_s to match the resolution of this grid, discretizing our kernel k to be piece-wise constant over each gridpoint. Theoretically, such a choice restricts the model's flexibility at resolutions smaller than that of the grid. However, we do not expect these resolutions to be relevant to RMP calculations — any such effect will be dominated by the noise in the observed accidental locations for crime scene prints. Attempting to model any structure at such a resolution would amount to overfitting.

In addition to preventing overfitting and facilitating interpretation, the discretized kernel provides computational and modeling advantages.

Computationally, the discretization eliminates the need to keeping track of each real valued accidental locations $x_n^s \in [0, 100] \times [0, 200]$. Instead, we need only store the discrete gridpoint values in $\mathcal{A} = \{1, \ldots, 100\} \times \{1, \ldots, 200\}$. Similarly, we can directly store a Λ_s as a vector of real values and sampling accidental locations from it is equivalent to simply drawing from a multinomial. The grid $\mathcal{A}$ also provides a natural resolution with which to visualize Λ_s. In addition to these computational conveniences, discretization also provides computational speed-ups of the Bayesian inference procedure.

The speed-up is best illustrated by comparing to the standard unimodal symmetric kernel choice in Bayesian nonparametrics — the Gaussian density. If we were to replace the k in the model with a bivariate

Gaussian density, every kernel function would have positive density over the entire $[0, 100] \times [0, 200]$ spatial domain. Consequently, each auxiliary variable Z_n^s could take on any of $|\mathcal{A}|$ values instead of the 49 associated with our discrete kernel. Exploring the large space of possible Z_n^s would lead to additional computational burden for both our MCMC and importance sampling algorithms. Eventually we would need to evaluate $|\mathcal{A}|$ Gaussian densities for each data point.

Moreover, the Gaussian densities would need to be truncated at values outside the $[0, 100] \times [0, 200]$, requiring the computation of normalization constants for any gridpoints close enough to the boundary. If the goal was to infer the bandwidth of the kernel as part of the MCMC procedure, these normalization constants would need to be repeatedly recalculated for each updated bandwidth.

From a modeling perspective, the discrete kernel model also makes more sense than the Gaussian kernel in the context of our data. For many shoes in JESA (e.g. shown in Figure 2.3c), the contact surface drops off steeply and remains zero for large portions of the space, leaving no opportunity for accidentals in these regions. The discrete kernel can accommodate this behavior; its redistribution of the density is restricted to be very local, preventing it from directing density toward these impossible regions. In contrast, the smooth decay of a Gaussian kernel forces it to assign at least some density from each kernel to the entire space, regardless of the contact surface in that area. To limit the density wastage around these steep drop-offs, inference of the Gaussian kernel would promote a very small bandwidth, thus limiting the amount of possible smoothing. Our discrete kernel is better equipped to deal with such a problem, its flexible parameterization allows it to distribute most of the density over nearby gridpoints.

2.7.2 Additional Related Work

The NCoRM framework represents one of many models for collections dependent probability distributions that use normalized random measures. The prototypical normalized completely random measure is the Dirichlet process (Ferguson, 1973) which serves as a building block for much of the literature. Within the spatial statistics literature, Dirichlet process mixture models were first applied by Gelfand et al. (2005) in the context of modeling random functions in space. They have also been applied to model intensities for spatial point processes (e.g. Kottas and Sansó (2007); Taddy (2010); Jewell et al. (2015)). Popular approaches for modeling vectors of dependent probability distributions include the dependent Dirichlet process (MacEachern, 2000), the hierarchical Dirichlet process (Teh et al., 2005), and the nested Dirichlet process (Rodriguez et al., 2008). Non-Dirichlet process-based techniques include (Chen et al., 2013; Foti and Williamson, 2012; Lijoi et al., 2014).

Much of the literature pertaining to vectors of probability measures assumes that the vectors are exchangeable, with the dependent Dirichlet process (MacEachern, 2000) and the kernel stick-breaking process (Dunson and Park, 2008) comprising two notable exceptions. Other recent work pertaining to the modeling vectors of non-exchangeable probability distributions was surveyed in Foti and Williamson (2015). However, we found that the existing literature lacked the tools to incorporate our desired dependence structure for the shoes in JESA, which prompted us to extend the NCoRM framework.

Contrasting with completely random measure-based techniques, another frequently used tool for modeling spatial point processes is the log-Gaussian Cox process (Møller et al., 1998; Adams et al., 2009). The log-Gaussian Cox process is able to capture more sophisticated spatial dependencies by explicitly modeling the log intensity as a draw from a Gaussian process, with the kernel of this process prescribing the spatial correlation structure. We draw on this work by using a log-Gaussian prior on w_E (a finite resolution log-Gaussian process).

2.7.3 Details of Parameterization of w

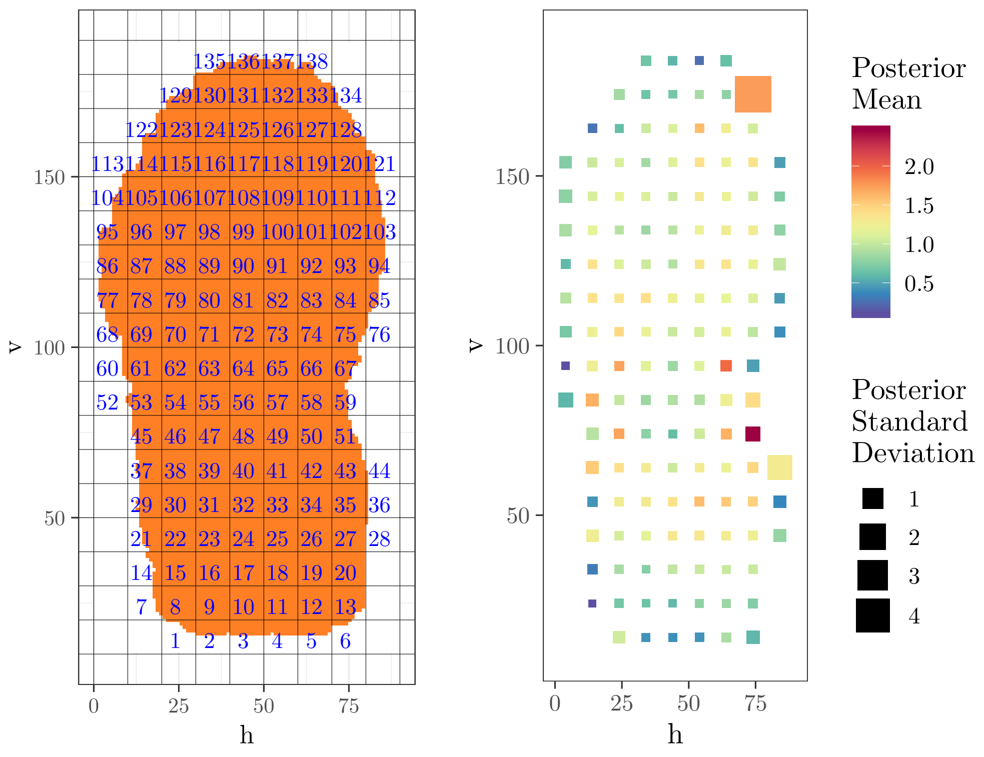
Because inferring 20000 unique entries w represents a large computational burden, it is helpful to reduce its dimension by parametrizing it as piece-wise constant over a coarser region. We define these regions, illustrated in Figure 2.10a, using two criteria. First, we reduce of the resolution from the original 200×100 grid to a 20×10 grid of unique values, with each new region now corresponding to 100 of the original grid points. Second, it is evident from Figures 2.3d and 2.4b that a sizable proportion of $\mathcal{A}$ — specifically the gridpoints at the sides and extremities of the bounding box — have no practical probability of being marked by an accidental. We choose to force their respective w_a's to be 0 in the prior, essentially omitting them from analysis.

After this restriction, we use the remaining grid regions that have at least one positive atom to define our 138 distinct regions. We use $w_E \in \mathbb{R}_+^{138}$ to denote the vector of unique values assigned to each of these regions, assigning it a lognormal prior. The prior mean for $\log(w_E)$ is fixed at 0. The precision Σ is fixed such that each diagonal entry is 1. Off-diagonal entires are 0 for non-adjacent regions, 0.2 otherwise. The full mapping between the entries in $\mathcal{A}$ and the indices of w_E is displayed in Figure 2.10a with the nonzero gridpoints depicted as orange pixels. Throughout the chapter, $\mathcal{A}$ refers to the subset of $\{1, \ldots, 100\} \times \{1, \ldots, 200\}$ that correspond to the nonzero gridpoints.

2.7.4 Details of Marginalization of ϵ^s

Recall that $\Gamma(\cdot)$ denotes the gamma function. Let $\Delta_n^s \in \{-3, 3\}^2$ be shorthand for $Z_n^s - x_n^s$, and let

$$\zeta^s = \left\{Z^s : \Delta_n^s \in \{-3, 3\}^2, n \in \{1, \ldots, N_s\}\right\} \tag{2.31}$$



(a) **(b)**

Figure 2.10: (a) displays the 20000 gridpoints $a \in \mathcal{A}$, colored white if we fix $w_a = 0$, orange otherwise. The black lines partition $\mathcal{A}$ into the coarser 10×10 grid associated with w_E, each nonzero grid region contains a blue number indicating the index in w_E to which it corresponds. (b) summarized the posterior distribution of w_E as a square corresponding to each entry in w_E. As per the legend, the color of the square indicates its posterior mean and the size of the square indicates is posterior standard deviation.

denote the set of possible values for Z^s. After introducing Z^s, the marginal density can be re-expressed as

$$p(x^s|\mathcal{C}^s;\Theta) = \sum_{Z^s \in \zeta^s} \left(\int \prod_{n=1}^{N_s} k(\Delta_n^s) \frac{w_{Z_n^s} \epsilon_{Z_n^s}^s \phi_{Z_n^s}^s}{\sum_{a \in \mathcal{A}} w_a \epsilon_a^s \phi_a^s} \mathrm{d}\rho(\epsilon^s) \right) \tag{2.32}$$

$$= \sum_{Z^s \in \zeta^s} \left(\prod_{n=1}^{N_s} k(\Delta_n^s) \int \frac{\prod_{a \in \mathcal{A}} (w_a \epsilon_a^s \phi_a^s)^{C_a^s}}{\left(\sum_{a \in \mathcal{A}} w_a \epsilon_a^s \phi_a^s\right)^{N_s}} \mathrm{d}\rho(\epsilon^s) \right). \tag{2.33}$$

Let $\mathrm{Ga}(\cdot|\alpha,\beta)$ denote the probability density function of a Gamma distribution with shape α and rate β. Recall that

$$u^s \sim \mathrm{Gamma}\left(n_s, \sum_{a \in \mathcal{A}} w_a \epsilon_a^s \phi_a^s \right). \tag{2.34}$$

Incorporating the density of u^s into (2.32) allows to analytically marginalize the $\epsilon_a^s \sim \text{Gamma}(q,1)$ to derive a simpler expression for $\text{rmp}_V(x^s|\mathcal{C}^s;\Theta)$.

$$\begin{aligned}
&\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\frac{\prod_{a\in\mathcal{A}}(w_a\epsilon_a^s\phi_a^s)^{C_a^s}}{\left(\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)^{N_s}}\mathrm{d}\rho(\epsilon^s)\right)\\
&=\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\frac{\prod_{a\in\mathcal{A}}(w_a\epsilon_a^s\phi_a^s)^{C_a^s}}{\left(\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)^{N_s}}\int\text{Ga}\left(u^s\mid n_s,\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)\mathrm{d}u^s\mathrm{d}\rho(\epsilon^s)\right)\\
&=\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\int\text{Ga}\left(u^s\mid n_s,\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)\frac{\prod_{a\in\mathcal{A}}\text{Ga}\left(\epsilon_a^s|q,1\right)(w_a\epsilon_a^s\phi_a^s)^{C_a^s}}{\left(\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)^{N_s}}\mathrm{d}\epsilon^s\mathrm{d}u^s\right)\\
&=\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\int\frac{\prod_{a\in\mathcal{A}}(w_a\epsilon_a^s\phi_a^s)^{C_a^s}}{\left(\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)^{N_s}}\text{Ga}\left(u^s\mid n_s,\sum_{a\in\mathcal{A}}w_a\epsilon_a^s\phi_a^s\right)\prod_{a\in\mathcal{A}}\text{Ga}\left(\epsilon_a^s|q,1\right)\mathrm{d}\epsilon^s\mathrm{d}u^s\right)\\
&=\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\int\prod_{a\in\mathcal{A}}\frac{(w_a\epsilon_a^s\phi_a^s)^{C_a^s}}{\Gamma(q)(\epsilon_a^s)^{1-q}}\frac{(u^s)^{N_s-1}}{\Gamma(N_s)}\exp\left(-\sum_{a\in\mathcal{A}}(w_a\phi_a^s u_s+1)\epsilon_a^s\right)\mathrm{d}\epsilon^s\mathrm{d}u^s\right)\\
&=\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\frac{(u^s)^{N_s-1}}{\Gamma(N_s)}\prod_{a\in\mathcal{A}}\frac{(w_a\phi_a^s)^{C_a^s}}{\Gamma(q)}\left(\int_0^\infty(\epsilon_a^s)^{C_a^s+q-1}\exp\left(-(w_a\phi_a^s u_s+1)\epsilon_a^s\right)\mathrm{d}\epsilon_a^s\right)\mathrm{d}u^s\right)\\
&=\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\int\frac{(u^s)^{N_s-1}}{\Gamma(N_s)}\prod_{a\in\mathcal{A}}\frac{(w_a\phi_a^s)^{C_a^s}}{\Gamma(q)}\frac{\Gamma(N_s+q)}{(w_a\phi_a^s u^s+1)^{q+C_a^s}}\mathrm{d}u^s\right)\\
&=\frac{1}{\Gamma(q)^{|\mathcal{A}|}}\int_0^\infty\frac{u^{N_s-1}}{\Gamma(N_s)}\sum_{Z^s\in\zeta^s}\left(\prod_{n=1}^{N_s}k(\Delta_n^s)\prod_{a\in\mathcal{A}}\frac{\Gamma(q+C_a^s)\,(w_a\phi_a^s)^{C_a}}{(u^s w_a\phi_a^s+1)^{q+C_a^s}}\right)\mathrm{d}u^s\\
&=\frac{1}{\Gamma(q)^{|\mathcal{A}|}}\int_0^\infty\frac{u^{N_s-1}}{\Gamma(N_s)}\mathbb{E}_{Z^s}\left(\prod_{a\in\mathcal{A}}\frac{\Gamma(q+C_a^s)\,(w_a\phi_a^s)^{C_a}}{(u^s w_a\phi_a^s+1)^{q+C_a^s}}\right)\mathrm{d}u^s
\end{aligned}$$

where $\mathbb{E}_{Z^s}$ denotes the expectation with respect to the distribution of Z^s given by (2.19).

2.7.5 Details of MCMC Proposal Steps

MCMC update for Z_n^s

Let Z_{-n}^s denote $(Z_i^s)_{i\neq n}$ and $C_{a,-n}^s$ denote $C_a^s - I(Z_n^s = a)$. We update each Z_n^s using a Gibbs step, sampling from the conditional distribution of $Z_n^s \mid x^s, \mathcal{C}^s, Z_{-n}^s, u^s, \phi, w, q, k$ given by

$$\mathbb{P}(Z_n^s = z) \propto (q + C_{z,-n}^s)\frac{w_z\phi_z^s}{u^s w_z\phi_z^s + 1}k(x_n^s - z). \tag{2.35}$$

MCMC update for q

Let $\mathcal{D}_i = \sum_{s\in\mathbb{J}}\sum_{a\in\mathcal{A}} I(C_a^s = i)$ for non-negative integers i, and let $\mathcal{D} = (\mathcal{D}_i)_{0\leq i\leq B}$ where B is equal to the largest value of i for which $\mathcal{D}_i > 0$. We update q using a slice sampler on the condition distribution of

$q \mid Z, U, \phi, w, (\mathcal{C}^s)_{s\in\mathbb{J}}, \mathcal{D}$ with density proportional to

$$p(q) \propto \frac{q \exp(-2q) \prod_{i=0}^{B} \Gamma(q+i)^{\mathcal{D}_i}}{\left(\prod_{s\in\mathbb{J}} \prod_{a\in\mathcal{A}} (u^s w_a \phi_a^s + 1)\right)^q}. \tag{2.36}$$

We use the stepping out method as described by Neal (2003), with a step width of 0.2. Note that the computation of $\left(\prod_{s\in\mathbb{J}} \prod_{a\in\mathcal{A}} (u^s w_a \phi_a^s + 1)\right)$ can be recycled as the stepping out algorithm runs, and that the equality $\Gamma(q+i+1) = (q+i)\Gamma(q+i)$ can be exploited to speed-up the calculation of $\prod_{i=0}^{B} \Gamma(q+i)^{\mathcal{D}_i}$.

MCMC update for u^s

We update each u^s using a slice sampler for the conditional distribution of $u^s \mid Z^s, \phi, w, q, (\mathcal{C}^s)_{s\in\mathbb{J}}$ with density proportional to

$$p(u) \propto \frac{u^{N_s - 1}}{\prod_{a\in\mathcal{A}} (u w_a \phi_a^s + 1)^{q + C_a^s}}. \tag{2.37}$$

Our slice sampler uses the stepping out method as described by Neal (2003), with a step width given by $20\sqrt{N_s}(|\mathcal{A}|q)^{-1}$.

MCMC update for ϕ_i

Let $\phi_{-i} = (\phi_j)_{j\neq i}$, $A_{\phi_i}^s = \{a \in \mathcal{A} : \phi_a^s = \phi_i\}$, and $A_{\phi_i} = (A_{\phi_i}^s)_{s\in\mathbb{J}}$. We update each ϕ_i $(i = 1, \ldots, 32)$ using a slice sampler for the conditional distribution of $\phi_i \mid \phi_{-i}, Z, w, q, A_{\phi_i}$ with density proportional to

$$p(\phi_i) \propto \phi_i^{\sum_{s\in\mathbb{J}} |A_{\phi_i}^s|} \prod_{s\in\mathbb{J}} \prod_{a\in A_{\phi_i}^s} \frac{1}{(u^s w_a \phi_i + 1)^{q + C_a^s}}. \tag{2.38}$$

Our slice sampler uses the stepping out method as described by Neal (2003), with a step width given by 0.01.

MCMC update for p^h and p^v

Let $p_{-i}^h = (p_j^h)_{j\neq i}$, $p_{-i}^v = (p_j^v)_{j\neq i}$,

$$\Delta_i^h = \sum_{s\in\mathbb{J}} \sum_{n=1}^{N_s} I(|Z_{n,1}^s - x_{n,1}^s| = i), \tag{2.39}$$

$$\Delta_i^v = \sum_{s\in\mathbb{J}} \sum_{n=1}^{N_s} I(|Z_{n,2}^s - x_{n,2}^s| = i). \tag{2.40}$$

We update each p_i^h $(i = 1, \ldots, 4)$ using a slice sampler for the conditional distribution of $p_i^h \mid p_{-i}^h, Z, x$ with density proportional to

$$p(p_i^h) \propto \exp\left(\frac{-(p_i^h)^2}{8}\right) \frac{\prod_{\ell=i}^4 \left(\sum_{j=\ell}^4 \exp(p_j^h)/(2j-1)\right)^{\Delta_\ell^h}}{\left(\sum_{j=1}^4 \exp(p_j^h)\right)^{\sum_{s\in\mathbb{J}} N_s}}. \tag{2.41}$$

We update each p_i^v $(i = 1, \ldots, 4)$ using a slice sampler for the conditional distribution of $p_i^v \mid p_{-i}^v, Z, x$ with density proportional to

$$p(p_i^v) \propto \exp\left(\frac{-(p_i^v)^2}{8}\right) \frac{\prod_{\ell=i}^4 \left(\sum_{j=\ell}^4 \exp(p_j^v)/(2j-1)\right)^{\Delta_\ell^v}}{\left(\sum_{j=1}^4 \exp(p_j^v)\right)^{\sum_{s\in\mathbb{J}} N_s}}. \tag{2.42}$$

Our slice samplers use the stepping out method as described by Neal (2003), with a step width of 1.

MCMC update for w

For w, we depart from slice sampling and instead use elliptical slice sampling (Murray et al., 2010), leveraging the Gaussian prior on $\log(w)$ to sample from the conditional distribution of $w \mid \phi, Z, w, q$ with

Let $\phi_{-i} = (\phi_j)_{j\neq i}$, $A_{\phi_i}^s = \{a \in \mathcal{A} : \phi_a^s = \phi_i\}$, and $A_{\phi_i} = (A_{\phi_i}^s)_{s\in\mathbb{J}}$. We update each ϕ_i $(i = 1, \ldots, 32)$ using a slice sampler for the conditional distribution of $\phi_i \mid \phi_{-i}, Z, w, q, A_{\phi_i}$ with density proportional to

$$p(w) \propto N(\log(w), \Sigma) \prod_{s\in\mathbb{J}} \prod_{a\in\mathcal{A}} \frac{w_a^{C_a^s}}{(u^s w_a \phi_a^s + 1)^{q+C_a^s}}. \tag{2.43}$$

2.7.6 Importance Sampling Strategy

Details of Importance Distribution

The goal of our importance sampling strategy is to approximate the quantity given in (2.24). For convenience, we replicate this expression below, swapping in the index τ instead of s to denote that it is a held-out shoe.

$$p(x^\tau \mid \mathcal{C}^\tau, T) = \mathbb{E}_\Theta \left(p(x^\tau|\mathcal{C}^\tau, \Theta) \mid (x^s, \mathcal{C}^s)_{s\in T}\right).$$

This expression can be viewed as the composition of two integrals, an outer integral over the posterior distribution of Θ and an inner integral over the local auxiliary variables Z^τ and u^τ.

The outer integral is with respect to the posterior density (2.23). Running the MCMC strategy outlined in §2.4.1 with training data T (i.e. $(x^s, \mathcal{C}^s)_{s\in T}$) produces a chain of L draws $(\Theta^\ell)_{\ell=1,\ldots,L}$. We can use this chain to approximate the posterior density given in (2.23), thus approximating the outer integral as

$$p(x^\tau \mid \mathcal{C}^\tau, T) = L^{-1} \sum_{\ell=1}^{L} \left(p(x^\tau | \mathcal{C}^\tau, \Theta) \mid \Theta^\ell . \right).$$

For each draw in the Markov chain, we can then approximate the inner integral $p(x^\tau|\mathcal{C}^\tau, \Theta)$ in the sum by importance sampling. Recall that the inner integral is given by

$$\begin{aligned} p(x^\tau|\mathcal{C}^\tau, \Theta^\ell) &= \int_0^\infty \frac{u^{N_\tau - 1}}{\Gamma(N_\tau)} \mathbb{E} \left(\frac{1}{\Gamma(q^\ell)^{|\mathcal{A}|}} \prod_{a \in \mathcal{A}} \frac{\Gamma(q^\ell + C_a^\tau) \left(w_a^\ell (\phi^\ell)_a^\tau \right)^{C_a^\tau}}{(u^\tau w_a^\ell (\phi^\ell)_a^\tau + 1)^{q^\ell + C_a^\tau}} \right) \mathrm{d}u^\tau \\ &= \int_0^\infty \sum_{Z^\tau \in \zeta^\tau} \frac{u^{N_\tau - 1}}{\Gamma(N_\tau)} \frac{\prod_{n=1}^{N_\tau} k(\Delta_n^\tau)}{\Gamma(q^\ell)^{|\mathcal{A}|}} \prod_{a \in \mathcal{A}} \frac{\Gamma(q^\ell + C_a^\tau) \left(w_a^\ell (\phi^\ell)_a^\tau \right)^{C_a^\tau}}{(u^\tau w_a^\ell (\phi^\ell)_a^\tau + 1)^{q^\ell + C_a^\tau}} \mathrm{d}u^\tau \\ &= \int_0^\infty \sum_{Z^\tau \in \zeta^\tau} r(u^\tau, Z^\tau) \mathrm{d}u^\tau . \end{aligned}$$

Here, C_a^τ denotes the number of times each $a \in \mathcal{A}$ occurs in Z^τ, $\Delta_n^\tau \in \{-3, 3\}^2$ is shorthand for $Z_n^\tau - x_n^\tau$, and

$$\zeta^\tau = \left\{ Z^\tau : \Delta_n^\tau \in \{-3, 3\}^2, n \in \{1, \ldots, N_\tau\} \right\}. \tag{2.44}$$

The above integral and sum of $r(u^\tau, Z^\tau)$ cannot be evaluated analytically.

Instead, we use importance sampling to evaluate it for each $\ell \in 1, \ldots, L$, treating the integral and sum of $r(u^\tau, Z^\tau)$ as an expectation of a function of u^τ, Z^τ. Implicitly, $r(u^\tau, Z^\tau)$ acts as the product of density and a function of which we are taking the expectation. In practice, there is no need to make a distinction between what serves as the density and what serves as the function — the product is our target. By drawing values of Z^τ and u^τ from an easy-to-sample-from importance distribution and applying the correct importance weights to our draws, we can target this expectation using a Monte Carlo strategy.

In deriving a good importance distribution, we target three properties: cheaply generated random variates, tractable importance weights, and — most importantly — an importance distribution that serves as a good surrogate for the target expectation. Such a surrogate distributes its density in similar places as target integrand to achieve a low variance estimator. Motivated by these properties, we now derive the importance distributions for u^τ and Z^τ.

Before our marginalization of the ϵ terms, the distribution of the auxiliary variable u^τ was independent of Z^τ, given by

$$u^\tau \sim \text{Gamma} \left(N_\tau, \sum_{a \in \mathcal{A}} w_a^\ell (\epsilon^\ell)_a^\tau (\phi^\ell)_a^\tau \right). \tag{2.45}$$

Marginalizing ϵ introduced additional dependence the two, making their joint distribution unwieldy. For our importance distribution, we opt for independence by replacing each of the $(\epsilon^\ell)^\tau_a$ terms in the original gamma distribution with their expected value $\mathbb{E}((\epsilon^\ell)^\tau_a | q^\ell) = q^\ell$. This leads to the importance distribution

$$u^\tau \sim \text{Gamma}\left(N_s, q^\ell \sum_{a\in\mathcal{A}} w^\ell_a (\phi^\ell)^\tau_a\right), \tag{2.46}$$

as given in (2.25) within the main text.

For each Z^τ_n, we could use the distribution given by (2.19). However, we can do better. Noting that a factor $\left(w^\ell_a(\phi^\ell)^\tau_a\right)$ occurs in (2.21) for each $Z^\tau_n = a$, we incorporate these terms in our distribution as well, letting

$$\mathbb{P}(Z^\tau_n = x^\tau_n + a) = \frac{w_{a+x^\tau_n}\phi^\tau_{a+x^\tau_n} k(a)}{\sum_{b\in B} w_{b+x^\tau_n}\phi^\tau_{b+x^\tau_n} k(b)}$$

serve as our importance distribution. This approach is especially helpful when accidentals occur in areas with sparse contact surface. The ϕ information in prevents the distribution from overdrawing unlikely gridpoints surrounded by little contact surface.

The resultant full importance distribution has a hybrid density/mass function given by

$$h(u^\tau, Z^\tau) = \frac{(u^\tau)^{N_s-1}}{\Gamma(N_s)} \frac{\exp\left(-u^\tau q \sum_{a\in\mathcal{A}} w^\ell_a(\phi^\ell)^\tau_a\right)}{\left(q\sum_{a\in\mathcal{A}} w^\ell_a(\phi^\ell)^\tau_a\right)^{N_\tau}} \prod_{n=1}^{N_s} \frac{w^\tau_{Z^\tau_n}\phi^\tau_{Z^\tau_n} k(\Delta^\tau_n)}{\sum_{b\in\zeta^\tau_n} w_b\phi^\tau_b k(b - x^\tau_n)}.$$

To derive the contribution of each generated sample, we divide the integrand $r(u^\tau, Z^\tau)$ by the importance density $h(u^\tau, Z^\tau)$. The result is

$$\begin{aligned} w(u^\tau, Z^\tau) &= \frac{r(u^\tau, Z^\tau)}{h(u^\tau, Z^\tau)} & (2.47)\\ &= \frac{\prod_{n=1}^{N_\tau}\left(\sum_{b\in\zeta^\tau_n} w^{ell}_b(\phi^\ell)^\tau_b k^\ell(b - x^\ell_n)\right)}{\Gamma(q^\ell)^{|\mathcal{A}|}\left(q^\ell\sum_{a\in\mathcal{A}} w^\ell_a(\phi^\ell)^\tau_a\right)^{N_\tau}} \frac{\exp\left(u^\ell q^\ell \sum_{a\in\mathcal{A}} w^\ell_a(\phi^\ell)^\tau_a\right)}{\prod_{a\in\mathcal{A}} \frac{(u^\ell w^\ell_a(\phi^\ell)^\tau_a+1)^{q^\ell+C^\ell_a}}{\Gamma(C^\ell_a+q^\ell)}}. & (2.48)\end{aligned}$$

Thus, using one importance sample ($M = 1$) for each MCMC draw $\Theta^\ell = (\phi^\ell, w^\ell, q^\ell, (p^h)^\ell, (p^v)^\ell)$ yields the approximation

$$p(x^\tau \mid \mathcal{C}^\tau, T) \approx \sum_{\ell=1}^{L} \frac{\prod_{n=1}^{N_\tau}\left(\sum_{b\in B} w_{b+x^\tau_n}(\phi^\ell)^\tau_{b+x^\tau_n} k^\ell(b)\right)}{L\Gamma(q^\ell)^{|\mathcal{A}|}\left(q^\ell\sum_{a\in\mathcal{A}} w_a(\phi^\ell)\tau_a\right)^{N_\tau}} \frac{\exp\left(u^\ell q^\ell \sum_{a\in\mathcal{A}} w^\ell_a(\phi^\ell)^\tau_a\right)}{\prod_{a\in\mathcal{A}} \frac{(u^\ell w^\ell_a(\phi^\ell)^\tau_a+1)^{q^\ell+C^\ell_a}}{\Gamma(C^\ell_a+q^\ell)}}. \tag{2.49}$$

where L is the total number of MCMC draws.

lllustration and Discussion of Chain Mixing

The reliability of the Monte Carlo approximation described in Section 2.4 depends on the mixing of the Markov chain. Here, we demonstrate the mixing of the chain for the targeted quantities by highlighting on the trace plots (Figures 2.11 and 2.12) of the held-out probability estimates shown in Figure 2.7 (specifically, those listed for "Our model" for Split 1). Such quantities (the posterior probability of held-out data under our fitted model) are the target of our model.

Recall that the quantities shown in Figure 2.7 (and summarized in Table 2.1) were obtained by the following process. The Markov chain strategy outlined in §2.4.1 was run for 30000 iterations, after which the first 10000 iterations were discarded as warm-up. For each shoe $\tau = 1, \ldots, 50$, the importance sampling estimates

$$\hat{p}(x^\tau \mid \mathcal{C}^\tau, \Theta^\ell) = w(u^\tau, Z^\tau) \tag{2.50}$$

were computed (by (2.48)) using one importance sample for each of 20000 remaining chain iterations ($\ell = 1, \ldots, 20000$). The metric given by (2.29) was then calculated for each $\hat{p}(x^\tau \mid \mathcal{C}^\tau, \Theta^\ell)$. To obtain the summaries shown in Figure 2.7, the mean was taken of each chain. Here, we examine the contents of each of the 50 chains before averaging.

Figures 2.11 and 2.12 demonstrate trace plots for each of the 50 held-out shoes over the 20000 iterations of the Markov chain. The chains are presented in order of mean per-shoe performance — the same sequence they appear on the x-axis in Figure 2.7 (Data Split 1). By inspection, almost every chain mixed very well. The notable exception is the chain for held-out shoe 2, having an effective sample size of 19.3 (all others exceed 35). The slower mixing of this case is not necessarily surprising — this chain appears to be more diffuse than the others, indicating that the probability evaluation is especially uncertain.

Nonetheless, an effective sample size of 19.3 still provides a reasonable estimate of the mean of the chain. If it is essential to have more accurate estimates of less robust functions (such as more extreme quantiles), the chain would have to be run for a longer number of iterations.

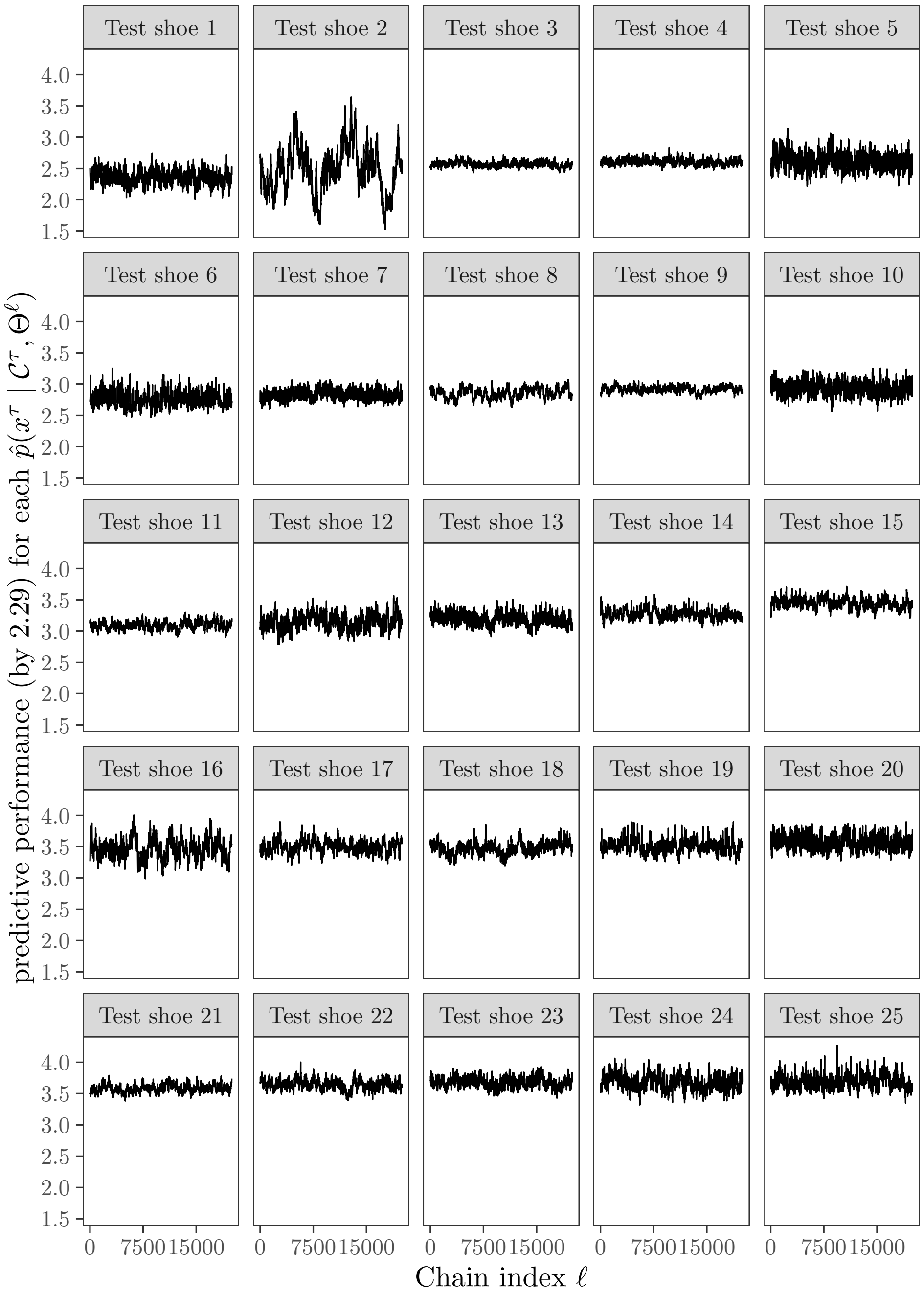


Figure 2.11: Trace plots demonstrating the Markov chain for the estimated held-out predictive performance per accidental corresponding to shoes 1 through 25 of Data Split 1. The shoes are ordered sequentially by the mean predictive performance, and the results pertain to our full model.

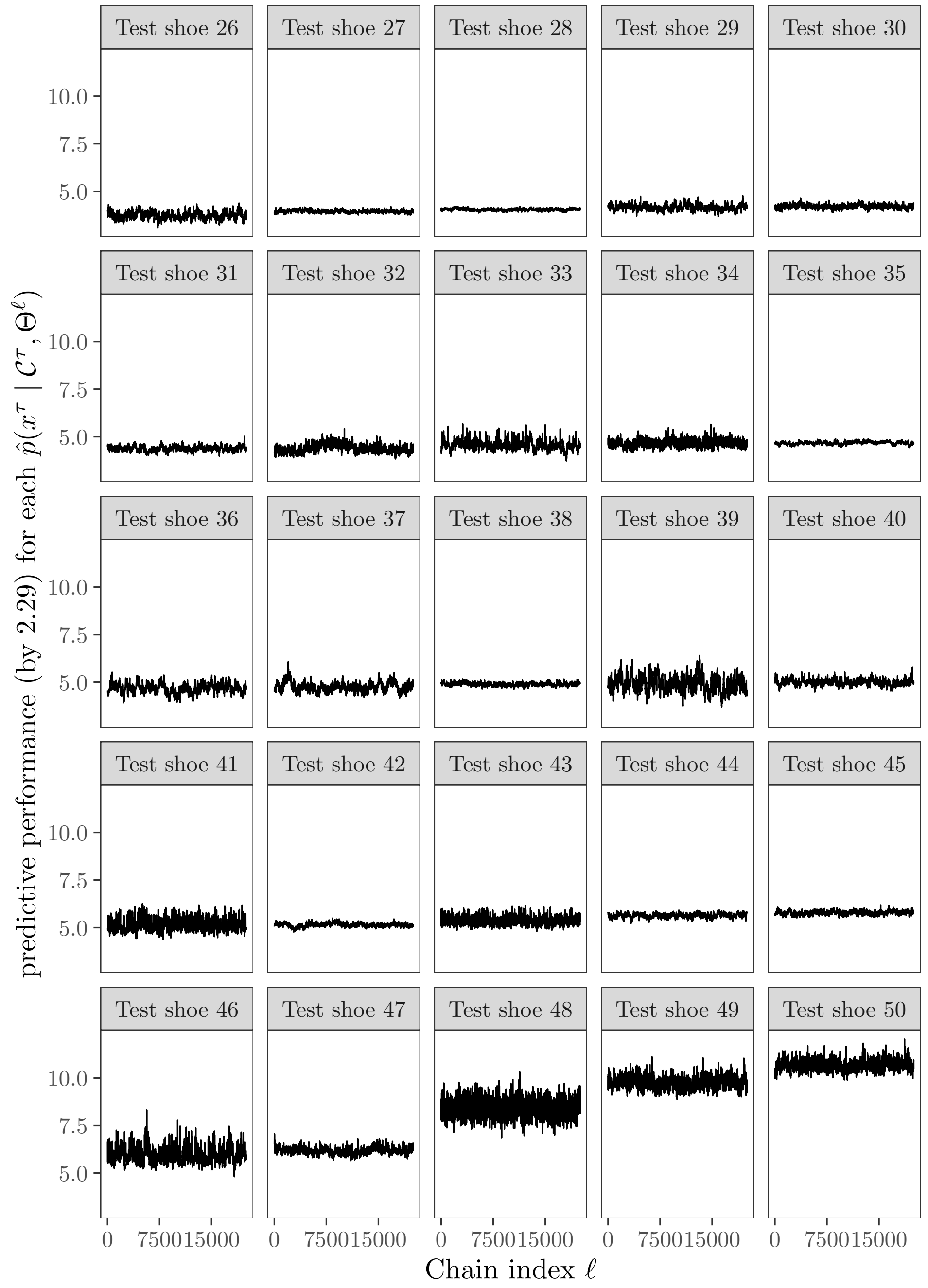


Figure 2.12: A companion to Figure 2.11. Trace plots demonstrating the Markov chain for the estimated held-out predictive performance per accidental corresponding to shoes 26 through 50 of Data Split 1. The shoes are ordered sequentially by the mean predictive performance, and the results pertain to our full model.

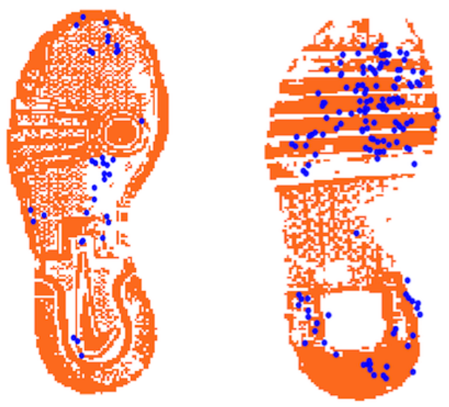

(a) **(b)**

Figure 2.13: The contact surfaces and overlaid accidentals of two example shoes (a) and (b) from the JESA database. In both of these cases, some of the accidentals do not occur on the contact surface

Chapter 3

Projective, Sparse, and Learnable Latent Position Network Models

3.1 Introduction

Network data consist of relational information between entities, such as friendships between people or interactions between cell proteins. Often, these data take the form of binary measurements on dyads, indicating the presence or absence of a relationship between entities. Such network data can be modeled as a stochastic graph, with each individual dyad being a random edge. Stochastic graph models have been an active area of research for over fifty years across physics, sociology, mathematics, statistics, computer science, and other disciplines (Newman, 2003).

Many leading stochastic graph models assume that the inhomogeneity in connection patterns across nodes is explained by node-level latent variables. The most tractable version of this assumption is that the dyads are conditionally independent given the latent variables. In this article, we focus on a subclass of these conditionally independent dyad models—the distance-based latent position network model (LPM) of Hoff et al. (2002).

In LPMs, each node is assumed to have a latent position in a continuous space. The edges follow independent Bernoulli distributions with probabilities given by a decreasing function of the distance between the nodes' latent positions. By the triangle inequality, LPMs exhibit edge transitivity; friends of friends are more likely to be friends. When the latent space is assumed to be $\mathbb{R}^2$ or $\mathbb{R}^3$, the inferred latent positions can provide an embedding with which to visualize and interpret the network.

Recently, there has been an effort to classify stochastic graph models into general unified frameworks. One notable success story has been that of the graphon for exchangeable networks (Diaconis and Janson, 2008).

The graphon characterizes all stochastic graphs invariant under isomorphism as latent variable models. LPMs can be placed within the graphon framework by assuming the latent positions are random effects drawn independently from the same (possibly unknown) probability distribution. However, graphons can be inappropriate for some modeling tasks, due to their asymptotic properties.

The typical asymptotic regime for statistical theory of network models considers the number of nodes growing to infinity in a single graph. Implicitly, this approach requires that the network model define a distribution over a sequence of increasingly sized graphs. There are several natural questions to ask about this sequence. Prominent questions include:

1. At what rate does the number of edges in these graphs grow?

2. Is the model's behavior consistent across networks of different sizes?

3. Can one eventually learn the model's parameters as the graph grows?

For all non-trivial[*] models falling within the graphon framework, the answer to question 1 is identical; the expected number of edges grows quadratically with the number of nodes (Orbanz and Roy, 2015). Such sequences of graphs—in which the average degree grows linearly—are called *dense*. In contrast, many real-world networks are thought to have sub-linear average degree growth. This property is known as *sparsity* (Newman, 2010, Chapter 6.9)).

For sparse graphs, graphon models are unsuitable. Accordingly, recent years have seen an effort to develop sparse graph models that preserve the advantages of graphons. In particular, the sparse graphon framework (Bollobás et al., 2007; Borgs et al., 2014) and the graphex framework (Caron and Fox, 2014; Veitch and Roy, 2015; Borgs et al., 2016) both provide straightforward ways to modify network models from the dense regime to accommodate sparsity.

In this article, we add to the sparse graph literature by formulating a new sparse LPM. We target three criteria: *sparsity* (§3.2.1), *projectivity* (§3.2.2) and *learnablity* (§3.4.1). Projectivity of a model ensures consistency of the distributions it assigns to graphs of different sizes, and learnability ensures consistent estimation of the latent positions as the number of nodes grows.

As we outline in Section 3.5, the existing methods for sparsifying graphons of Borgs et al. (2014) and Veitch and Roy (2015) do not satisfy these criteria; they either violate projectivity or make it difficult to establish learnability. We thus take a more specialized approach to develop our sparse LPMs, turning to non-exchangeable network models for inspiration. Specifically, our new LPM framework extends the Poisson random connection model (Meester and Roy, 1996)—a specialized LPM framework in which the nodes' latent positions are generated according to a Poisson process. We modify the observation window approach

[*]The only exception is an empty graph, for which all edges are absent with probability one.

proposed by Krioukov and Ostilli (2013) to allow our LPMs to exhibit arbitrary levels of sparsity without sacrificing projectivity.

To obtain learnability results for our LPM framework, we develop and modify a combination of results related to low rank matrix estimation (Davenport et al., 2014), the Davis-Kahan Theorem (Yu et al., 2014), and eigenvalue concentration in random Euclidean distance matrices. The strategy culminates in a concentration inequality for a restricted maximum likelihood estimator of the latent positions that applies to wide a variety of LPMs, providing a straightforward sufficient conditions for LPM learnability.

The remainder of this article is organized as follows. Section 3.2 defines sparsity (§3.2.1) and projectivity (§3.2.2) for graph sequences. It also defines the LPM, establishing sparsity and projectivity results for its exchangeable (§3.2.4) and random connection model (§3.2.5) formulations. Section 3.3 describes our new framework for modeling projective sparse LPMs, and includes results that demonstrate that the resultant graph sequences are projective and sparse. Section 3.4 defines learnability of latent position models, and provides conditions under which sparse latent position models are learnable. Finally, Section 3.5 elaborates on the connections between our approach, sparse graphon-based LPMs, and the graphex framework. It also includes a discussion of the limitations of our work. All proofs are deferred to Section 3.6.

3.2 Background

3.2.1 Sparsity

Let $(Y^n)_{n=1,\ldots,\infty}$ be a sequence of increasingly sized $(n \times n)$ random adjacency matrices associated with a sequence of increasingly sized simple undirected random graphs (on n nodes). Here, each entry Y^n_{ij} indicates the presence of an edge between nodes i and j for a graph on n nodes.

We say the sequence of stochastic graph models defined by $(Y^n)_{n=1,\ldots,\infty}$ is *sparse in expectation* if

$$\lim_{n\to\infty} \mathbb{E}\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} Y^n_{ij}}{n^2}\right) = 0. \tag{3.1}$$

In other words, a sequence of graphs is sparse in expectation if the expected number of edges scales sub-quadratically in the number of nodes.

Recall that a node's degree is defined as the number of nodes to which it is adjacent. Sparsity in expectation is equivalent to the expected average node degree growing sub-linearly. If instead the average degree grows linearly, we say the graph is *dense* in expectation.

In this article, we are also interested in distinguishing between degrees of sparsity. We say that a graph is $e(n)$-*sparse in expectation* if

$$\lim_{n\to\infty} \mathbb{E}\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} Y_{ij}}{e(n)}\right) = C \tag{3.2}$$

for some constant $C \in \mathbb{R}_+$. That is, the number of edges scales $\Theta(e(n))$. A dense graph could also be called n^2-sparse in expectation.

Note that sparsity and $e(n)$-sparsity are asymptotic properties of graphs, defined for increasing sequences of graphs but not for finite realizations. These definitions differ from the informal use of "sparse graph" to refer to a single graph with few edges. It also differs from the definition of sparsity for weighted graphs used in Rastelli (2018). In practice, we typically observe a single finite realization of a graph, but the notion of sparsity remains useful because many network models naturally define a sequence of networks.

3.2.2 Projectivity

Let $(\mathbb{P}^n)_{n=1\ldots\infty}$ denote the probability distributions corresponding to a growing sequence of random adjacency matrices $(Y^n)_{n=1,\ldots,\infty}$ for a sequence of graphs. We say that the sequence $(\mathbb{P}^n)_{n=1\ldots\infty}$ is *projective* if, for any $n_1 < n_2$, the distribution over adjacency matrices induced by $\mathbb{P}^{n_1}$ is equivalent to the distribution over $n_1 \times n_1$ sub-matrices induced by the leading n_1 rows and columns of an adjacency matrix following $\mathbb{P}^{n_2}$. That is, $(\mathbb{P}^n)_{n=1,\ldots,\infty}$ is projective if for any $y \in \{0,1\}^{n_1\times n_1}$,

$$\mathbb{P}^{n_1}(Y^{n_1} = y) = \mathbb{P}^{n_2}(Y^{n_2} \in X), \tag{3.3}$$

where $X = \Big\{x \in \{0,1\}^{n_2\times n_2} : x_{ij} = y_{ij} \text{ if } 1 \le i,j \le n_1\Big\}$.

Projectivity ensures a notion of consistency between networks of different sizes, provided that they are generated from the same model class. This property is particularly useful for problems of superpopulation inference (D'Amour and Airoldi, 2016), such as testing whether separate networks were drawn from the same population, predicting the values of dyads associated with a new node, or shrinking together estimates from separate networks in a hierarchical model. Such problems require that parameter inferences be comparable across differently sized graphs. Without projectivity, it is unclear how to make comparisons without additional assumptions.

Projectivity has thus received considerable attention recently in the networks literature (Snijders, 2010; Shalizi and Rinaldo, 2013; Crane and Dempsey, 2016; Schweinberger et al., 2017; Kartun-Giles et al., 2018). Our definition of projectivity departs from others in the literature in that it depends on a specific ordering of the nodes. Other definitions require consistency under subsampling of any n_1 nodes, not just the first n_1 nodes. The two definitions coincide when exchangeability is assumed, but differ otherwise.

3.2.3 Latent Position Network Models

The notion that entities in networks possess latent positions has a long history in the social science literature. The idea of a "social space" that influences the social interactions of individuals traces back to at least the seventeenth century (Sorokin, 1927, p. 3). A thorough history of the notions of social space and social distance as they pertain to social networks is provided in McFarland and Brown (1973).

In the statistical network modeling literature, assigning continuous latent positions to nodes dates back to the 1970s, in which multi-dimensional scaling was used to summarize similarities between nodes in the data (Wasserman and Faust, 1994, p. 385). However, it was not until Hoff et al. (2002) that the modern notion of latent continuous positions were used to define a probabilistic model for stochastic graphs in the statistics literature. In this article, we focus on this probabilistic formulation, with our definition of latent position models (LPMs) following that of the distance model of Hoff et al. (2002).

Consider a binary graph on n nodes. The LPM is characterized by each node i of the network possessing a latent position Z_i in a metric space (S, ρ). Conditional on these latent positions, the edges are drawn as independent Bernoulli random variables following

$$\mathbb{P}(Y_{ij} = 1 | Z_i, Z_j) = K(\rho(Z_i, Z_j)). \tag{3.4}$$

Here, $K : \mathbb{R}_+ \to [0, 1]$ is known as the link probability function; it captures the dependency of edge probabilities on the latent inter-node distances. For the majority of this article, we assume K is independent of n (§3.5.1 is an exception). Furthermore, we focus on link probability functions that smoothly decrease with distance and are integrable on the real line, such as $\text{expit}(-\rho^2)$, $\exp(-\rho^2)$ and $(1 + \rho^2)^{-1}$. Though the general formulation of the LPM in Hoff et al. (2002) allows for dyad-specific covariates to influence connectivity, our exposition assumes that no such covariates are available. We have done this for purposes of clarity; our framework does not specifically exclude them.

3.2.4 Exchangeable Latent Position Network Models

Originally, Hoff et al. (2002) proposed modeling the nodes' latent positions as independent and identically distributed random effects drawn from a distribution f of known parametric form. This approach remains popular in practice today, with S assumed to be a low-dimensional Euclidean space $\mathbb{R}^d$ and f typically assumed to be multivariate Gaussian or a mixture of multivariate Gaussians (Handcock et al., 2007). We refer to this class of models as *exchangeable LPMs* because they assume the nodes are infinitely exchangeable. Exchangeable latent position network models are projective, but must be dense in expectation.

Proposition 1. *Exchangeable latent position network models define a projective sequence of models.*

Proof. Provided in §3.6.2. □

Proposition 2. *Exchangeable latent position network models define dense in expectation graph sequences.*

Proof. Provided in §3.6.3. □

Consequently, LPMs with exchangeable latent positions cannot be sparse. To develop sparse LPMs, we must consider alternative assumptions.

3.2.5 Poisson Random Connection Model

Instead of the latent positions being generated independently from a distribution over S, we can treat them as drawn according to a point process over S. This approach—known as the random connection model—has been well-studied in the context of percolation theory (Meester and Roy, 1996). Most of this focus has been on random geometric graph (Penrose, 2003), a version of a LPMs for which K is an indicator function of the distance (i.e. $K(\rho(Z_i, Z_j)) \propto I(\rho(Z_i, Z_j) < \epsilon)$). Here, we instead study the random connection model as a statistical model, focusing the case where K is a smoothly decaying and integrable function.

In particular, we consider the *Poisson* random connection model (Gilbert, 1961; Penrose, 1991), for which the point process is assumed to be a homogeneous Poisson process (Kingman, 1993) over $S \subseteq \mathbb{R}^d$. Because Poisson random connection models on finite-measure S are equivalent to exchangeable LPMs, the interesting cases occur when S has infinite measure, such as $\mathbb{R}^d$. In these cases, the expected number of points is almost-surely infinite, resulting in an infinite number of nodes.

These infinite graphs can be converted into a growing sequence of finite graphs via the following procedure. Let G denote an infinite graph generated according to a Poisson random connection model on S. Let

$$S_1 \subset S_2 \subset \cdots \subset S_n \subset \cdots \subset S \tag{3.5}$$

denote a nested sequence of finitely-sized *observation windows* in S. For each S_i, define G_i to be the subgraph of G induced by keeping only those nodes with latent positions in S_i. Because these positions form a Poisson process, each G_i consists of a Poisson distributed number of nodes with mean given by the size of S_i. Each G_i is thus almost-surely finite, and the sequence of graphs $(G_i)_{i=1,\dots\infty}$ contains a stochastically increasing number of nodes.

For many choices of S, such as $\mathbb{R}^d$, this approach straightforwardly extends to a continuum of graphs by considering a continuum of nested observation windows of $(S_t)_{t \in \mathbb{R}_+}$. In such cases, the number of nodes follows a continuous-time stochastic process, stochastically increasing in t.

As far as we are aware, the above approach was first proposed by Krioukov and Ostilli (2013) in the context of defining a growing sequence of geometric random graphs. Their exposition concentrated on a one-dimensional example with $S = \mathbb{R}_+$ and observation windows given by $S_t = [0, t]$. For this example, one would expect to observe n nodes if $t = n$, with the total number of nodes for a given t being random. As noted by Krioukov and Ostilli (2013), the formulation can altered to ensure that n nodes are observed by treating n as fixed and treating the window size t_n as the random quantity. Here, t_n it equal to the smallest window width such that $[0, t_n]$ contains exactly n points. These two viewpoints (random window size and random number of nodes) are complementary for analyzing the same underlying process.

Under the appropriate conditions, the one-dimensional Poisson random connection model results in networks which are n-sparse in expectation. We formalize this notion as Proposition 3. The finite window approach approach also defines a projective sequence of models, as stated in Proposition 4.

Proposition 3. *For a Poisson random connection model on $\mathbb{R}_+$ with an integrable link probability function, the graph sequence resulting from the finite window approach is n-sparse in expectation.*

Proof. Provided in §3.6.3 $\square$

Proposition 4. *Consider a Poisson random connection model on $\mathbb{R}_+$ with link probability function K. Then, the graph sequence resulting from the finite window approach is projective.*

Proof. Provided in §3.6.2. $\square$

These results indicate that the Poisson random connection model restricted to observation windows is capable of defining a sparse graph sequences, but only for a specific sparsity level if the link probability function is integrable. For our new framework, we extend this observation window approach to higher dimensional S. By including an auxiliary dimension, we achieve all rates between n-sparsity and n^2-sparsity (density) in expectation.

3.3 New Framework

When working in a one-dimensional Euclidean latent space $S = \mathbb{R}_+$, the observation window approach for the Poisson random connection model is straightforward—the width of the window grows linearly with t, with nodes arriving as the window grows. As shown in Proposition 3, this process results in graph sequences which are n-sparse in expectation whenever K is integrable. However, extending to d dimensions ($\mathbb{R}^d$) provides freedom in defining how the window grows; different dimensions of the window can be grown at different rates.

We exploit this extra flexibility to develop our new sparse LPM model. Specifically, through the inclusion of an auxiliary dimension—an additional latent space coordinate which influences when a node becomes visible without influencing its connection probabilities—we can control the level of sparsity of the graph by trading off how quickly we grow the window in the auxiliary dimension versus the others.

In this section, we formalize this auxiliary dimension approach, showing that it allows us to develop a new LPM framework for which the level of sparsity can be controlled while maintaining projectivity. Our exposition consists of two parts: first, we present the framework in the context of a general S. Then, we concentrate on a special subclass with $S = \mathbb{R}^d$ for which it is possible to prove projectivity, sparsity, and establish learnability results. We refer to this special class as rectangular LPMs.

3.3.1 Sparse Latent Position Model

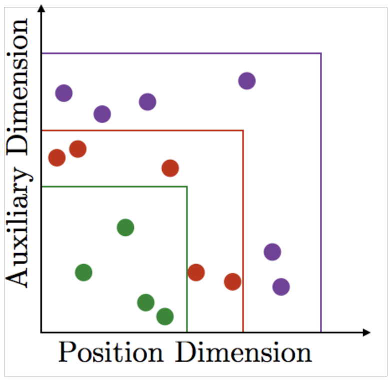
Our new LPM's definition follows closely with that of the Poisson random connection model restricted to finite windows: the positions in the latent space are given by a homogeneous Poisson point process, and the link probability function K is independent of the number of nodes. The main departure from the random connection model is formulating K such that it depends on the inter-node distance in just a subset of the dimensions—specifically all but the auxiliary dimension. The following is a set of ingredients to formulate a sparse LPM.

- *Position Space:* A measurable metric space $(S, \mathcal{S}, \rho)$ equipped with a Lebesgue measure ℓ_1.
- *Auxiliary Dimension:* The measure space $(\mathbb{R}_+, \mathcal{B}, \ell_2)$ where $\mathcal{B}$ is Borel and ℓ_2 is Lebesgue.
- *Product Space:* The product measure space $(S^*, \mathcal{S}^*, \lambda)$ on $(S \times \mathbb{R}_+, \mathcal{S} \times \mathcal{B})$, equipped with $\lambda = \ell_1 \times \ell_2$, the coupling of ℓ_1 and ℓ_2.
- *Continuum of observation windows:* A function $H : \mathbb{R}_+ \to \mathcal{S}^*$ such that $t_1 < t_2 \Rightarrow H(t_1) \subset H(t_2)$ and $|H(t)| = t$.
- *Link probability function*: A function $K : \mathbb{R}_+ \to [0, 1]$.

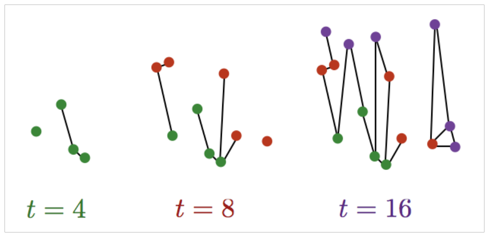
Jointly, we say the triple $((S, \mathcal{S}, \rho), H, K)$ defines a stochastic graph sequence called a sparse LPM. The position space plays the role of the latent space as in traditional LPMs, with the link probability function K controlling the probability of an edge given the corresponding latent distance. The auxiliary dimension plays no role in connection probabilities. Instead, a node's auxiliary coordinate—in conjunction with its latent position and the continuum of observation windows—determines when it appears.

Specifically, a node with position (Z, r) is observable at *time* $t \in \mathbb{R}_+$ if and only if $(Z, r) \in H(t)$. Here, time need not correspond to physical time; it is merely an index for a continuum of graphs as in the case for the Poisson random connection model. We refer to t_i—defined as the smallest $t \in \mathbb{R}_+$ for which $(Z_i, r_i) \in H(t)$—as the arrival time of the ith node where (Z_i, r_i) are the corresponding latent position and auxiliary value for node i.

Considered jointly, the coordinates defined by the latent positions and auxiliary positions assigned to nodes can be viewed as a point process over $S \times \mathbb{R}_+$. As in the Poisson random connection model, we assume this point process is a unit-rate Poisson. The continuum of observation windows $H(t)$ controls the portion



(a) A realization of a point process on the product space. Square observation windows $H(t)$ for $t = 4, 8, 16$ are depicted in green, red, and purple, respectively. The points are coloured according to the first observation window for which they are observable.



(b) Latent position graphs corresponding to the three observation windows depicted in Figure 1(a). The link probability function used is a decreasing function of distance in the position dimension.

Figure 3.1: An example of a point process and observation windows which generate a sequence of sparse latent position graphs

of the point process which is observed at time t. Since the size of H is increasing in t, this model defines a growing sequence of graphs with the number of nodes growing stochastically in t as follows.

- Generate a unit-rate Poisson process Ψ on $(S^*, \mathcal{S}^*)$.
- Each point $(Z, r) \in S \times \mathbb{R}_+$ in the process corresponds to a node with latent position Z and auxiliary coordinate r.
- For a dyad on nodes with latent positions Z_i and Z_j, include an edge with probability $K(\rho(Z_i, Z_j))$.
- At time t the subgraph induced by by restricting Ψ to $H(t)$ is visible.

A graph of size n can be obtained from the above framework by choosing any t_n such that $|\Psi \cap H(t_n)| = n$. Each $t_n < t_{n+1}$ with probability one (by Lemma 3.3.3). Thus, the above generative process is well-defined for any n, and the nodes are well-ordered by their arrival times.

Due to its flexibility, the above framework defines a broad class of LPMs. For instance, the exchangeable LPM can be viewed as a special case of the above framework in which the observation window grows only in the auxiliary dimension. However, the full generality of this framework makes it difficult to establish general sparsity and learnability results. For this reason, we have chosen to focus on a subclass to derive our sparsity, projectivity, and learnability proofs. We refer to this class as rectangular LPMs. We have chosen this class because it allows us to emphasize the key insights in the proofs without having to do too much extra bookkeeping.

3.3.2 Rectangular Latent Position Model

For rectangular LPMs, we impose further criteria on the basic sparse LPM. The latent space is assumed to be Euclidean ($S = \mathbb{R}^d$). The continuum of observation windows $H(t)$ are defined by the nested regions

$$H(t) = [-g(t), g(t)]^d \times \left[0, \frac{t}{(2g(t))^d}\right] \tag{3.6}$$

where $g(t) = t^{p/d}$ for $0 \leq p \leq 1$ controls the rate at which the observation window grows for the latent position coordinates. The growth rate in the auxiliary dimension is chosen to be $2^{-d}t^{1-p}$ to ensure that the volume of $H(t)$ is t. We further assume that

$$\int_0^\infty u^{d-1}K(u)\mathrm{d}u < \infty \tag{3.7}$$

to ensure that the average distance between a node and its neighbors remains bounded as n grows. We now demonstrate the projectivity and sparsity of rectangular LPMs as Theorems 3.1 and 3.2.

Theorem 3.1. *Rectangular sparse latent position network models define a projective sequence of models.*

Proof. Provided in §3.6.2 □

Theorem 3.2. *A d-dimensional rectangular latent space model is n^{2-p}-sparse in expectation, where $g(n) = n^{p/d}$.*

Proof. Provided in §3.6.3 □

By specifying the appropriate value of p for a rectangular LPM, it is thus possible to obtain any polynomial level of sparsity within n-sparse and n^2-sparse (dense) in expectation. Other intermediate rates of sparsity such as $n\log(n)$ can also be obtained considering non-polynomial $g(n)$. We now investigate for which levels of sparsity it is possible to do reliable statistical inference of the latent positions.

3.4 Learnability

3.4.1 Preliminaries

Recall that the edge probabilities in a LPM are controlled by two things: the link probability function K and the latent positions $Z \in S^n$. In this section, we consider the problem of consistently estimating the latent positions for a LPM using the observed adjacency matrix. We focus on the case where both K and $S = \mathbb{R}^d$ are known, relying on assumptions that are compatible with rectangular LPMs.

In the process of establishing our consistent estimation results for Z, we also establish consistency results for two other quantities: the squared latent distance matrix $D^Z \in \mathbb{R}^{n\times n}$ defined by $D^Z_{ij} = ||Z_i - Z_j||^2$ and the link probability matrix $P^Z \in [0,1]^{n\times n}$ defined by $P^Z_{ij} = K((D^Z_{ij})^{1/2})$. These results are also of independent interest because—like Z—the distance matrix and link probability matrix also characterize a LPM when K is known.

We use the following notation and terminology to communicate our results. Let $||\cdot||_F$ denote the Frobenius norm of a matrix, $\xrightarrow{p}$ denote convergence in probability, $\mathcal{O}_d$ denote the space of orthogonal matrices on $\mathbb{R}^{d\times d}$, and $\mathcal{Q}_{nd} \subset \mathbb{R}^{n\times d}$ denote the set of all $n \times d$ matrices with identical rows.

We say that a LPM has *learnable latent positions* if there exists an estimator $\hat{Z}(Y^n)$ such that

$$\lim_{n\to\infty} \inf_{O\in\mathcal{O}_d, Q\in\mathcal{Q}_{nd}} \frac{||\hat{Z}(Y^n)O - Q - Z||_F^2}{n} \xrightarrow{p} 0. \tag{3.8}$$

That is, a LPM has learnable positions if there exists an estimator $\hat{Z}(Y^n)$ of the latent positions such that the average distance between $\hat{Z}(Y^n)$ and the true latent positions converges to 0. The infimum over the transformations induced by $O \in \mathcal{O}_d$ and $Q \in \mathcal{Q}_{nd}$ is included to account for the fact that the likelihood of a LPM is invariant to isometric translations (captured by Q) and rotations/reflections (captured by O) of the latent positions (Shalizi and Asta, 2017).

We say that a LPM has *learnable squared distances* if there exists an estimator $\hat{Z}(Y^n)$ such that

$$\lim_{n\to\infty} \frac{||D^{\hat{Z}(Y^n)} - D^Z||_F^2}{n^2} \xrightarrow{p} 0. \tag{3.9}$$

That is, a LPM has learnable squared distances if the average squared difference between the estimator for the matrix of squared distances induced by $\hat{Z}(Y^n)$ and the true matrix of squared distances D^Z converges to 0. Unlike the latent positions, D^Z is uniquely identified by the likelihood; there is no need to account for rotations, reflections, or translations.

Finally, we say a LPM that is $e(n)$-sparse in expectation has *learnable link probabilities* if there exists an estimator $\hat{Z}(Y^n)$ such that

$$\lim_{n\to\infty} \frac{||P^{\hat{Z}(Y^n)} - P^Z||_F^2}{e(n)} \xrightarrow{p} 0. \tag{3.10}$$

Note that a scaling factor of $e(n)$ is used instead of the more intuitive n^2 to account for the sparsity. Otherwise the link probability matrix for a sparse graph could be trivially estimated because $n^{-2}||P^Z||_F^2 \xrightarrow{p} 0$.

3.4.2 Related Work on Learnability

Before presenting our results, we summarize some of the existing work on learnability of LPMs in the literature. Choi and Wolfe (2011) considered the problem of estimating LPMs from a classical statistical learning theory perspective. They established bounds on the growth function and shattering number for

LPMs with link function given by $K(\delta) = (1+\exp\delta)^{-1}$. However, we have found that their inequalities were not sharp enough to be helpful for proving learnability for sparse LPMs.

Shalizi and Asta (2017) provide regularity conditions under which LPMs have learnable positions on general spaces S, assuming that the link probability function K is known and possesses certain regularity properties. Specifically, they require that the absolute value of the logit of the link probability function is slowly growing, which does not necessarily hold in our setting.

Our learnability results more closely resemble those of Ma and Ma (2017), who consider a latent variable network model of the form $\text{logit}(\mathbb{P}(A_{ij} = 1)) = \alpha_i + \alpha_j + \beta X_{ij} + Z_i^T Z_j$, originally due to Hoff (2005). Here, α_i denote node-specific effects, X_{ij} denote observed dyadic covariates and β denotes a corresponding linear coefficient. If there are no covariates and $\alpha_i = ||Z_i||^2/2$, their approach defines a LPM with $K(\delta) = \text{expit}(-\delta^2)$. Ma and Ma (2017) provide algorithms and regularity conditions for consistent estimation of both the logit-transformed probability matrix and $Z^T Z$ under this model, using results from Davenport et al. (2014). Here, we use similar concentration arguments to establish Lemmas 3.2.1 and 3.2.2, but our results differ in that we consider a more general class of link functions, and also establish learnability of latent positions via an application of the Davis-Kahan theorem.

Our learnability of latent positions result (Lemma 3.2.3) resembles that of Sussman et al. (2014), who establish that the latent positions for dot-product network models can be consistently estimated. The dot product model—a latent variable model which is closely related to the LPM—has a link probability function defined by $K(Z_i, Z_j) = Z_i \cdot Z_j$ with $Z_j, Z_j \in S$. The latent space $S \subset \mathbb{R}^d$ is defined such that all link probabilities must fall with $[0, 1]$. Our proof technique follows a similar argument as the one used to prove their Proposition 4.3.

It should be noted that learnability of the link probability matrix for the sparse LPM could be established by applying results from *Universal Singular Value Thresholding* (Chatterjee, 2015; Xu, 2017). However, it is unclear how to extend such estimators to establish learnability of the latent positions; estimated probability matrices from universal singular value thresholding do not necessarily translate to a valid set of latent positions for a given link function.

Other related work includes Arias-Castro et al. (2018), which considers the problem of estimating latent distances between nodes when the functional form of the link probability function is unknown. They show that, if the link probability function is non-increasing and zero outside of a bounded interval, the lengths of the shortest paths between nodes can be used to consistently rank the distances between the nodes. Diaz et al. (2018) and Rocha et al. (2017) also propose estimators in similar settings with more specialized link functions. None of these approaches are appropriate for our case—we are interested in recovering the latent positions under the assumption K is known with positive support on the entire real line.

3.4.3 Learnability Results

Our learnability results assume the following criteria for a LPM:

1. The link probability function K is known, monotonically decreasing, differentiable, and upper bounded by $1-\epsilon$ for some $\epsilon > 0$.

2. The latent space $S \subseteq \mathbb{R}^d$.

3. There exists a known differentiable function $G(n)$ such that

$$I(||Z_n|| \leq G(n)) \xrightarrow{p} 1. \tag{3.11}$$

We refer to the above conditions as *regularity criteria* and refer to any LPM that meets them as *regular*. Note that criterion 3 implies that the sequence of latent positions is tight (Kallenberg, 2002, p. 66). The class of regular LPMs contains several popular LPMs. Notably, both rectangular and exchangeable LPMs due to Hoff et al. (2002) are regular, as shown in Lemmas 3.3.8 and Lemma 3.3.9.

Our approach for establishing learnability of Z involves proposing a particular estimator for Z which meets the learnability requirement as n grows. Our proposed estimator is a restricted maximum likelihood estimator for Z, provided by the following equation:

$$\hat{Z}(Y^n) = \text{argmax}_{z:||z_i|| \leq G(n) \forall i \in 1:n} L(z : Y^n) \tag{3.12}$$

where $L(z : Y^n)$ denotes the log likelihood of latent positions $z = (z_1, \dots z_n) \in \mathbb{R}^{n \times d}$ for a $n \times n$ adjacency matrix Y^n. We use $D^{\hat{Z}(Y^n)}$ and $P^{\hat{Z}(Y^n)}$ to denote the corresponding estimates of the squared distance matrix and link probability matrix. Note that the log likelihood $L(z : Y^n)$ is given by

$$L(z : Y^n) = \sum_{i=1}^{n} \sum_{j=1}^{n} Y_{ij}^n \log\left(K(||z_i - z_j||)\right) + (1 - Y_{ij}^n) \log\left(1 - K(||z_i - z_j||)\right). \tag{3.13}$$

To establish consistency, we first provide a concentration inequality for the maximum likelihood estimate of Z in Lemma 3.2.3. En route to deriving Lemma 3.2.3, we also derive inequalities for the associated squared distance matrix $D^Z \in \mathbb{R}^{n \times n}$ defined by $D_{ij}^Z = ||Z_i - Z_j||_F^2$ (Lemma 3.2.2) and the link probability matrix $P^Z \in [0,1]^{n \times n}$ defined by $P_{ij}^Z = K((D_{ij}^Z)^{1/2})$ (Lemma 3.2.1). We combine these results in Theorem 3.3 to provide conditions under which it is possible to consistently estimate Z, D^Z, and P^Z.

Our results are sensitive to the particular choices of link probability function K and upper bounding function G. For this reason, we introduce the following notation to communicate our results.

$$\alpha_n^K = \sup_{0 \le x \le 2G(n)} \frac{|K'(x)|}{|x| K(x) \epsilon}, \tag{3.14}$$

$$\beta_n^K = \sup_{0 \le x \le 2G(n)} \frac{x^2 K(x)}{K'(x)^2}, \tag{3.15}$$

where $K'(x)$ denotes the derivative of $K(x)$ and ϵ is given by the criteria on K imposed by regularity criterion 1.

Lemma 3.2.1. *Consider a sequence adjacency matrices Y^n generated by a regular LPM with $||Z_n|| \le G(n)$ for all n. Let $P^{\hat{Z}(Y^n)}$ denote the estimated link probability matrix obtained via $\hat{Z}(Y^n)$ from (3.12). Then,*

$$\mathbb{P}\left(||P^{\hat{Z}(Y^n)} - P^Z||_F^2 \ge 8e\alpha_n^K G(n)^2 n^{1.5}(d+2) \right) \le \frac{C}{n^2} \tag{3.16}$$

for some constant $C > 0$.

Proof. Provided in §3.6.4. □

Lemma 3.2.2. *Consider a sequence adjacency matrices Y^n generated by a regular LPM with $||Z_n|| \le G(n)$ for all n. Let $D^{\hat{Z}(Y^n)}$ denote the matrix of estimated squared distances obtained via $\hat{Z}(Y^n)$ from (3.12). Then,*

$$\mathbb{P}\left(||D^{\hat{Z}(Y^n)} - D^Z||_F^2 \ge 128e\alpha_n^K \beta_n^K G(n)^2 n^{1.5}(d+2) \right) \le \frac{C}{n^2} \tag{3.17}$$

for some constant $C > 0$.

Proof. Provided in §3.6.4. □

Establishing concentration of the estimated latent positions is complicated by the need to account for the minimization over all possible rotations, translations, and reflections. The following matrix, known as the double-centering matrix, is a useful tool to account for translations:

$$\mathcal{C}_n = I_n - \frac{1}{n} 1_n 1_n^T \tag{3.18}$$

Here, I_n denotes the n-dimensional identity matrix and 1_n denotes $n \times 1$ matrix consisting of ones.

Lemma 3.2.3. *Consider a sequence adjacency matrices Y^n generated by a regular LPM with $||Z_n|| \leq G(n)$ for all n. Furthermore, let $\lambda_1 \geq \cdots \geq \lambda_d$ denote the d nonzero eigenvalues of $\mathcal{C}_n ZZ^T \mathcal{C}_n$. Then,*

$$\mathbb{P}\left(\inf_{\substack{O\in\mathcal{O}_d \\ Q\in\mathcal{Q}_{nd}}} ||\hat{Z}(Y^n)O - Z - Q||_F^2 \geq \frac{(\lambda_1-\lambda_d)^2}{(4d)^{-1}\lambda_1} + \frac{512e(d+2)\left(d+8\frac{\lambda_1}{\lambda_d}\right)}{\frac{\lambda_d}{nG(n)^2}\left(\alpha_n^K\beta_n^K n^{0.5}\right)^{-1}}\right) \leq \frac{C}{n^2} \tag{3.19}$$

for some constant $C > 0$, where $\mathcal{O}_d$ denotes the space of orthogonal matrices on $\mathbb{R}^{d\times d}$, $\mathcal{Q}_{nd} \subset \mathbb{R}^{n\times d}$ is composed of matrices with n identical d-dimensional rows, and $\hat{Z}(Y^n)$ is obtained via (3.12).

Proof. Provided in §3.6.4. $\square$

These three concentration results can be translated into sufficiency conditions for learnability. We summarize these in Theorem 3.3.

Theorem 3.3. *A regular LPM that is $e(n)$-sparse in expectation has:*

1. *learnable link probabilities if $\alpha_n^K e(n)^{-1} n^{1.5} G(n)^2 \to 0$ as n grows.*
2. *learnable squared distances if $\beta_n^K \alpha_n^K n^{-0.5} G(n)^2 \to 0$ as n grows.*
3. *learnable latent positions if $n^{-1}(\lambda_1 - \lambda_d)^2 \lambda_1^{-1} \to 0$ and $\beta_n^K \alpha_n^K n^{0.5} G(n)^2 \lambda_1 \lambda_d^{-2} \to 0$ as n grows.*

Proof. Provided in §3.6.4. $\square$

It may seem counter-intuitive that the conditions for learnability of Z, P^Z and D^Z differ, even though their estimators are all derived from the same quantity. For example, if β_n^K grows quickly enough, the LPM may have learnable link probabilities but not squared distances. This disparity can be understood by considering the metrics implied by each form learnability.

Suppose that $\delta_{ij} = ||Z_i - Z_j||$ is very large. Then mis-estimating δ_{ij} by a constant $c > 0$ (i.e. $\hat{\delta}_{ij} = \delta_{ij} + c$) contributes $(2\delta_{ij}c + c^2)^2$ to the error in $||D^{\hat{Z}} - D^Z||_F^2$. This contribution to the error is sizable, and can hinder convergence if made too often. However, the influence of the same mistake on $||P^{\hat{Z}} - P^Z||_F^2$ is minor; because the probability $K(\delta)$ is already small for large δ, $(K(\delta + c) - K(\delta))^2$ does not contribute much to the error. For small distances, the opposite may be true; a small mistake in estimated distance may lead to a large mistake in estimated probability. Thus, learnability of squared distances penalizes mistakes differently than learnability of link probabilities. However, there are typically far more large distances than small distances, meaning that the distance metric imposed by learnability of link probabilities is typically less stringent than for learnability of squared distances.

Theorem 3.3 can be used to establish Corollary 1, a learnability result for rectangular LPMs.

Corollary 1. *Consider a d-dimensional rectangular LPM with* $g(n) = n^{p/d}$ *and link probability function* $K(\delta) = (C + \delta^2)^{-a}$ *for some* $C > 0$, *where* $a > max(\{d/2, 1\})$ *and* $0 \leq p \leq 1$. *Such a network has learnable*

1. *link probabilities if* $2p < (1 + 2/d)^{-1}$,
2. *distances if* $2p < d\,(2a + 6)^{-1}$,
3. *latent positions if* $2p < d\,(2a + 4)^{-1}$.

Thus, for any $b \in (1.5, 2]$, *it is possible to construct a LPM that is projective,* n^b*-sparse in expectation, and has learnable latent positions, distances, and link probabilities.*

Proof. Provided in §3.6.4. □

Corollary 1, combined with the projectivity of rectangular LPMs, guarantees the existence of a LPM that is projective, learnable, and sparse for any sparsity level that is denser than $n^{3/2}$-sparse in expectation. Thus, we have shown that we have met our desiderata for LPMs laid out in the introduction.

Perhaps surprisingly, our result in Corollary 1 depends upon the dimension of the latent space. The higher the dimension, the richer the levels of learnable sparsity. Moreover, the learnability results in Theorem 3.3 only apply to rectangular LPMs with link functions that decay polynomially. The β_n^K term is too large for the exponential-style decays that are commonly considered in practice (Hoff et al., 2002; Rastelli et al., 2016). We elaborate on these points in §3.5.3.

In contrast, it is possible to prove learnability of exchangeable LPMs with exponentially decaying K. Corollary 2 guarantees learnability of the exchangeable LPM for two exponential-style link functions. As far as we are aware, these are the first result learnability results for the latent positions for the original exchangeable LPM.

Corollary 2. *Consider a LPM on* $S = \mathbb{R}^d$ *with the latent positions distributed according a isotropic Gaussian random vector with variance* σ^2. *Suppose that the link probability function is given by either*

$$K(\delta) = (1 + \exp(\delta^2))^{-1} \text{ or } K(\delta) = \tau e^{-\delta^2}. \tag{3.20}$$

for $\tau \in (0, 1)$. *Such a network has learnable link probabilities, distances, and latent positions provided that* $\sigma^2 < 1/4$.

Proof. Provided in §3.6.4. □

Notably, the set of link functions in Corollary 2 does not include the traditional expit link function that was suggested in the original paper LPM by Hoff et al. (2002). The expit class of link functions implies a value α_n^k—defined as in (3.14)—that is unbounded (see Table 3.1 for a summary of the α_n^k and β_n^K values for various link functions), meaning that Lemma 3.3 cannot be applied to prove learnability for this class of LPMs. This does not necessarily mean that expit LPMs are not learnable, just that determining their learnability remains an open problem. Note however, that some classes of sparse LPMs (such as the example considered in Theorem 3.4 (§3.6.5)) *are* provably unlearnable. We elaborate on this point in §3.5.3.

The results in Theorem 3.3 can also be used to obtain learnability results for more specialized LPMs such as sparse graphon-based LPMs. We provide such a result in §3.5.1 when comparing sparse graphons with our approach.

$K(x) =$	$K'(x) =$	$\frac{\lvert K'(x)\rvert}{\lvert x\rvert K(x)\epsilon} =$	$\frac{x^2\lvert K(x)\rvert}{K'(x)^2} =$	$\alpha_n^K \sim$	$\beta_n^K \sim$
$\frac{1}{1+e^x}$	$\frac{-e^x}{(1+e^x)^2}$	$\frac{e^x}{\epsilon x(1+e^x)}$	$\frac{(1+e^x)^3x^2}{e^{2x}}$	∞	$\Theta(e^{G(n)}G(n)^2)$
$\frac{1}{1+e^{x^2}}$	$\frac{-2xe^{x^2}}{(1+e^{x^2})^2}$	$\frac{2e^{x^2}}{\epsilon(1+e^{x^2})}$	$\frac{(1+e^{x^2})^3}{4e^{2x^2}}$	$\Theta(1)$	$\Theta(e^{G(n)^2})$
τe^{-x^2}	$-2x\tau e^{-x^2}$	2	$\frac{e^{x^2}}{4\tau}$	$\Theta(1)$	$\Theta(e^{G(n)^2})$
$\frac{1}{(c+x^2)^a}$	$\frac{-2ax}{(c+x^2)^{a+1}}$	$\frac{2a}{c+x^2}$	$\frac{(c+x^2)^{a+2}}{4a^2}$	$\Theta(1)$	$\Theta(G(n)^{2a+4})$

Table 3.1: Values of α_n^K and β_n^K for different choices of link function $K(x)$

3.5 Comparisons and Remarks

It would seem that existing tools for constructing sparse graph models, such as the sparse graphon framework (Bollobás et al., 2007; Borgs et al., 2014) or the graphex framework (Caron and Fox, 2014; Veitch and Roy, 2015; Borgs et al., 2014) could be used to develop sparse latent position models. Unfortunately, these approaches introduce sparsity in ways that produce undesirable side effects for LPMs. We now describe both the sparse graphon framework (§3.5.1) and the graphex framework (§3.5.2), emphasizing how they result sparse LPMs which fail to meet our desiderata of projectivity and learnability. Finally, we make some concluding remarks on the results we have derived this article (§3.5.3).

3.5.1 Sparse Graphon-based Latent Position Models

Borgs et al. (2014) proposed a modification of graphon models to allow sparse graph sequences. Exchangeable LPMs are within the graphon family, so it is straightforward to specialize their approach to define sparse graphon-based LPMs.

As in exchangeable latent space models, the latent positions for a sparse graphon-based LPM are each drawn from a common distribution f, independently of each other the number of nodes n. However, the link probability function $\mathbb{P}(Y_{ij} = 1|Z_i, Z_j) = K_n(\rho(Z_i, Z_j))$ can depend on n. Specifically, $K_n(x) = \min(\{s_nK(x), 1\})$ where $K : \mathbb{R}_+ \to \mathbb{R}_+$ and $(s_n)_{1\ldots\infty}$ is a non-increasing sequence. These models express sparse graph sequences, with the sequence $(s_n)_{1\ldots\infty}$ controlling the sparsity of the resultant graph sequence.

Proposition 5. *Sparse graphon-based latent space models define a $n^2 s_n$-sparse in expectation graph sequence.*

Proof. Proof provided in §3.6.3. □

However, the resultant sparse graph sequences are no longer projective.

Proposition 6. *Sparse-graphon latent space models do not define a projective sequence of models if $(s_n)_{n=1\ldots\infty}$ is not constant.*

Proof. Proof provided in §3.6.2. □

The learnability results in Theorem 3.3 can also be applied to sparse-graphon based LPMs.

Corollary 3. *Consider the following sparse graphon-based version of the exchangeable LPM. Let $S = \mathbb{R}^d$ with the latent positions distributed according a isotropic Gaussian random vector with any variance $\sigma^2 < 1/4$. Suppose that the link probability function is given by either*

$$K_n(\delta) = n^{-p}(1 + \exp(\delta^2))^{-1} \text{ or } K_n(\delta) = \tau n^{-p} e^{-\delta^2} \tag{3.21}$$

for $\tau \in (0, 1)$, $0 \leq p \leq 1$. Such a network has learnable link probabilities, squared distances, and latent positions if $p < 1/2 - 2\sigma^2(1 + c)$ for $c > 0$. Given an appropriate σ^2, this LPM can be both n^b-sparse and for $b \in (1.5, 2]$.

Proof. Proof provided in §3.6.4 □

Thus, sparse graphon-based LPMs can achieve learnability under the same rate or sparsity as we derived for rectangular LPMs in Corollary 1. However, this formulation allows for link probability functions with lighter tails, and holds for all dimensions d. Thus, there may be a trade-off between projectivity and learnability under light-tailedness of the link probability function.

It is also worth noting that the sparse graph representation of Bollobás et al. (2007) is more general than the sparse graphon representation described above. It allows for latent variables assigned defined through a point process rather than generated independently from the same distribution. For LPMs, this set-up equates to the traditional random connection model (§3.2.5).

3.5.2 Comparison with the Graphex Framework

Beyond the random connection model (Meester and Roy, 1996), there has been a recent renewed interest in using point processes to define networks. This was primarily spurned by the developments in Caron (2012) and Caron and Fox (2014) in which they propose a new graph framework—based on point processes—for infinitely exchangeable and sparse networks. This approach was generalized as the graphex framework in

Veitch and Roy (2015). Other variants and extensions of this work include Borgs et al. (2016); Herlau et al. (2016); Palla et al. (2016); Todeschini et al. (2016).

In the graphex framework, a graph is defined by a homogeneous Poisson process on an augmented space $\mathbb{R}_+ \times \mathbb{R}_+$, with the points representing nodes. The two instances of $\mathbb{R}_+$ play the roles of the parameter space and the auxiliary space. The parameter space determines the connectivity of nodes through a function $W : \mathbb{R}_+^2 \to [0, 1]$. Connectivity is independent of the auxiliary dimension $\mathbb{R}_+$ that determines the order in which the nodes are observed. Clearly, our sparse LPM set-up shares many similarities with the graphex framework. Both assign latent variables to nodes according to a homogeneous Poisson process defined on a space composed of a parameter space to influence connectivity and an auxiliary space to influence order of node arrival. The graphex is defined in terms of a one-dimensional parameter space, but it can be equivalently expressed as a multi-dimensional parameter space as we do for the sparse LPM. The link probability function K for the sparse LPM depends solely on the distance between points, but it would be straightforward to extend to the more general set-up for W as in the graphex. However, it would take additional work to determine the sparsity levels and learnability properties of such graphs.

The major difference between our framework and the graphex framework is how a finite subgraph is observed. To observe a finite graphex-based graph, one restricts the point process to a window $\mathbb{R}_+ \times [0, \nu]$. Here, the restriction is limited to the auxiliary space, with the parameter space remaining unrestricted. This alone is not enough to lead to a finite graph, as a unit rate Poisson process on $\mathbb{R}_+ \times [0, \nu]$ still has an infinite number of points almost-surely. To compensate, an additional criterion for node visibility is included. A node is visible only if it has at least one neighbor. For some choices of W, this results in a finite number of visible nodes for a finite ν. Veitch and Roy (2015) show that the expected number of nodes n_ν and edges e_ν are given by

$$\mathbb{E}\left(n_\nu\right) = \nu \int_0^\infty 1 - \exp\left(-\nu \int_{\mathbb{R}_+} W(x, y)\mathrm{d}y\right)\mathrm{d}x, \tag{3.22}$$

$$\mathbb{E}\left(e_\nu\right) = \frac{1}{2}\nu^2 \int_0^\infty \int_0^\infty W(x, y)\mathrm{d}x\mathrm{d}y \tag{3.23}$$

respectively. Thus, the degree of sparsity in the graph is controlled through the definition of W. Clearly, for a finite-node restriction to be defined, the two dimensional integral over W in (3.23) must be finite. Otherwise, the number of nodes is infinite for any ν.

A sparse graphex-based LPM cannot be implemented in the naive manner because, if W is solely a function of distance between nodes, the two dimensional integral (3.23) is infinite. One modification to prevent this to modify W to have bounded support, e.g. $W(x, y) = K(|x - y|)I(0 \leq x, y \leq C)$. However, this framework is equivalent to the graphon framework and results in dense graphs (Veitch and Roy, 2015). It does not define a sparse LPM.

Alternatively, we could relax the graphex such that latent positions are generated according to an inhomogeneous point process over the parameter sparse. This can be done though the definition of W. For instance, consider

$$W(x, y) = K\left(|\exp(x) - \exp(y)|\right). \tag{3.24}$$

with K being the link probability function as defined in the traditional LPM. In this set-up, W can be viewed as the composition of two operations. First, an exponential transformation is applied to the latent positions resulting in an inhomogeneous rate function given by $f(x) = 1/(1+x)$. Then, we proceed as if it were a traditional LPM in this new space, connecting the nodes according to K on their transformed latent positions. Finally, the isolated nodes are discarded. This approach defines a sparse and projective latent space model, with the level of sparsity controlled by K. Though (3.22) and (3.23) provide a means with which to calculate the sparsity level, these expressions do not yield analytic solutions for most K. As a result, the graphex framework is far more difficult to work with when defining sparse LPMs; they lack the straightforward control over the level of sparsity provided by the growth function $g(t)$ in rectangular LPMs.

Furthermore, it is difficult to apply the tools derived in Theorem 3.3 to establish learnability for graphex-based LPMs. The difficulty stems from the fact that regularity requires a probably bound on the distance between the first n nodes observed and the origin. Because of the irregular sampling scheme in which isolated nodes are discarded, it is difficult to establish such a bound for the graphex. Furthermore, any such bound is usually large due to the fact the latent positions at any n are generated according to an improper distribution. For this reason, whether or not such graphex-based LPMs are learnable is an open problem.

3.5.3 Remarks

We have established a new framework for sparse and projective latent position models that enables straightforward control the level of sparsity. The sparsity is a result of assuming the latent positions of nodes are a realization of a Poisson point process on an augmented space, and that the growing sequence of graphs is obtained by restricting observable nodes to those with positions in a growing sequence of nested observation windows.

The notion of projectivity we consider here is slightly weaker than the one usually considered in the literature (e.g. Shalizi and Rinaldo (2013)). Our definition requires consistency under marginalization of the most recently arrived node, rather than consistency under marginalization of any node. We do not consider this to be a major limitation—if the entire sequence of graphs were observed, the order of the nodes would be apparent.

In practice, only a single network of finite size is available when conducting inference. However, in these cases the order of nodes is not required—we make no use of it when defining the maximum likelihood estimator. A finite observation from our new sparse LPM is equivalent to finite observation from an equivalent exchangeable LPM with f given by the shape of $H(n_t)$. This follows from Lemma 3.3.1 which indicates that the distribution of latent positions can be viewed as iid after conditioning on the number of nodes and randomly permuting the ordering. This means that the analysis and inference tools developed for exchangeable LPMs extend immediately to our approach when analyzing a single, finite network. From this viewpoint, we have merely proposed a different asymptotic regime for studying the same classes of models available under the exchangeability assumption.

Theorem 3.3 provides some consistency results under this asymptotic regime. However, the rates of learnability we achieved are upper bounds—the inequalities in Lemmas 3.2.1-3.2.3 are not necessarily tight. They are derived to hold even for the worse-case regular LPMs regardless of how the latent positions are generated. We demonstrate in Theorem 3.4 (§3.6.5) that there are some classes regular LPMs for which it is impossible to learn the latent positions. This class of models includes any regular LPMs with $G(n) = n^{p/d}$ and K exponentially decreasing. In these cases, it is possible for the LPM to result in graphs which are disconnected with probability trending to one by clustering the latent positions at two extreme points of the space.

Though the regularity criteria technically allow for such instances by placing no assumptions on the distribution of Z besides bounded norms, these clusters arise with vanishing probability when the latent positions are assumed to follow a homogeneous Poisson process such as in rectangular LPMs. For this reason, a future research direction to explore is to establish better learnability rates for rectangular LPMs by tightening the bounds Lemmas 3.2.1-3.2.3 through assumption on the distribution of the latent positions.

3.6 Proofs

3.6.1 Intermediary Results

The following are useful lemmas toward establishing the main results in this article.

Lemma 3.3.1. *Restriction Theorem in Kingman (1993, p. 17) Let Λ be a Poisson process with mean measure μ on S, and let S_1 be a measurable subset of S. Then the random countable set*

$$\Lambda_1 = \Lambda \cap S_1 \tag{3.25}$$

can be regarded as a Poisson process on S with mean measure

$$\mu_1(A) = \mu(A \cap S_1) \tag{3.26}$$

or as a Poisson process on S_1 possessing a mean measure that is the restriction of μ to S_1.

Lemma 3.3.2. *For a rectangular LPM, the number of nodes which are visible at time t is Poisson distributed with mean t.*

Proof. According to Lemma 3.3.1, the latent positions of nodes visible at time t follow a unit-rate Poisson process over $H(t)$. Therefore, the number of nodes is Poisson distributed with expectation equal to the volume of $H(t)$, which is t. $\square$

Lemma 3.3.3. *Let t_n denote the arrival time of the nth node in a sparse LPM. Then, $t_n \sim$ Gamma(n, 1) if $H(t)$ has volume t.*

Proof. Let $n_t = |\Psi \cap H(t)|$ where Ψ denotes the unit rate Poisson process of latent positions. Then, it is straightforward to verify that n_t follows a one-dimensional homogeneous Poisson process on the positive real line. Note that t_n can be equivalently expressed as

$$t_n = \inf\left\{t \geq 0 : |\Psi \cap H(t)| = n\right\}. \tag{3.27}$$

That is, t_n is the index of the smallest observation window containing n nodes for all positive integers n. Under this perspective, t_n can be viewed as a stopping time of n_t. It is well-known that t_1, the first arrival time of a unit-rate Poisson process, follows an exponential distribution with rate 1. Then, by the strong Markov property of Poisson processes $t_n - t_{n-1}$ is identical in distribution to t_1. Thus, t_n is equivalent to the sum of n independent exponential distributions, meaning it follows Gamma(n, 1). $\square$

Lemma 3.3.4. *Consider a sparse rectangular LPM. Let z denote the latent position of a node chosen uniformly at random of the nodes visible at time t. Then z follows a uniform distribution over $[-g(t), g(t)]^d$.*

Proof. If a node is visible at time t, its latent position and auxiliary coordinate pair (z, r) are a point in a unit-rate Poisson process restricted H(t). By Lemma 3.3.1, this point process is a Poisson process with unit rate over the restricted space. Thus, if a node is visible at time (z, r), it is uniformly distributed over $H(t) = [-g(t), g(t)]^d \times [0, t/(2g(t))^d]$. Marginalizing r provides the result. $\square$

Lemma 3.3.5. *Let K be a decreasing non-negative function such that*

$$0 < \int_0^\infty r^{d-1} K(r)\, dr < \infty, \tag{3.28}$$

for $d \in \mathbb{Z}_+$. Then,

$$0 < \int_{y \in [-B,B]^d} K(||x - y||)\, dy < \infty \tag{3.29}$$

for any $B \in \mathbb{R}_+$.

Proof. Note that for all decreasing positive functions K, the function

$$R(x) = \int_{y \in [-B,B]^d} K(||y-x||)\mathrm{d}y \tag{3.30}$$

is maximized when x is at the origin. Thus, for all $x \in \mathbb{R}^d$,

$$\begin{aligned}
\int_{y \in [-B,B]^d} K(||y-x||)\mathrm{d}y &\leq \int_{y \in [-B,B]^d} K(||y||)\mathrm{d}y && (3.31)\\
&\leq \int_{y \in \mathbb{R}^d : ||y|| < B} K(||y||)\mathrm{d}y && (3.32)\\
&\propto \int_0^B r^{d-1} K(r)\mathrm{d}r && (3.33)\\
&< \infty. && (3.34)
\end{aligned}$$

The positivity follows from K being non-negative. □

Lemma 3.3.6. *Consider a rectangular LPM, with t_i denoting the arrival time of the ith node. Let π denote permutation chosen uniformly at random from all permutations on $\{1, \ldots, n-1\}$. Then, conditional on $t_n = T$, each $t_{\pi(i)}$'s marginal distribution is uniform on $[0,T]$ for $i = 1, \ldots, n-1$. Consequently, the latent position $Z^{\pi(i)}$ of node $\pi(i)$ is uniformly distributed on $[-g(T), g(T)]^d$.*

Proof. Let $(w_i)_{i=1,\ldots,n}$ denote the inter-arrival of times of the nodes. That is, $w_1 = t_1$ and $w_i = t_i - t_{i-1}$. As argued in the proof of Lemma 3.3.3, each w_i is exponentially distributed. Thus, the density of $t_1, \ldots, t_{n-1}$ given $t_n = T$ satisfies:

$$f(t_1, \ldots, t_{n-1} | t_n = T) \propto I(0 \leq t_1 \leq t_2 \leq \cdots \leq t_{n-1} \leq t_n) \tag{3.35}$$

which is the same density as the order statistics of a uniform distribution on $[0,T]$. Thus, a randomly chosen waiting time $t_{\pi(i)}$ is uniformly distributed on $[0,T]$. Let $r^{\pi(i)}$ denote the auxiliary coordinate of node $\pi(i)$. It follows that $\mathbb{P}((Z^{\pi(i)}, r^{\pi(i)}) \in [-g(a), g(a)]^d \times [0, a/g(a)^d]) = a/T$ for all $0 \leq a \leq T$. It follows that $Z^{\pi(i)}$ is uniformly distributed on $[-g(T), g(T)]^d$. □

Lemma 3.3.7. *Consider a rectangular sparse LPM model restricted to $H(t_n)$ such that n nodes are visible. Let $\{Z_1, \ldots, Z_n\}$ denote the latent positions of these nodes. Let $\delta^{(n)} = \max_{i=1,\ldots n} ||Z_i||$ denote the largest Euclidean distance between a visible node's latent position and the origin. Then,*

$$\mathbb{P}(\delta^{(n)} > \sqrt{d}g(n + \sqrt{n\log(n)})) \leq \log(n)^{-1} \tag{3.36}$$

indicating that

$$\lim_{n\to\infty} \mathbb{P}(\delta^{(n)} > \sqrt{d}g(n + \sqrt{n\log(n)})) \to 0. \tag{3.37}$$

Consequently,

$$\lim_{n\to\infty} \mathbb{P}(\delta^{(n)} > 2\sqrt{d}g(n)) \to 0. \tag{3.38}$$

Proof. Let Z_{ij} denote the jth latent coordinate of node i. By construction, $||Z_{ij}|| \leq g(t_n)$ for any $i \leq n, j \leq d$. Thus, $\delta^{(n)} \leq \sqrt{d}g(t_n)$. By Lemma 3.3.3, know that $t_n \sim$ Gamma(n, 1). By Chebyshev inequality,

$$\mathbb{P}(|t_n - n| > \sqrt{n\log(n)}) \leq \log(n)^{-1} \tag{3.39}$$

$$\Rightarrow \mathbb{P}(t_n > n + \sqrt{n\log(n)}) \leq \log(n)^{-1} \tag{3.40}$$

$$\Rightarrow \mathbb{P}(g(t_n) > g(n + \sqrt{n\log(n)})) \leq \log(n)^{-1} \tag{3.41}$$

$$\Rightarrow \mathbb{P}(d^{-1/2}\delta^{(n)} > g(n + \sqrt{n\log(n)})) \leq \log(n)^{-1} \tag{3.42}$$

$$\Rightarrow \mathbb{P}(\delta^{(n)} > \sqrt{d}g(n + \sqrt{n\log(n)})) \leq \log(n)^{-1} \tag{3.43}$$

The result in (3.37) follows from taking the limit, and the result in (3.38) follows from $g(n + \sqrt{n\log(n)}) \leq 2g(n)$ for all non-decreasing $g(n) = n^{p/d}$ and $n \geq 1$. ☐

Lemma 3.3.8. *Rectangular LPMs are regular with* $G(n) = 2\sqrt{d}n^{p/d}$.

Proof. Criteria 1 and 2 of a regular LPM hold by definition of a rectangular LPM. Lemma 3.3.7 guarantees that satisfaction of criterion 3. ☐

Lemma 3.3.9. *Consider a LPM on* $S = \mathbb{R}^d$*, with the latent position vectors independently and identically distributed according to an isotropic Gaussian with* σ^2*. If the link probability function is upper bounded by* $1 - \epsilon$*, then the LPM is regular with* $G(n) = \sqrt{2\sigma^2(1+c)\log(n)}$ *for any* $c > 0$.

Proof. Criteria 1 and 2 for regularity hold trivially. Thus, it is sufficient to prove criteria 3 for the prescribed $G(n)$. Let $Z_1, \ldots, Z_n$ denote the latent positions. Then $||Z_i||^2/\sigma^2$ follows a χ^2 distribution with parameter d. We can apply the concentration inequality on χ^2 random variables implied by Laurent and Massart (2000, Lemma 1), to conclude, for any $t > 0$

$$\mathbb{P}\left(||Z_i|| > \sigma\sqrt{d + 2t + 2\sqrt{dt}}\right) \leq \exp(-t) \tag{3.44}$$

$$\Rightarrow \mathbb{P}\left(||Z_i|| > \sqrt{2}\sigma(u + \sqrt{d})\right) \leq \exp(-u^2) \tag{3.45}$$

for any $u > 0$. Applying the union bound results in

$$\mathbb{P}\left(\max_{1\leq i\leq n} ||Z_i|| > \sqrt{2}\sigma(u+\sqrt{d})\right) \leq n\exp\left(-u^2\right) \tag{3.46}$$

As long as $u^2 \geq (1+c)\log(n)$, for $c > 0$, the above probability goes to 0. Note that $\sqrt{2\sigma^2(1+c)\log(n)}$ dominates $\sqrt{2d\sigma^2}$ as n grows. Thus, $G(n) = \sqrt{2\sigma^2(1+c)\log(n)}$ yields the desired result for $c > 0$. □

Lemma 3.3.10. *Symmetrization Lemma*
Let

$$\Omega = \left\{X \in \mathbb{R}^{n\times d} : ||X_i|| \leq G(n)\forall i \in [n]\right\} \tag{3.47}$$

for $G(n) \in \mathbb{R}_+$. Let $L(x : Y^n)$ denote the log likelihood of the latent positions $x \in \Omega$ as defined in (3.13) for a link function K. Let $\bar{L}(x) = L(x : Y^n) - L(\boldsymbol{0} : Y^n)$ and $\mathbb{E}(\bar{L}(x))$ denote its expectation. Then, for $h \geq 1$,

$$\begin{aligned}&\mathbb{E}\left(\sup_{x\in\Omega}|\bar{L}(x) - \mathbb{E}(\bar{L}(x))|^h\right)\\ \leq\ & 2^h\mathbb{E}\left(\sup_{x\in\Omega}\left|\sum_{j=1}^{n}\sum_{i=1}^{n} R_{ij}\left(Y_{ij}^n\log\left(\frac{K(\delta_{ij}^x)}{K(0)}\right) + (1-Y_{ij}^n)\log\left(\frac{1-K(\delta_{ij}^x)}{1-K(0)}\right)\right)\right|^h\right)\end{aligned} \tag{3.48}$$

where R denotes an array of independent Rademacher random variables and $\delta_{ij}^x = ||x_i - x_j||$.

Proof. This proof follows the same argument of that of Ledoux and Talagrand (1991, Lemma 6.3). Let $\bar{L}_{ij}(x)$ denote the contribution of Y_{ij}^n to the standardized log likelihood. Thus,

$$\bar{L}(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\bar{L}_{ij}(x) \text{ and} \tag{3.49}$$

$$\bar{L}(x) - \mathbb{E}(\bar{L}(x)) = \sum_{i=1}^{n}\sum_{j=1}^{n}\ell_{ij}(x) \tag{3.50}$$

where each $\ell_{ij}(x) = \bar{L}_{ij}(x) - \mathbb{E}(\bar{L}_{ij}(x))$ is a zero mean random variable. For each i, j, let $\ell'_{ij}(x)$ denote a random variable that is independently drawn from the distribution of $\ell_{ij}(x)$. Then, $\ell_{ij}(x) - \ell'_{ij}(x)$ is a symmetric zero mean random variable with the same distribution as $R_{ij}(\ell_{ij}(x) - \ell'_{ij}(x))$. Moreover, we can view $\sup_{x\in\Omega}|f(x)|$ as defining a norm on the Banach space of functions $f : \Omega \to \mathbb{R}$. These facts, along with the convexity of exponentiating by h, imply the following.

$$\mathbb{E}\left(\sup_{x\in\Omega}\left|\bar{L}(x)-\mathbb{E}(\bar{L}(x))\right|^h\right) \tag{3.51}$$

$$= \mathbb{E}\left(\sup_{x\in\Omega}\left|\sum_{i=1}^n\sum_{j=1}^n \ell_{ij}(x)\right|^h\right) \tag{3.52}$$

$$\leq \mathbb{E}\left(\sup_{x\in\Omega}\left|\sum_{i=1}^n\sum_{j=1}^n \ell_{ij}(x)-\ell'_{ij}(x)\right|^h\right) \text{ (cf. (Ledoux and Talagrand, 1991, Equation 2.5))} \tag{3.53}$$

$$= \mathbb{E}\left(\sup_{x\in\Omega}\left|\sum_{i=1}^n\sum_{j=1}^n R_{ij}\left(\ell_{ij}(x)-\ell'_{ij}(x)\right)\right|^h\right) \tag{3.54}$$

$$= \mathbb{E}\left(\left|\sup_{x\in\Omega}\sum_{i=1}^n\sum_{j=1}^n R_{ij}\left(\bar{L}_{ij}(x)-\bar{L}'_{ij}(x)\right)\right|^h\right) \tag{3.55}$$

$$\leq \mathbb{E}\left(\left(\sup_{x\in\Omega}\left|\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}_{ij}(x)\right| + \sup_{x\in\Omega}\left|\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}'_{ij}(x)\right|\right)^h\right) \tag{3.56}$$

$$\leq \mathbb{E}\left(\frac{1}{2}\sup_{x\in\Omega}\left|2\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}_{ij}(x)\right|^h + \frac{1}{2}\sup_{x\in\Omega}\left|2\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}'_{ij}(x)\right|^h\right) \tag{3.57}$$

by convexity of exponentiating by h (3.58)

$$= \frac{1}{2}\mathbb{E}\left(\sup_{x\in\Omega}\left|2\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}_{ij}(x)\right|^h\right) + \frac{1}{2}\mathbb{E}\left(\sup_{x\in\Omega}\left|2\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}'_{ij}(x)\right|^h\right) \tag{3.59}$$

$$= 2^h\mathbb{E}\left(\sup_{x\in\Omega}\left|\sum_{i=1}^n\sum_{j=1}^n R_{ij}\bar{L}_{ij}(x)\right|^h\right). \tag{3.60}$$

The result follows from the definitions of the $\bar{L}_{ij}$. □

Lemma 3.3.11. *Contraction Theorem (Ledoux and Talagrand, 1991, Theorem 4.12).*
Let $F:\mathbb{R}_+\to\mathbb{R}_+$ be convex and increasing. Let $\phi_i:\mathbb{R}\to\mathbb{R}$ for $i\leq N$ satisfy $\phi_i(0)=0$ and $|\phi_i(s)-\phi_i(t)|\leq |s-t|$ for all $s,t\in\mathbb{R}$. Then, for any bounded subset $\Omega\subset\mathbb{R}$,

$$\mathbb{E}\left(F\left(\frac{1}{2}\sup_{t\in\Omega^N}\left|\sum_{i=1}^N R_i\phi_i(t_i)\right|\right)\right) \leq \mathbb{E}\left(F\left(\sup_{t\in\Omega^N}\left|\sum_{i=1}^N R_i t_i\right|\right)\right) \tag{3.61}$$

where $R_1,\dots,R_N$ denote independent Rademacher random variables.

Corollary 4. *Let R denote an $n\times n$ array of independent Rademacher random variables, $K:\mathbb{R}_+\to[0,1-\epsilon]$ denote a link function that satisfies the regularity criteria in §3.4.3 (i.e. monotonically decreasing, differentiable function that is upper bounded by $1-\epsilon$ for some ϵ), and*

$$\Omega = \left\{X\in\mathbb{R}^{n\times d} : ||X_i|| \leq G(n) \text{ for all } i\in[n]\right\} \tag{3.62}$$

with $G(n) \in \mathbb{R}_+$, and $Y^n \in \{0,1\}^{n\times n}$. Define α_n^K as in (3.14). That is,

$$\alpha_n^K = \sup_{0\le x\le 2G(n)} \frac{|K'(x)|}{|x|K(x)\epsilon}. \tag{3.63}$$

Then,

$$\mathbb{E}\left(\sup_{x\in\Omega}\left|\sum_{j=1}^n\sum_{i=1}^n R_{ij}\left(Y_{ij}^n \log\left(\frac{K(\delta_{ij}^x)}{K(0)}\right) + (1-Y_{ij}^n)\log\left(\frac{1-K(\delta_{ij}^x)}{1-K(0)}\right)\right)\right|^h\right) \tag{3.64}$$

$$\le (2\alpha_n^K)^h \mathbb{E}\left(\sup_{x\in\Omega}\left(\left|\sum_{j=1}^n\sum_{i=1}^n R_{ij}||x_i - x_j||^2\right|^h\right)\right) \tag{3.65}$$

for $h \ge 1$, where $\delta_{ij}^x = ||x_i - x_j||$.

Proof. We can apply Lemma 3.3.11 to obtain this result as follows.

For all $x \in \Omega$, $i,j \in [n]$, we know by the triangle inequality that $||x_i - x_j||^2 \le 4G(n)^2$. Moreover, $K(2G(n)) \le K(||x_i - x_j||) \le 1-\epsilon$ because K is regular. A Taylor expansion of $\log(K(\sqrt{\cdot}))$ around 0 reveals that

$$\log(K(\sqrt{u})) - \log(K(\sqrt{0})) \tag{3.66}$$

$$= \frac{uK'(\sqrt{w})}{2\sqrt{w}K(\sqrt{w})} \text{ for some } w \in [0, 4G(n)^2] \tag{3.67}$$

$$= \frac{uK'(v)}{2vK(v)} \text{ for some } v \in [0, 2G(n)]. \tag{3.68}$$

Taking the supremum over possible values of v, it follows that

$$\left|\frac{\log(K(\sqrt{u})) - \log(K(\sqrt{0}))}{\alpha_n^K}\right| \le u. \tag{3.69}$$

Similarly, a Taylor expansion of $\log(1-K(\sqrt{\cdot}))$ around 0 yields

$$\log(1-K(\sqrt{u})) - \log(1-K(\sqrt{0})) \tag{3.70}$$

$$= \frac{-uK'(\sqrt{w})}{2v(1-K(\sqrt{w}))} \text{ for some } w \in [0, 4G(n)^2] \tag{3.71}$$

$$= \frac{-uK'(v)}{2v(1-K(v))} \text{ for some } v \in [0, 2G(n)]. \tag{3.72}$$

Similarly, taking the supremum over possible values of v yields

$$\left|\frac{\log(1-K(\sqrt{u}))-\log(1-K(\sqrt{0}))}{\alpha_n^K}\right| \leq u. \tag{3.73}$$

Together, we have

$$\left|\frac{Y_{ij}^n \log\left(\frac{K(u)}{K(0)}\right)+(1-Y_{ij}^n)\log\left(\frac{1-K(u)}{1-K(0)}\right)}{\alpha_n^K}\right| \leq u. \tag{3.74}$$

Moreover, for any i, j, the function on the lefthand side is 0 at $u = 0$. Thus, the function meets the criteria required of the ϕ functions in Lemma 3.3.11 and the result follows from convexity of exponentiating by h. □

Lemma 3.3.12. *Let $\Sigma, \hat{\Sigma} \in \mathbb{R}^{n\times n}$ be symmetric, with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and $\hat{\lambda}_1 \geq \cdots \geq \hat{\lambda}_n$, respectively. Furthermore, assume there is a $d \leq n$ such that $\lambda_d > \lambda_{d+1} = \cdots \lambda_n = 0$. Let $V, \hat{V} \in \mathbb{R}^{n\times d}$ have orthonormal columns satisfying $\Sigma V_j = \lambda_j$ and $\hat{\Sigma} V_j = \hat{\lambda}_j$ for $j \in \{1, \ldots, d\}$. Then, there exists an orthogonal matrix $O \in \mathbb{R}^{d\times d}$ such that*

$$||\hat{V}O - V||_F \leq \frac{2^{3/2}||\hat{\Sigma}-\Sigma||_F}{\lambda_d}. \tag{3.75}$$

Proof. This is a special case of the Davis-Kahan Theorem (Yu et al., 2014, Theorem 2). □

3.6.2 Projectivity Proofs

Proof of Proposition 1

Proof. Let Y^{n_1} and Y^{n_2} denote random graphs with n_1 and n_2 nodes ($n_1 < n_2$) generated according to an exchangeable LPM, and let $\mathbb{P}^{n_1}$ and $\mathbb{P}^{n_2}$ be their corresponding distributions. Let Z_i^j denote the random latent position of node i in Y^j for $j = n_1, n_2$. By definition, $Z_i^{n_1}$ and $Z_i^{n_2}$ are iid draws from the same distribution f on S. Thus, the $(Z_i^j)_{i=1\ldots n_1}$ have identical distributions for each j. As a result, $K(\rho(Z_{i_1}^{n_1}, Z_{i_1}^{n_1}))$ has the same distribution as $K(\rho(Z_{i_2}^{n_1}, Z_{i_2}^{n_1}))$ for any $1 \leq i_1, i_2 \leq n_1$. Because the distributions for each dyad coincide, the distributions over adjacency matrices coincide. □

Proof of Proposition 4

Proof. Let Y^{n_1} and Y^{n_2} denote random graphs distributed with n_1 and n_2 nodes ($n_1 < n_2$) obtained by the finite window approach on the Poisson random connection model on $\mathbb{R}_+$, and let $\mathbb{P}^{n_1}$ and $\mathbb{P}^{n_2}$ be their corresponding distributions. Let Z_i^j denote the random latent position of node i in Y^j for $j = n_1, n_2$. For both cases, the random variables $Z_1^j - 0$, $Z_2^j - Z_1^j$, ..., $Z_{n_1}^j - Z_{n_1-1}^j$ are iid exponential random variables, by the interval theorem for point processes (Kingman, 1993, p. 39). Thus, the $(Z_i^j)_{i=1\dots n_1}$ have identical distributions for each j. The rest follows identically as for Proposition 1. □

Proof of Theorem 3.1

Proof. Let Y^{n_1} and Y^{n_2} denote random graphs distributed with n_1 and n_2 nodes ($n_1 < n_2$) obtained from a rectangular LPM on $\mathbb{R}_+^d$. Let $\mathbb{P}^{n_1}$ and $\mathbb{P}^{n_2}$ be their corresponding distributions. Let t_i^j, denote the arrival time for the ith node in Y^j for $j = n_1, n_2$. Following, Lemma 3.3.3, both $t_i^{n_1}$ and $t_i^{n_2}$ are equally distributed. Therefore, $Z_i^{n_1}$ and $Z_i^{n_2}$ must also be equally distributed. The rest follows as in the proofs for Proposition 1. □

Proof of Proposition 6

Proof. Suppose $(s_n)_{n=1\dots\infty}$ is not constant. Then there is an $n_2 > n_1 \geq 2$ such that $s_n \neq s_{n_2}$. Let Y^{n_1} and Y^{n_2} denote random graphs with n_1 and n_2 nodes. Notice that the marginal distribution of Y_{12}^n in a graph with n nodes is given by

$$\begin{aligned} \mathbb{P}^n(Y_{12} = 1) &= \mathbb{E}\left(\mathbb{P}(Y_{12}^n = 1 | Z_1, Z_2)\right) && (3.76)\\ &= \mathbb{E}\left(s_n K\left(\rho(Z_1, Z_2)\right)\right) && (3.77)\\ &= s_n \mathbb{E}\left(K\left(\rho(Z_1, Z_2)\right)\right). && (3.78) \end{aligned}$$

Clearly, $\mathbb{P}^{n_1}(Y_{12} = 1) \neq \mathbb{P}^{n_2}(Y_{12} = 1)$ because $s_{n_1} \neq s_{n_2}$ and $Z_1, Z_2 \sim f$ independently of k. Thus the model cannot be projective. □

3.6.3 Sparsity Proofs

Proof of Proposition 2

Proof. Let n be the number of nodes in the latent position network model. Then the expected number of edges $\sum_{i=1}^n \sum_{j=1}^n Y_{ij}$ is given by

$$\begin{aligned} \mathbb{E}\left(\frac{\sum_{i=1}^n \sum_{j=1}^n Y_{ij}}{n^2}\right) &= \frac{1}{n^2}\sum_{i=1}^n \sum_{j=1}^n \mathbb{E}(\mathbb{E}(Y_{ij} | Z_i, Z_j)) && (3.79)\\ &= \frac{1}{n^2}\sum_{i=1}^n \sum_{j=1}^n \mathbb{E}K(\rho(Z_i, Z_j)) && (3.80)\\ &= \mathbb{E}K(\rho(Z_i, Z_j)) && (3.81) \end{aligned}$$

where $\mathbb{E}K(\rho(Z_i, Z_j))$ is constant due to Z_i being independent and identically distributed. Thus, as long as the network is not empty, it is dense. □

Proof of Proposition 3

Proof. A special case of Theorem 3.2 with $d = 1$ and $g(t) = t$. □

Proof of Theorem 3.2

Proof. Let π be a permutation on $(1, \ldots, n)$ chosen uniformly at random from the set of permutations on $(1, \ldots, n)$. Then,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} Y_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} Y_{\pi(i)\pi(j)}. \tag{3.82}$$

Let $\Omega = [-g(t_{n+1}), g(t_{n+1})]^d$. By Lemma 3.3.6,

$$\begin{aligned}
\mathbb{E}(Y_{\pi(i)\pi(j)}|t_{n+1}) &= \int_{z,z'\in\Omega} K(||z-z'||)\frac{1}{2^d g(t_{n+1})^d}\mathrm{d}z'\frac{1}{2^d g(t_{n+1})^d}\mathrm{d}z && (3.83)\\
&\leq \frac{1}{4^d g(t_{n+1})^{2d}}\int_{z\in\Omega} C\mathrm{d}z && (3.84)\\
&\propto \frac{1}{2^d g(t_{n+1})^d} && (3.85)
\end{aligned}$$

for some $C \in \mathbb{R}_+$ by Lemma 3.3.5. Thus,

$$\begin{aligned}
\mathbb{E}\left(\frac{g(n)^d}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} Y_{ij} \mid t_{n+1}\right) &= \mathbb{E}\left(\frac{g(n)^d}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} Y_{\pi(i)\pi(j)} \mid t_{n+1}\right) && (3.86)\\
&= g(n)^d\mathbb{E}(Y_{\pi(i)\pi(j)}|t_{n+1}) && (3.87)\\
&\propto \frac{g(n)^d}{g(t_{n+1})^d}. && (3.88)
\end{aligned}$$

We can analytically integrate over possible t_{n+1} because t_{n+1} follows Gamma($n+1,\ 1$), as given by Lemma 3.3.3.

$$\mathbb{E}\left(\frac{g(n)^d}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}Y_{ij}\right) = \mathbb{E}\left(\mathbb{E}\left(\frac{g(n)^d}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}Y_{ij} \mid t_{n+1}\right)\right) \tag{3.89}$$

$$\propto \mathbb{E}\left(\frac{g(n)^d}{g(t_{n+1})^d}\right) \tag{3.90}$$

$$= n^p \int_0^{\infty} t^{-p}\frac{1}{\Gamma(n+1)}t^n \exp(-t)\mathrm{d}t \tag{3.91}$$

$$= n^p \frac{\Gamma(n-p+1)}{\Gamma(n+1)} \tag{3.92}$$

which converges to one as n goes to infinity. □

Proof of Proposition 5

Proof. Let n be the number of nodes in the LPM. Then the expected number of edges $\sum_{i=1}^{n}\sum_{j=1}^{n}Y_{ij}$ is given by

$$\mathbb{E}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}Y_{ij}\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}(\mathbb{E}(Y_{ij}|Z_i, Z_j)) \tag{3.93}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}K_n(\rho(Z_i, Z_j)) \tag{3.94}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n}s_n\mathbb{E}K(\rho(Z_i, Z_j)) \tag{3.95}$$

$$= n^2 s_n \mathbb{E}K(\rho(Z_i, Z_j)) \tag{3.96}$$

where $\mathbb{E}K(\rho(Z_i, Z_j))$ is constant due to Z_i being independent and identically distributed. □

3.6.4 Learnability Proofs

Proof of Lemma 3.2.1

Much of the argument provided here can be viewed specialization of the results established in Davenport et al. (2014, Theorem 6). For clarity, we include the entirety of the argument, illustrating our non-standard choices for many of the components, as well as some small differences such as using a restricted maximum likelihood estimator.

The notation for our proofs is simplified by working with the following standardized version of the likelihood

$$\bar{L}(z : Y^n) = L(z : Y^n) - L(z = \mathbf{0} : Y^n) \tag{3.97}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} Y_{ij}^n \log\left(\frac{K(\delta_{ij}^z)}{K(0)}\right) + (1 - Y_{ij}^n) \log\left(\frac{1 - K(\delta_{ij}^z)}{1 - K(0)}\right) \tag{3.98}$$

where $\delta_{ij}^z = ||z_i - z_j||$. Note that the standardized likelihood and non-standardized version of the likelihood are maximized by the same value of z for a given Y^n. Going forward, we use the shorthand $\bar{L}(z)$; Y^n is implied.

In order to establish concentration of $||P^{\hat{z}(Y^n)} - P^z||_F^2$, we first establish concentration of related quantities. Specifically, Lemma 3.3.13 establishes concentration of $\mathrm{KL}(P^{\hat{z}(Y^n)}, P^z)$. Here, $\mathrm{KL}(P, Q)$ denotes the Kullback-Leibler divergence (Cover and Thomas, 2006) between two link probability matrices P and Q. It is a non-negative and given by

$$\mathrm{KL}(P, Q) = \sum_{i=1}^{n} \sum_{j=1}^{n} \log\left(\frac{P_{ij}}{Q_{ij}}\right) + (1 - P_{ij}) \log\left(\frac{1 - P_{ij}}{1 - Q_{ij}}\right). \tag{3.99}$$

Lemma 3.3.13. *Consider a sequence of adjacency matrices Y^n generated by a LPM meeting the regularity criteria provided in §3.4.3. Further assume that the true latent positions are within*

$$\Omega = \left\{X \in \mathbb{R}^{n \times d} : ||X_i|| \leq G(n)\right\} \tag{3.100}$$

where X^i denotes the ith row of X. Let $P^{\hat{Z}(Y^n)}$ denote the estimated link probability matrix obtained via $\hat{Z}(Y^n)$ from (3.12). Then,

$$\mathbb{P}\left(KL(P^{\hat{z}(Y^n)}, P^z) \geq 16e\alpha_n^K G(n)^2 n^{1.5}(d+2)\right) \leq \frac{C}{n^2} \tag{3.101}$$

for some $C > 0$.

Proof. Note that for any $z_0 \in \Omega$, we have

$$\bar{L}(z_0) - \bar{L}(z) = \mathbb{E}(\bar{L}(z_0) - \bar{L}(z)) + \bar{L}(z_0) - \mathbb{E}(\bar{L}(z_0)) - (\bar{L}(z) - \mathbb{E}(\bar{L}(z))) \tag{3.102}$$

$$\leq \mathbb{E}(\bar{L}(z_0) - \bar{L}(z)) + |\bar{L}(z_0) - \mathbb{E}(\bar{L}(z_0))| + |(\bar{L}(z) - \mathbb{E}(\bar{L}(z)))| \tag{3.103}$$

$$\leq \mathbb{E}(\bar{L}(z_0) - \bar{L}(z)) + 2\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))| \tag{3.104}$$

$$\leq -\mathrm{KL}(P^{z_0}, P^z) + 2\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|. \tag{3.105}$$

Let $z_0 = \hat{Z}(Y^n)$ denote the maximum likelihood estimator given in (3.12). Then, because $\bar{L}(z_0) - \bar{L}(z) \geq 0$,

$$\mathrm{KL}(P^{\hat{z}(Y^n)}, P^z) \leq 2 \sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|. \tag{3.106}$$

So we can upper bound $\mathrm{KL}(P^{\hat{z}(Y_n)}, P^z)$ by bounding

$$\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|.$$

Let h be an arbitrary positive integer (we will later let it be $2\log(n)$). Applying the Markov inequality for $\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|^h$ yields:

$$\mathbb{P}\left(\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|^h > c(n)^h\right) \leq \frac{\mathbb{E}(\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|^h)}{c(n)^h} \tag{3.107}$$

for a positive function $c : \mathbb{N} \to \mathbb{R}_+$. To bound the expectation, we use a symmetrization argument (provided as Lemma 3.3.10 in §3.6.1) followed by a contraction argument (stated as Corollary 4 in §3.6.1).

$$\mathbb{E}\left(\sup_{x \in \Omega} |(\bar{L}(x) - \mathbb{E}(\bar{L}(x)))|^h\right) \tag{3.108}$$

$$\leq 2^h \mathbb{E}\left(\sup_{x \in \Omega} \left|\sum_{j=1}^{n} \sum_{i=1}^{n} R_{ij} \left(Y_{ij}^n \log\left(\frac{K(\delta_{ij}^z)}{K(0)}\right) + (1 - Y_{ij}^n) \log\left(\frac{1 - K(\delta_{ij}^z)}{1 - K(0)}\right)\right)\right|^h\right) \tag{3.109}$$

$$\text{by Lemma 3.3.10} \tag{3.110}$$

$$\leq (4\alpha_n^K)^h \mathbb{E}\left(\sup_{x \in \Omega} \left(\left|\sum_{j=1}^{n} \sum_{i=1}^{n} R_{ij} ||x^i - x^j||^2\right|^h\right)\right) \text{ by Corollary 4.} \tag{3.111}$$

$$= (4\alpha_n^K)^h \mathbb{E}\left(\sup_{x \in \Omega} |\langle R, D^x \rangle|^h\right) \tag{3.112}$$

where $R = (R_{ij})$ is a matrix of independent Rademacher random variables, D^x denotes the matrix of squared distances implied by x, and α_n^K is defined as in (3.14).

Let $||\cdot||_o$ denote the operator norm and $||\cdot||_*$ denote the nuclear norm. To bound $\mathbb{E}\left(\sup_{x\in\Omega}\langle R, D^x\rangle^h\right)$, we make use of the fact that $|\langle A, B\rangle| \leq ||A||_o||B||_*$. Then,

$$\mathbb{E}\left(\sup_{x\in\Omega}|\langle R, D^x\rangle|^h\right) \leq \mathbb{E}\left(\sup_{\Omega}||R||_o^h||D^x||_*^h\right) \tag{3.113}$$

$$= \mathbb{E}(||R||_o^h)\sup_{x\in\Omega}||D^x||_*^h \tag{3.114}$$

$$\leq Cn^{h/2}\sup_{x\in\Omega}||D^x||_*^h \tag{3.115}$$

where $\mathbb{E}(||R||_o^h)$ was bounded using Seginer (2000, Theorem 1.1) and $C > 0$ is a constant provided that $h \leq 2\log(n)$.

Recall that the rank of a squared distance matrix D^x is at most $d+2$ where d is the dimension of the positions x. Moreover, each of the eigenvalues of D^x must be upper bounded by the product of maximum distance in D^x and n (where n is the number of points). For $x \in \Omega$, the maximum entry in D^x is at most $4G(n)^2$. Thus, $\sup_{x\in\Omega}||D^x||_* \leq (d+2)4nG(n)^2$. Therefore,

$$\mathbb{E}\left(\sup_{x\in\Omega}|\langle E, D^x\rangle|^h\right) \leq Cn^{3h/2}(2G(n))^{2h}(d+2)^h. \tag{3.116}$$

Combining the above results yields

$$\mathbb{P}\Big(\mathrm{KL}(P^{\hat{z}(Y_n)}, P^z) \geq c(n)\Big) \leq \frac{2^{4h}Cn^{3h/2}(\alpha_n^K)^h G(n)^{2h}(d+2)^h}{c(n)^h}. \tag{3.117}$$

Let $c(n) = 2^4C_0\alpha_n^K G(n)^2(d+2)n^{3/2}$ for some constant C_0. Then, by letting $h = 2\log(n)$, we get

$$\mathbb{P}\Big(\mathrm{KL}(P^{\hat{z}(Y_n)}, P^z) \geq c(n)\Big) \leq CC_0^{-h} \tag{3.118}$$

$$= CC_0^{-2\log(n)} \tag{3.119}$$

$$= \frac{C}{n^{2\log(C_0)}} \tag{3.120}$$

The result follows from letting $C_0 = e$. □

We can now leverage Lemma 3.3.13 into Corollary 5, a concentration bound on the squared Hellinger distance $d_H^2(P^{\hat{z}(Y^n)}, P^z)$. Here, $d_H^2(P, Q)$ denotes the squared Hellinger distance between two link probability matrices P and Q given by

$$d_H^2(P, Q) = \sum_{i=1}^n\sum_{j=1}^n(\sqrt{P_{ij}} - \sqrt{Q_{ij}})^2 + (\sqrt{1-P_{ij}} - \sqrt{1-Q_{ij}})^2. \tag{3.121}$$

Corollary 5. *Consider a sequence adjacency matrices Y^n generated by a LPM meeting the criteria provided in Section 3.4.3. Further assume that the true latent positions are within*

$$\Omega = \left\{ X \in \mathbb{R}^{n \times d} : ||X_i|| \leq G(n) \right\}. \tag{3.122}$$

Let $P^{\hat{Z}(Y^n)}$ denote the estimated link probability matrix obtained via $\hat{Z}(Y^n)$ from (3.12). Then,

$$\mathbb{P}\left(d_H^2(P^{\hat{z}(Y^n)}, P^z) \geq 16e\alpha_n^K G(n)^2 n^{1.5}(d+2) \right) \leq \frac{C}{n^2} \tag{3.123}$$

Proof. (3.123) Follows from Lemma 3.3.13 and the fact that Kullback-Leibler divergence upper bounds the squared Hellinger distance (Gibbs and Su, 2002). □

Finally, the Frobenius norm $||P - Q||_F^2$ between P and Q is upper bounded by the squared Hellinger distance.

We can thus proceed with our proof of Lemma 3.2.1.

Proof. The result follows from Corollary 5 because the squared Hellinger distance between two link probability matrices upper bounds the squared Frobenius norm between them. This follows from the fact that $u \leq \sqrt{u}$ for all $u \in [0, 1]$. □

Note that, rather than immediately upper bounding $||P - Q||_F^2$ by $\mathrm{KL}(P, Q)$, we introduce $d_H^2(P, Q)$ in Corollary 5 due to its utility in proving Lemma 3.2.2.

Proof of Lemma 3.2.2

Proof. Let $\hat{d}_{ij}$ and d_{ij} denote the (i, j)th entry of $D^{\hat{Z}(Y^n)}$ and D^Z respectively. Then, $d_H^2(P^{\hat{Z}(Y^n)}, P^Z) =$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(K(\sqrt{d_{ij}})^{1/2} - K(\sqrt{\hat{d}_{ij}})^{1/2} \right)^2 + \left((1 - K(\sqrt{d_{ij}}))^{1/2} - (1 - K(\sqrt{\hat{d}_{ij}}))^{1/2} \right)^2 \tag{3.124}$$

can be lower-bounded as follows. Let $K'(t)$ denote the derivative of K with respect to t. By Taylor expansion of $\sqrt{K(\sqrt{t})}$ around $t = d_{ij}$,

$$\sqrt{K(\sqrt{\hat{d}_{ij}})} = \sqrt{K(\sqrt{d_{ij}})} + \frac{K'(\sqrt{\theta_1})(\hat{d}_{ij} - d_{ij})}{4\sqrt{\theta_1}\sqrt{K(\sqrt{\theta_1})}} \text{ for some } \theta_1 \in [0, 4G(n)^2] \tag{3.125}$$

$$= \sqrt{K(\sqrt{d_{ij}})} + \frac{K'(v)(\hat{d}_{ij} - d_{ij})}{4v\sqrt{K(v)}} \text{ for some } v \in [0, 2G(n)] \tag{3.126}$$

$$\sqrt{1 - K(\sqrt{\hat{d}_{ij}})} = \sqrt{1 - K(\sqrt{d_{ij}})} - \frac{K'(\sqrt{\theta_2})(\hat{d}_{ij} - d_{ij})}{4\sqrt{\theta_2}\sqrt{1 - K(\sqrt{\theta_2})}} \text{ for some } \theta_2 \in [0, 4G(n)^2] \tag{3.127}$$

$$= \sqrt{1 - K(\sqrt{d_{ij}})} - \frac{K'(v)(\hat{d}_{ij} - d_{ij})}{4v\sqrt{1 - K(v)}} \text{ for some } v \in [0, 2G(n)]. \tag{3.128}$$

Combining these results with (3.124) yields

$$d_H^2(P^{\hat{Z}(Y^n)}, P^Z) \geq \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{8} \inf_{\theta \in [0, 2G(n)]} \frac{K'(\theta)^2}{\theta^2} \frac{(\hat{d}_{ij} - d_{ij})^2}{\left(K(\theta)^{-1/2} + (1 - K(\theta))^{-1/2}\right)^{-2}} \tag{3.129}$$

$$\geq \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{8} \inf_{\theta \in [0, 2G(n)]} \frac{K'(\theta)^2}{\theta^2 K(\theta)(1 - K(\theta))} (\hat{d}_{ij} - d_{ij})^2 \tag{3.130}$$

$$\geq \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{8\beta_n^K} (\hat{d}_{ij} - d_{ij})^2 \tag{3.131}$$

$$= \frac{1}{8\beta_n^K} ||D^{\hat{Z}(Y^n)} - D^Z||_F^2 \tag{3.132}$$

where β_n^K is defined as in (3.15). Combining this inequality with Corollary 5 yields

$$\mathbb{P}\left(d_H^2(P_{\hat{z}}, P_z) \geq 16e\alpha_n^K G(n)^2 n^{1.5}(d+2)\right) \leq \frac{C}{n^2} \tag{3.133}$$

$$\Rightarrow \mathbb{P}\left(\frac{1}{8\beta_n^K} ||D^{\hat{Z}(Y^n)} - D^Z||_F^2 \geq 16e\alpha_n^K G(n)^2 n^{1.5}(d+2)\right) \leq \frac{C}{n^2} \tag{3.134}$$

$$\Rightarrow \mathbb{P}\left(||D^{\hat{Z}(Y^n)} - D^Z||_F^2 \geq 128e\beta_n^K \alpha_n^K G(n)^2 n^{1.5}(d+2)\right) \leq \frac{C}{n^2}. \tag{3.135}$$

□

Proof of Lemma 3.2.3

Before proceeding with the details of our proof of Lemma 3.2.3, it is useful to first introduce some notation and summarize our general strategy. Our proof involves translating our concentration inequality for the latent squared distances (provided as Lemma 3.2.2) to an analogous one for the latent positions. To do this, we combine classical multidimensional scaling (Borg and Groenen, 2005, Chapter 12) with the Davis-Kahan theorem (Lemma 3.3.12).

Classical multi-dimensional scaling recovers a set of positions $Z \in \mathbb{R}^{n \times d}$ corresponding to a squared distance matrix $D \in \mathbb{R}^{n \times n}$. It does so by eigendecomposing the double centered distance matrix $-0.5\mathcal{C}_n D \mathcal{C}_n$ with $\mathcal{C}_n$ defined in (3.18). Here,

$$Z = V\Lambda^{1/2} \tag{3.136}$$

is used to denote the recovered positions with $\Lambda \in \mathbb{R}^{d \times d}$ denoting a diagonal matrix consisting of the d nonzero eigenvalues of $-0.5\mathcal{C}_n D \mathcal{C}_n$ and $V \in \mathbb{R}^{n \times d}$ denoting a matrix with columns comprised of the corresponding eigenvectors. This technique is guaranteed to recover Z exactly (up to translations, rotations and reflections).

Multidimensional scaling of both D^Z and $D^{\hat{Z}(Y^n)}$ recovers the versions of the true latent positions and maximum likelihood estimates

$$Z = V\Lambda^{1/2} \tag{3.137}$$

$$\hat{Z}(Y^n) = \hat{V}\hat{\Lambda}^{1/2}. \tag{3.138}$$

We resolve the identifiability issues (stemming from translations, rotations, and reflections) of these values in our result by minimizing over $O \in \mathcal{O}_p$ (rotations and reflections) and $Q \in \mathcal{Q}_{nd}$ (translations). We avoid explicitly minimizing over translations on our proof. The versions of Z and $\hat{Z}(Y^n)$ obtained from multi-dimensional scaling end up being sufficient and

$$\inf_{\substack{O \in \mathcal{O}_d \\ Q \in \mathcal{Q}_{nd}}} ||\hat{Z}(Y^n)O - Z - Q||_F^2 \leq \inf_{O \in \mathcal{O}_d} ||\hat{V}\hat{\Lambda}^{1/2}O - V\Lambda^{1/2}||_F^2. \tag{3.139}$$

In addition to using the Davis-Kahan theorem to establish concentration of the eigenvectors $\hat{V}$, we apply Weyl's inequality (Horn and Johnson, 1990) to guarantee concentration of the eigenvalues $\hat{\Lambda}$ to Λ.

A few properties of orthonormal matrices are useful in our proof. By definition, the columns of $\hat{V}$ and V have norm 1. Multiplying a matrix by does not modify its Frobenius norm. Multiplying a matrix by O, V, or $\hat{V}$ does not increase a matrix's Frobenius norm due to the columns being orthonormal.

We can now proceed with the formal proof. Let $Z = V\Lambda^{1/2}$ and $\hat{Z}(Y^n) = \hat{V}\hat{\Lambda}^{1/2}$ be obtained from multidimensional scaling on $D^{\hat{Z}(Y^n)}$ and D^Z. Recall that the diagonal entries in Λ are arranged such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$ with the same decreasing structure for $\hat{\lambda}$ (but allowing for $\hat{\lambda}_d = 0$). Let $O \in \mathcal{O}_p$ denote a generic orthogonal matrix. Then,

$$
\begin{aligned}
&||\hat{Z}O - Z||_F \\
&= ||\hat{V}\hat{\Lambda}^{1/2}O - V\Lambda^{1/2}||_F \\
&\le ||\hat{V}\hat{\Lambda}^{1/2}O - \hat{V}\Lambda^{1/2}O||_F + ||\hat{V}\Lambda^{1/2}O - V\Lambda^{1/2}||_F \\
&= ||\hat{V}(\hat{\Lambda}^{1/2} - \Lambda^{1/2})O||_F + ||\hat{V}\Lambda^{1/2}O - V\Lambda^{1/2}||_F \\
&= ||\hat{\Lambda}^{1/2} - \Lambda^{1/2}||_F + ||\hat{V}\Lambda^{1/2}O - V\Lambda^{1/2}||_F \\
&= ||\hat{\Lambda}^{1/2} - \Lambda^{1/2}||_F + ||\hat{V}\left(\Lambda^{1/2} - \lambda_1^{1/2}I_d\right)O - V\left(\Lambda^{1/2} - \lambda_1^{1/2}I_d\right) + \lambda_1^{1/2}\hat{V}O - \lambda_1^{1/2}V||_F \\
&\le ||\hat{\Lambda}^{1/2} - \Lambda^{1/2}||_F + ||\hat{V}\left(\Lambda^{1/2} - \lambda_1^{1/2}I_d\right)O||_F + ||V\left(\Lambda^{1/2} - \lambda_1^{1/2}I_d\right)||_F + ||\lambda_1^{1/2}\hat{V}O - \lambda_1^{1/2}V||_F \\
&\le ||\hat{\Lambda}^{1/2} - \Lambda^{1/2}||_F + ||\Lambda^{1/2} - \lambda_1^{1/2}I_d||_F + ||\Lambda^{1/2} - \lambda_1^{1/2}I_d||_F + \lambda_1^{1/2}||\hat{V}O - V||_F \\
&= ||\hat{\Lambda}^{1/2} - \Lambda^{1/2}||_F + 2||\Lambda^{1/2} - \lambda_1^{1/2}I_d||_F + \lambda_1^{1/2}||\hat{V}O - V||_F \\
&= \sqrt{\sum_{i=1}^{d}\left(\hat{\lambda}_i^{1/2} - \lambda_i^{1/2}\right)^2} + 2\sqrt{\sum_{i=1}^{d}\left(\lambda_i^{1/2} - \lambda_1^{1/2}\right)^2} + \lambda_1^{1/2}||\hat{V}O - V||_F \\
&= \sqrt{\sum_{i=1}^{d}\left(\frac{\hat{\lambda}_i - \lambda_i}{\lambda_i^{1/2} + \hat{\lambda}_i^{1/2}}\right)^2} + 2\sqrt{\sum_{i=1}^{d}\left(\frac{\lambda_i - \lambda_1}{\lambda_1^{1/2} + \lambda_i^{1/2}}\right)^2} + \lambda_1^{1/2}||\hat{V}O - V||_F \\
&\le \lambda_d^{-1/2}||\hat{\Lambda} - \Lambda||_F + 2\lambda_1^{-1/2}||\Lambda - \lambda_1 I_d||_F + \lambda_1^{1/2}||\hat{V}O - V||_F \\
&\le \lambda_d^{-1/2}||\hat{\Lambda} - \Lambda||_F + 2\lambda_1^{-1/2}\sqrt{d-1}|\lambda_1 - \lambda_d| + \lambda_1^{1/2}||\hat{V}O - V||_F.
\end{aligned} \tag{3.140}
$$

Therefore,

$$
\inf_{O\in\mathcal{O}_d}||\hat{Z}O - Z||_F \le \lambda_d^{-1/2}||\hat{\Lambda} - \Lambda||_F + 2\lambda_1^{-1/2}\sqrt{d-1}|\lambda_1 - \lambda_d| + \inf_{O\in\mathcal{O}_d}\lambda_1^{1/2}||\hat{V}O - V||_F. \tag{3.141}
$$

To bound (3.141), we must establish concentration for the three separate terms on the right hand side. We begin with the first term. By Weyl's inequality (Horn and Johnson, 1990),

$$
|\hat{\lambda}_i - \lambda_i| \le ||\hat{V}\hat{\Lambda}\hat{V}^T - V\Lambda V^T||_F \tag{3.142}
$$

for all $1 \le i \le d$. Furthermore, centering a matrix cannot increase its Frobenius norm. Therefore,

$$
\begin{aligned}
||\hat{V}\hat{\Lambda}\hat{V}^T - V\Lambda V^T||_F &= ||\mathcal{C}_n(D^{\hat{Z}(Y^n)} - D^Z)\mathcal{C}_n||_F &(3.143)\\
&\le ||D^{\hat{Z}(Y^n)} - D^Z||_F &(3.144)
\end{aligned}
$$

meaning that

$$\lambda_d^{-1/2}||\hat{\Lambda} - \Lambda||_F \leq \sqrt{d}\lambda_d^{-1/2}||D^{\hat{Z}(Y^n)} - D^Z||_F. \tag{3.145}$$

This allows us to use Lemma 3.2.2 to bound the first term. Furthermore, we can similarly bound the third component by employing Davis-Kahan. By Lemma 3.3.12,

$$\inf_{O \in \mathcal{O}_d} ||\hat{V}O - V||_F \leq 2^{3/2}\frac{||\hat{V}\hat{\Lambda}\hat{V}^T - V\Lambda V^T||_F}{\lambda_d} \tag{3.146}$$

$$\leq 2^{3/2}\frac{||D^{\hat{Z}(Y^n)} - D^Z||_F}{\lambda_d}. \tag{3.147}$$

Finally, we have

$$\inf_{O \in \mathcal{O}_d} ||\hat{Z}O - Z||_F \leq 2\sqrt{\frac{d-1}{\lambda_1}}|\lambda_1 - \lambda_d| + \left(\sqrt{\frac{d}{\lambda_d}} + 2^{3/2}\frac{\sqrt{\lambda_1}}{\lambda_d}\right)||D^{\hat{Z}(Y^n)} - D^Z||_F \tag{3.148}$$

implying that

$$\inf_{\substack{O \in \mathcal{O}_d \\ Q \in \mathcal{Q}_{nd}}} ||\hat{Z}(Y^n)O - Z - Q||_F^2 \leq \inf_{O \in \mathcal{O}_d} ||\hat{Z}O - Z||_F^2 \tag{3.149}$$

$$\leq 4d\frac{(\lambda_1 - \lambda_d)^2}{\lambda_1} + 4\left(\frac{d}{\lambda_d} + 8\frac{\lambda_1}{\lambda_d^2}\right)||D^{\hat{Z}(Y^n)} - D^Z||_F^2. \tag{3.150}$$

The result then follows from applying Lemma 3.2.2 to bound $||D^{\hat{Z}(Y^n)} - D^Z||_F^2$.

Proof of Theorem 3.3

Proof. We first consider the learnability of the link probabilities. Let $\delta_n = \alpha_n^K G(n) n^{1.5} e(n)^{-1}$ and suppose that $\delta_n \to 0$. Then,

$$\mathbb{P}\left(\frac{||P^{\hat{z}(Y^n)} - P^z||_F^2}{e(n)} > \delta_n\right) \leq \mathbb{P}\left(\frac{||P^{\hat{z}(Y^n)} - P^z||_F^2}{e(n)} > \delta_n e(n) \mid \sup_{1 \leq i \leq n} ||z_i|| \leq G(n)\right) \tag{3.151}$$

$$+ \mathbb{P}(\sup_{1 \leq i \leq n} ||z_i|| \geq G(n)) \tag{3.152}$$

$$\leq \frac{C}{n^2} + \mathbb{P}\left(\sup_{1 \leq i \leq n} ||z_i|| \geq G(n)\right) \tag{3.153}$$

$$\tag{3.154}$$

by Lemma 3.2.1. By the third regularity assumption, this expression converges to 0 in probability. Thus,

$$\frac{||P^{\hat{z}(Y^n)} - P^z||_F^2}{e(n)} \xrightarrow{p} 0 \tag{3.155}$$

because $\delta_n = o(1)$. The proof follows the same reasoning for squared distances and latent positions, simply swapping out Lemma 3.2.1 for Lemma 3.2.2 and Lemma 3.2.3, respectively. $\square$

Proof of Corollary 1

Proof. The results follow from Theorem 3.3, and the following observations. Because $g(n) = n^{p/d}$ and K is integrable by Lemma 3.3.5, Theorem 3.2 imples that $e(n) = n^{2-p}$. Lemma 3.3.14 implies that $\lambda_1, \lambda_d = \Theta(n^{1+2p/d})$ by (3.157) and $|\lambda_1 - \lambda_d| = o_p(n^{2p/d}\log(n))$ by (3.156). Table 3.1 shows that for $K(x) = (c+x^2)^{-a}$, $\alpha_n^K \sim \Theta(1)$ and $\beta_n^K \sim \Theta(G(n)^{2a+4})$. Recall that $G(n) = \Theta(n^{p/d})$ by Lemma 3.3.8. Thus, $\beta_n^K = \Theta(n^{\frac{p}{d}(2a+4)})$. Inserting these values into Theorem 3.3 provides the results in points (1) through (3).

Letting d grow large while simultaneously setting a to be the smaller integer larger than $d/2$ allows for learnability of all three targets for values of p that are arbitrarily close to 0.5. Thus, we can have learnability arbitrarily close to $e(n) = n^{1.5}$. $\square$

The proof above relied on the following Lemma 3.3.14 to bound the eigenvalues $\lambda_1, \dots, \lambda_d$. This lemma and its proof were inspired by the work of Sussman et al. (2014, Proposition 4.3).

Lemma 3.3.14. *Let λ_i denote the ith largest eigenvalue of $\mathcal{C}_n Z Z^T \mathcal{C}_n$, where $\mathcal{C}_n Z \in \mathbb{R}^{n \times d}$ is the centered matrix of n latent positions associated with a rectangular LPM generated with $g(n) = n^{p/d}$. Let $G(n) = 2\sqrt{d}n^{p/d}$. Then, if $p < d$ and $i \leq d$,*

$$\mathbb{P}\left(|\lambda_1 - \lambda_d| > 4d^2\delta\right) \leq 4d^2 \exp\left(\frac{-2\delta^2}{G(n)^4}\right) + 8d^2 \exp\left(\frac{-2\delta}{G(n)}\right), \tag{3.156}$$

for large enough n and

$$\frac{\lambda_i}{ng(n)^2} \xrightarrow{p} C \tag{3.157}$$

for all $i = 1, \dots d$ and some constant $C > 0$.

Proof. Recall that both $Z^T\mathcal{C}_n\mathcal{C}_n Z$ and $\mathcal{C}_n Z Z^T \mathcal{C}_n$ have the same d non-zero eigenvalues $\lambda_1, \dots, \lambda_d$. Furthermore,

$$||Z^T\mathcal{C}_n\mathcal{C}_n Z - \mathbb{E}(Z^T\mathcal{C}_n\mathcal{C}_n Z)||_F \leq ||Z^T Z - \mathbb{E}(Z^T Z)||_F + \frac{||Z^T 1_n Z - \mathbb{E}(Z^T 1_n Z)||_F}{n} \tag{3.158}$$

where $1_n \in \mathbb{R}^{n\times n}$ denotes a matrix filled with ones. Suppose t_{n+1} is known, and π is a random permutation on $1,\dots,n$. Then, each $Z^{\pi(i)}$ is uniformly distributed on $[-g(t_{n+1}), g(t_{n+1})]^d$ by Lemma 3.3.4. Thus, after randomly permuting the row indices in Z, they can be treated as independent samples from this distribution. Going forward, in a slight abuse of notation, we assume that the rows of Z have been randomly permuted, meaning each row can be treated as an iid uniform sample on $[-g(t_{n+1}), g(t_{n+1})]^d$.

Therefore, $(Z^TZ)_{ij} = \sum_{k=1}^n Z_{ki}Z_{kj}$ is the sum of n iid random variables, and each summand's absolute value is upper bounded by $ng(t_{n+1})^2$. The Hoeffding bound (Hoeffding, 1963) provides us with the following concentration result.

$$\mathbb{P}(|(Z^TZ)_{ij} - \mathbb{E}((Z^TZ)_{ij})| > \delta) \leq 2\exp\left(\frac{-2\delta^2}{g(t_{n+1})^4}\right). \tag{3.159}$$

Combining this with a union bound provides

$$\mathbb{P}(||(Z^TZ) - \mathbb{E}((Z^TZ))||_F > d^2\delta) \leq 2d^2\exp\left(\frac{-2\delta^2}{g(t_{n+1})^4}\right). \tag{3.160}$$

Furthermore, $(Z^T1_nZ)_{ij} = (\sum_{k=1}^n Z_{ki})(\sum_{k=1}^n Z_{kj})$ with both factors in this product being identically distributed. Moreover, the entries in either summand is bounded in absolute value according to $|Z_{kj}| \leq g(t_{n+1})$. Therefore, applying the Hoeffding bound to each entry in Z^TZ yields

$$\begin{aligned}\mathbb{P}(|(Z^T1_nZ)_{ij} - \mathbb{E}((Z^T1_nZ)_{ij})| > \delta) &\leq 2\mathbb{P}(|(\sum_{k=1}^n Z_{ki}) - \mathbb{E}(\sum_{k=1}^n Z_{ki}))| > \sqrt{\delta}) &(3.161)\\ &\leq 4\exp\left(\frac{-2\delta}{g(t_{n+1})}\right), &(3.162)\end{aligned}$$

achieved through a union bound on the two summands differing from their means by $\sqrt{\delta}$. Another union bound over all matrix entries results in

$$\mathbb{P}\left(||Z^T1_nZ - \mathbb{E}(Z^T1_nZ)||_F > d^2\delta\right) \leq 4d^2\exp\left(\frac{-2\delta}{g(t_{n+1})}\right). \tag{3.163}$$

Combining Equations (3.158), (3.160), and (3.163) yields

$$\mathbb{P}\left(||Z^T\mathcal{C}_n\mathcal{C}_nZ - \mathbb{E}(Z^T\mathcal{C}_n\mathcal{C}_nZ)||_F > 2d^2\delta\right) \leq 2d^2\exp\left(\frac{-2\delta^2}{g(t_{n+1})^4}\right) + 4d^2\exp\left(\frac{-2\delta}{g(t_{n+1})}\right). \tag{3.164}$$

To translate this into a result for the eigenvalues, we can apply Weyl's inequality (Horn and Johnson, 1990) to obtain

$$\mathbb{P}\left(|\lambda_i(Z^T\mathcal{C}_n\mathcal{C}_nZ) - \lambda_i(\mathbb{E}(Z^T\mathcal{C}_n\mathcal{C}_nZ))| > 2d^2\delta\right) \leq 2d^2\exp\left(\frac{-2\delta^2}{g(t_{n+1})^4}\right) + 4d^2\exp\left(\frac{-2\delta}{g(t_{n+1})}\right) \tag{3.165}$$

for $1 \leq i \leq d$. We can analytically determine the values of λ_i by noting that

$$\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)_{ii} = \frac{(n-1)g(t_{n+1})^2}{12} \tag{3.166}$$

$$\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)_{i \neq j} = 0, \tag{3.167}$$

which indicates that

$$\lambda_i(\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)) = \frac{(n-1)g(t_{n+1})^2}{12} \tag{3.168}$$

for $i \leq d$, 0 otherwise. Substituting these values into (3.169) we obtain

$$\mathbb{P}\left(\left|\lambda_i(Z^T \mathcal{C}_n \mathcal{C}_n Z) - \frac{(n-1)g(t_{n+1})^2}{12}\right| > 2d^2\delta\right) \leq 2d^2 \exp\left(\frac{-2\delta^2}{g(t_{n+1})^4}\right) + 4d^2 \exp\left(\frac{-2\delta}{g(t_{n+1})}\right) \tag{3.169}$$

for $1 \leq i \leq d$.

We can use this result to bound $|\lambda_1 - \lambda_d|$.

$$\begin{aligned}
&\mathbb{P}\left(|\lambda_1 - \lambda_d| > 4d^2\delta\right) && (3.170)\\
&\leq \mathbb{P}\left(\left|\lambda_1 - \frac{(n-1)g(t_{n+1})^2}{12}\right| + \left|\lambda_d - \frac{(n-1)g(t_{n+1})^2}{12}\right| > 4d^2\delta\right) && (3.171)\\
&\leq \mathbb{P}\left(\left|\lambda_1 - \frac{(n-1)g(t_{n+1})^2}{12}\right| > 2d^2\delta\right) + \mathbb{P}\left(\left|\lambda_d - \frac{(n-1)g(t_{n+1})^2}{12}\right| > 2d^2\delta\right) && (3.172)\\
&\leq 4d^2 \exp\left(\frac{-2\delta^2}{g(t_{n+1})^4}\right) + 8d^2 \exp\left(\frac{-2\delta}{g(t_{n+1})}\right) && (3.173)
\end{aligned}$$

by (3.169). Furthermore, because t_{n+1} follows Gamma$(n+1, 1)$ (Lemma 3.3.3), $g(t_{n+1}) < g(n + \sqrt{n})$ with probability tending to 1 (See proof of Lemma 3.3.7) allowing us to swap $G(n)$ in for $g(t_{n+1})$ to obtain the bound provided in (3.156). For $p \in (0, 1)$, substituting $\delta = ng(t_n)^{3/2}$ into (3.169) yields the result in (3.157). For $p = 0$, $\delta = \sqrt{n}$ does. □

Proof of Corollary 2

Proof. Note that this LPM is regular with $G(n) = \sqrt{2\sigma^2(1+c)\log(n)}$ for any $c > 0$ by Lemma 3.3.9, and $e(n) = n^2$. Furthermore, the d leading eigenvalues $\lambda_1, \dots \lambda_d$ of $\mathcal{C}_n Z Z^T \mathcal{C}_n$ are $\Theta_p(n\sigma^2)$ and $|\lambda_1 - \lambda_d| = \Theta(1)$ by Lemma 3.3.15. Consulting Table 3.1, we see that for both link functions $\alpha_n^K = \Theta(1)$ and $\beta_n^K = \Theta(e^{G(n)^2}) = \Theta(n^{2\sigma^2(1+c)})$. Thus, applying Theorem 3.3 indicates that we have learnable latent positions and distances provided that $2\sigma^2(1+c) < 1/2$. □

As for Corollary 1, we needed a lemma (in this case, Lemma 3.3.15) to control behaviour of the eigenvalues λ_1 and λ_d of $\mathcal{C}_n Z Z^T \mathcal{C}_n$.

Lemma 3.3.15. *Let λ_i denote the ith largest eigenvalue of $\mathcal{C}_n Z Z^T \mathcal{C}_n$, where $Z \in \mathbb{R}^{n \times d}$ has independent Gaussian entries with variance σ^2. Then,*

$$\frac{\lambda_i}{n\sigma^2} \xrightarrow{p} 1, \text{ and } |\lambda_i - \lambda_d| = \Theta(1) \tag{3.174}$$

for $i \leq d$.

Proof. The proof proceeds very similarly as for Lemma 3.3.14. Recall that both $Z^T \mathcal{C}_n \mathcal{C}_n Z$ and $\mathcal{C}_n Z Z^T \mathcal{C}_n$ have the same d non-zero eigenvalues $\lambda_1, \ldots, \lambda_d$. Furthermore,

$$||Z^T \mathcal{C}_n \mathcal{C}_n Z - \mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)||_F \leq ||Z^T Z - \mathbb{E}(Z^T Z)||_F + \frac{||Z^T 1_n Z - \mathbb{E}(Z^T 1_n Z)||_F}{n} \tag{3.175}$$

where $1_n \in \mathbb{R}^{n \times n}$ denotes a matrix filled with ones. Note that

$$||Z^T Z - \mathbb{E}(Z^T Z)||_F \leq \sum_{i=1}^{d} \sum_{j=1}^{d} |Z^T Z_{ij} - \mathbb{E}(Z^T Z)_{ij}| \tag{3.176}$$

Applying Ravikumar et al. (2011, Lemma 1) and the union bound yields

$$\mathbb{P}\left(||Z^T Z - \mathbb{E}(Z^T Z)||_F > d^2 \delta\right) \leq 4d^2 \exp\left(\frac{-n\delta^2}{3200\sigma^2}\right). \tag{3.177}$$

Furthermore, note that $(Z^T 1_n Z)_{ij}/n = (\sum_{k=1}^{n} Z_{ki}/\sqrt{n})(\sum_{k=1}^{n} Z_{kj}/\sqrt{n})$ can be viewed as the product of two Gaussian distributed random variables with variance σ^2. This means that $(Z^T 1_n Z)/n$ can be viewed as uu^T where u is a d-dimensional Gaussian vector with variance σ^2. Again applying Ravikumar et al. (2011, Lemma 1) and the union bound yields

$$\mathbb{P}(||(Z^T 1_n Z)_{ij} - \mathbb{E}((Z^T 1_n Z)_{ij})||_F > d^2 \delta) \leq 4d^2 \exp\left(\frac{-n\delta^2}{3200\sigma^2}\right). \tag{3.178}$$

Combing Equations (3.175), (3.177), and (3.178) yields

$$\mathbb{P}\left(||Z^T \mathcal{C}_n \mathcal{C}_n Z - \mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)||_F > 2d^2 \delta\right) \leq 8d^2 \exp\left(\frac{-n\delta^2}{3200\sigma^2}\right). \tag{3.179}$$

To translate this into a result for the eigenvalues, we can apply Weyl's inequality (Horn and Johnson, 1990). This results in

$$\mathbb{P}(|\lambda_i(Z^T \mathcal{C}_n \mathcal{C}_n Z) - \lambda_i(\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z))| > 2d^2 \delta) \leq 8d^2 \exp\left(\frac{-n\delta^2}{3200\sigma^2}\right) \tag{3.180}$$

for $1 \leq i \leq d$. We can analytically determine the values of λ_i by noting that

$$\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)_{ii} = (n-1)\sigma^2 \tag{3.181}$$

$$\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z)_{i \neq j} = 0, \tag{3.182}$$

which indicates that

$$\lambda_i(\mathbb{E}(Z^T \mathcal{C}_n \mathcal{C}_n Z) = (n-1)\sigma^2 \tag{3.183}$$

for $i \leq d$, 0 otherwise. Moreover, using the triangle inequality, a union bound, and (3.180), we see that

$$\mathbb{P}(|\lambda_1(Z^T \mathcal{C}_n \mathcal{C}_n Z) - \lambda_d(Z^T \mathcal{C}_n \mathcal{C}_n Z)| > 2d^2\delta) \tag{3.184}$$

$$\leq \mathbb{P}(|\lambda_1(Z^T \mathcal{C}_n \mathcal{C}_n Z) - (n-1)\sigma^2| > 2d^2\delta) + \mathbb{P}(|\lambda_d(Z^T \mathcal{C}_n \mathcal{C}_n Z) - (n-1)\sigma^2| > 2d^2\delta) \tag{3.185}$$

$$\leq 16d^2 \exp\left(\frac{-n\delta^2}{3200\sigma^2}\right). \tag{3.186}$$

The results follow from letting $\delta = 1$ for (3.180) and (3.186), respectively. □

Proof of Corollary 3

Proof. The proof set-up is almost identical to that of Corollary 3.6.4, with the sole departure being that β_n^K is also scaled by the inverse of sparsity term $s(n)^{-1} = n^p$ resulting in $\beta_n^K = \Theta(n^{2\sigma^2(1+c)+p})$ requiring that $2p < 1 - 4\sigma^2(1+c)$. By choosing a small value of σ^2, we can have p get arbitrarily close to 1/2. □

3.6.5 Towards a Negative Learnability Result

In this section, we establish conditions under which regular LPMs are not learnable.

Theorem 3.4. *Consider a regular LPM. Let n denote the number of nodes. Suppose*

$$\lim_{n \to \infty} n^2 K\left(\frac{G(n)}{1+c}\right) \to 0. \tag{3.187}$$

for some $c > 0$, then this class of LPMs do not have learnable latent positions.

Proof. Recall LeCam's theorem (Tsybakov, 2008), in the form it is used to determine minimax estimation rates:

Lemma 3.4.1. *Let $\mathcal{P}$ be a set of distributions parameterized by $\theta \in \Theta$. Let $\hat{\Theta}$ denote the class of possible estimators for $\theta \in \Theta$. For any pair $P_{\theta_1}, P_{\theta_2} \in \mathcal{P}$,*

$$\inf_{\hat{\theta} \in \hat{\Theta}} \sup_{\theta \in \Theta} \mathbb{E}_{P_\theta}(d(\hat{\theta}, \theta)) \geq \frac{\Delta}{8} \exp\left(-KL(P_{\theta_1}, P_{\theta_2})\right), \tag{3.188}$$

where $\Delta = d(\theta_1, \theta_2)$ for some distance $d(\cdot, \cdot)$, and KL denotes the Kullback-Leibler divergence (Cover and Thomas, 2006).

Let Θ be the set of possible latent positions and $\mathcal{P}$ be the distributions over graphs implied by a regular LPM with link probability function K. Without loss of generality, consider the latent space to be $S = \mathbb{R}^1$. We require that the latent positions $Z_1, \ldots, Z_n \in \mathbb{R}$ be such $|Z_i| \leq G(n)$ for some differentiable and non-decreasing function $G(n)$. Suppose $G(n) = (1 + c)g(n)$ for some non-decreasing differentiable function g, and let $c > 0$ be a small constant.

To get a decent lower bound for this setting via LeCam, we choose two candidate sets of positions $\theta_1, \theta_2 \in \Theta$ corresponding to probability models that do not differ much in KL-divergence, but have embeddings differing by a non-shrinking amount in n. To accomplish this, we exploit the fact that the larger the distance between two nodes, the smaller the change in the connection probability (and thus the KL divergence) due to a small perturbation in the distance.

Consider $\theta_1^n \in \mathbb{R}^n$ according to $\theta_1^n = (0, g(n), 0, g(n), \ldots)$. That is, every odd-indexed latent position is 0, and every even-indexed latent position is $g(n)$. Similarly, define $\theta_1^n \in \mathbb{R}^n$ according to $\theta_1^n = (0, G(n), 0, G(n), \ldots)$. Let the distance metric on Θ^2 follow from the definition of learnable latent positions. That is,

$$d(\theta_1^n, \theta_2^n) = \inf_{O \in \mathcal{O}_1, Q \in \mathcal{Q}_{n1}} \frac{||\theta_1^n T - Q - \theta_2^n||_F^2}{n} \tag{3.189}$$

$$= \frac{c}{2}. \tag{3.190}$$

Here, $\mathcal{O}_1$ and $\mathcal{Q}_{n1}$ capture all possible isometric transformations. Notice that this distance is constant in n. However,

$$KL(P_{\theta_2^n}, \theta_1^n) \leq \frac{n^2}{2} K(g(n)) \log\left(\frac{K(g(n))}{K((1+c)g(n))}\right) \tag{3.191}$$

$$\to 0 \tag{3.192}$$

as n goes to infinity due to the assumption in (3.187). Thus, by Lemma 3.4.1, this class of LPMs is not learnable.

□

Chapter 4

Faster MCMC for Gaussian Latent Position Network Models

4.1 Introduction

Network data—measurements of relationships across sets of entities—are becoming increasingly common across science and industry, largely due to technological advances in data collection and storage. Common sources of network data include social networks (Carrington et al., 2005), citation networks (Ji and Jin, 2016), gene regulatory networks (Hecker et al., 2009), disease transmission networks (Newman, 2002), neural connectomes (Chen et al., 2016), transportation networks (Xie and Levinson, 2009), and food webs (Chiu and Westveld, 2011). A broad range of statistical tools based on stochastic graphs (Goldenberg et al., 2010; Crane, 2018) are available for probabilistically modeling networks, ranging from the simple Erdős-Renyi model (Erdős and Rényi, 1960) to sophisticated latent variable models (e.g. Airoldi et al. (2008); Clauset et al. (2008); Fosdick et al. (2018); Dabbs et al. (2020)). Latent variable models can be defined to capture common network properties such as community structure, hierarchical structure, and degree heterogeneity.

In latent variable models—like most network models that model edges as random variables—the computational complexity of evaluating the likelihood is quadratic in the number of nodes. These models are thus computationally costly to fit to large networks, especially if one wishes to quantify uncertainty in a Bayesian modeling and inference framework (Gelman et al., 2013). For instance, traditional Markov chain Monte Carlo algorithms (Gamerman and Lopes, 2006) (e.g. Gibbs sampling, random walk Metropolis, or Metropolis within Gibbs) can require tens of thousands of likelihood evaluations to accurately quantify expectations and uncertainties. This computational burden is even larger when the chains are slow-mixing, which is often the case for Bayesian hierarchical models.

In this work, we develop a faster Markov chain Monte Carlo algorithm for a class of latent variable network models called the latent position network model (LPM). LPMs—originally proposed by Hoff et al. (2002)—have been applied to a variety of statistical problems, including modeling network interventions (Sweet et al., 2013), clustering entities (Handcock et al., 2007), modeling social influence (Sweet and Adhikari, 2020), controlling for causal confounders (Shalizi and McFowland III, 2016), and defining priors on unobserved graphs (Linderman et al., 2016). Each node in a LPM possesses a real-valued latent variable (its *position*), with each edge treated as an independent Bernoulli random draw depending on the participating nodes' latent positions. These probabilities are modeled as a decreasing function of the nodes' latent distance, thus promoting homophily and triadic closure (e.g. a friend of a friend is more likely to be a friend) across the network. Edge probabilities may also depend on covariates, such as whether the entities share a common observed trait.

The principal task in fitting a LPM is to infer the latent positions (and thus the latent distances between pairs of nodes), as well as the parameters of the link function (e.g. the effect of any covariates). In a Bayesian modeling and inference framework, the posterior distribution of these parameters quantifies uncertainty in the corresponding estimates. Evaluating and summarizing the posterior distribution requires intensive computation—its normalization constant is defined by an integral that lacks closed form and is thus intractable to evaluate exactly.

The standard tool for computing posterior summaries has been Markov chain Monte Carlo (MCMC) via Metropolis within Gibbs (Handcock et al., 2007; Raftery et al., 2012). This technique avoids the need to calculate the normalization constant of the posterior, and can approximate posterior expectations arbitrarily well if the chain is long enough. However, accurate inference via Metropolis within Gibbs can be computationally infeasible for large networks. This infeasibility is largely due to two phenomena: (1) The random walk step size required to obtain high acceptance rates shrinks as the number of nodes grows, resulting in slowly mixing chains with strong autocorrelation, and (2) the computational complexity of performing a full sweep of position updates is quadratic in the number of nodes, so each iteration for a large network is expensive to compute. We address these challenges in this chapter through the development of a more efficient MCMC algorithm.

We are not the first to recognize these problems, nor are we the first to propose solutions. In recent years, multiple approaches for approximating the likelihood have been proposed to successfully scale up Bayesian inference of LPMs to large networks. Raftery et al. (2012) proposed a case-control based approach, subsampling the non-edge dyads to approximate each acceptance ratio in Metropolis within Gibbs. Rastelli et al. (2018) proposed a discrete-grid approximation of the latent positions, simplifying each likelihood evaluation. Salter-Townshend and Murphy (2013) proposed the use of variational inference as an alternative to MCMC. Though each of these approaches speeds up posterior inference, the improvements come at the cost of biasing the results with the likelihood approximations. As such, these methods do not have the same asymptotic guarantees as the traditional Metropolis within Gibbs approach. Regardless of how long the chain is run, a bias persists. In this respect, our work differs previous approaches; our faster MCMC algorithm is exact.

The main tool we use to accomplish this task is Hamiltonian Monte Carlo (HMC). HMC (Duane et al., 1987; Neal, 2011; Betancourt, 2017) and its variants (Girolami and Calderhead, 2011; Hoffman and Gelman, 2014; Betancourt, 2016) are a class of MCMC algorithms that leverage Hamiltonian dynamics to construct efficient gradient-informed proposals for differentiable posterior distributions. A properly tuned HMC proposal produces large moves in a chain whilst maintaining high Metropolis-Hastings acceptance rates. As such, HMC is often much more efficient than traditional random walk-based methods, especially for high-dimensional distributions with strong correlations amongst the variables.

Over the past few years, the use of HMC algorithms for Bayesian inference has been democratized in the open source software Stan (Carpenter et al., 2017). Stan implements a specialized tuning and sampling strategy for HMC that is robust across a broad class of Bayesian models. Built-in diagnostic tools make it easy to identify and address mixing problems within the Markov chain. Nevertheless, Stan's robustness depends on making some sacrifices (e.g. all variables must be updated simultaneously, and discrete latent variables must be marginalized). Usually, this rigidity is worthwhile; the limited scope of Stan's algorithm still covers a wide range of models, and can be a relatively small price to pay for the ease of implementation and built-in mixing diagnostics. However, this is not the case for large LPMs; MCMC for large LPMs often stretches one's computational resources to their limit. We thus need all tools at our disposal (including sampling discrete random variables and block updates of variables) to optimize our inference strategy.

The specialized HMC-based sampling strategy we present in this chapter is specifically intended for Gaussian LPMs (Rastelli et al., 2016), a class of LPMs for which the link probability function decays like a half-Gaussian probability density function. This class of LPMs was originally studied because they are easy to work with analytically. We show here that the Gaussian-inspired link function also provides computational advantages—the log posterior can be split into Gaussian and non-Gaussian components, thus facilitating efficient integration of HMC via split HMC (Shahbaba et al., 2014). Moreover, we further increase the efficiency for sparse networks by developing an exact dyad subsampling scheme based on Firefly Monte Carlo (FlyMC (Maclaurin and Adams, 2015)). This scheme allows us to subsample that non-edge dyads, decreasing the complexity of the non-Gaussian component of the posterior while maintaining an exact MCMC strategy. To ensure a complete LPM fitting algorithm, we also including Markov chain updates for the parameters of the link function, and show that the FlyMC approach can simplify inferring the sparsity parameters. Our approach is compatible with inferring the effect of categorical covariates on the link, as well as incorporating prior dependence between latent positions in the network (e.g. as in longitudinal latent position models (Kim et al., 2018)).

The remainder of the chapter is organized as follows. Section establishes notation and provides the necessary background information pertaining to LPMs, Gaussian LPMs and Hamiltonian Monte Carlo.

Section 4.3 outlines the ingredients of our new computation methodology for Gaussian LPMs: split Hamiltonian Monte Carlo and firefly Monte Carlo, then combines them with updates to the link function parameters to define a new Markov chain Monte Carlo strategy. Section 4.4 describes a metric for assessing the efficiency Monte Carlo algorithms for LPMs, then presents two empirical studies to demonstrate the superiority of our algorithm. Study 1 uses synthetically-generated examples to demonstrate the superior performance of our method compared to a variety of existing approaches in the literature such as Metropolis within Gibbs, elliptical slice sampling, Stan, and the No-U-turn sampler. Study 2 demonstrates the extent to which our algorithms outperform Metropolis within Gibbs for fitting information-sharing models amongst teachers and staff in a school district. Section 4.5 contains some concluding remarks.

4.2 Preliminaries

The following notation will be used throughout the chapter. We use $\mathbb{R}$ to denote the set of real numbers, $\mathbb{R}_+$ to denote the set of non-negative real numbers, $\mathbb{N}$ to denote the set of natural numbers, and $[n]$ to denote the set $\{1, \ldots, n\}$ of natural numbers less than or equal to n. For a set S, we use S^d to denote the collection of all d-length vectors with entries from S and $S^{n \times d}$ to denote collection of possible $n \times d$ matrices with entries from S. For two sets S_1, S_2, $S_1 \times S_2$ denotes their Cartesian product.

For a vector $z \in \mathbb{R}^d$, z_i denotes its ith entry and $\|z\|$ denote its Euclidean norm. For a matrix $B \in \mathbb{R}^{n \times d}$, $B_{i\cdot}$ denotes its ith row, $B_{\cdot i}$ denotes its ith column, B_{ij} denote its (i, j)th entry, $B^T \in \mathbb{R}^{d \times n}$ denotes its transpose, $\|B\|$ denotes its Frobenius norm, and B^{-1} denotes its inverse. We use I_n to denote the $n \times n$ identity matrix.

We represent networks among n entities as undirected binary graphs on n nodes. We use $A \in \{0, 1\}^{n \times n}$ to denote the adjacency matrix of the graph, with $A_{ij} = 1$ indicating the presence of an edge between nodes i and j, and $A_{ij} = 0$ indicating its absence. Our focus is on undirected graphs, so $A_{ij} = A_{ji}$ for all dyads $(i, j) \in [n]^2$. For simplicity, we use A to refer to both a graph and its adjacency matrix interchangeably, using $[n]$ index the nodes according to the order of their rows in the adjacency matrix. We use the shorthand $E_A \subseteq [n]^2$ to denote the set of edges associated with A, and $\{(i, j) \notin E_A\}$ to denote the set of possible edges absent from E_A. The combinatorial Laplacian of A is denoted as $L^A \in \mathbb{R}^{n \times n}$. Specifically, $L^A = D^A - A$ where D^A is a diagonal matrix of the node degrees $D^A_{ii} = \sum_{j=1}^n A_{ij}$.

4.2.1 Latent Position Network Models

In a distance-based latent position network model (LPM) of (Hoff et al., 2002), each node $i \in [n]$ is modeled as having a d-dimensional latent position $z_i \in \mathbb{R}^d$ for some positive integer d (typically chosen to be 2 or 3 in practice to facilitate visualization). It is convenient (for notation and computation) to arrange these latent positions in a matrix $Z \in \mathbb{R}^{n \times d}$, where $Z_{i\cdot} = z_i$. The edges A_{ij} are modeled as being generated according to

$$\mathbb{P}(A_{ij} = 1|Z) = K(\|z_i - z_j\|, x_{ij}) \tag{4.1}$$

where $\|z_i - z_j\|$ denotes the distance between nodes i and j, x_{ij} represents any relevant edge-specific covariates for nodes i and j, and K is the *link function*—a non-increasing function from the product of $\mathbb{R}_+ \times \mathcal{X}$ to $[0, 1]$. Here, $\mathcal{X}$ denotes the range of possible values of the covariate x_{ij} for each dyad $(i, j) \in [n]^2$.

In this work, we focus on the case where each covariate x_{ij} is categorical, taking on of $C \in \mathbb{N}$ distinct values (without loss of generality, we use $\mathcal{X} = [C]$ to encode the categories of such variables). Categorical covariates are common in applied problems. For example, the school district information-sharing network (Spillane et al., 2018; Sweet and Adhikari, 2020) considered in Section 4.4.3 involves a binary covariate ($C = 2$) indicating whether or not each pair of individuals work in the same school ($x_{ij} = 1$ if individuals i and j work at the same school, and $x_{ij} = 2$ otherwise). Similarly, the classroom example of Hoff et al. (2002) involves a covariate indicating whether or not pupils are of the same sex. We use $x \in [C]^{n \times n}$ to refer to the collection of all $(x_{ij})_{i \in [n], j \in [n]}$. If no covariates are present for a model, then x simply collapses to a $n \times n$ matrix of ones. Extending the techniques we describe here to accommodate real-valued covariates is a potential avenue for future work.

In their original version of the LPM, Hoff et al. (2002) proposed modeling K as a logistic function of the latent distance and the covariate according to

$$K(\|z_i - z_j\|, x_{ij}) = (1 + \exp(\alpha + \beta_{x_{ij}} + \|z_i - z_j\|))^{-1}. \tag{4.2}$$

Here, the parameters $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}^C$ control the total number of edges and the effect of the covariates, respectively. Recently, Rastelli et al. (2016) proposed an alternative form for K inspired by the functional form of the Gaussian probability density function—aptly named the Gaussian Latent Position Model (GLPM). Their original exposition did not consider covariates, taking the form

$$K(\|z_i - z_j\|) = \tau \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right). \tag{4.3}$$

Here, the parameter $\tau \in [0, 1]$ controls the number of edges (i.e. sparsity level) in the network and $\gamma^2 > 0$ controls the decay of the link probabilities. Thus far, two advantages of GLPMs over logistic LPMs have been identified in the literature. The Gaussian-like choice of K yields closed-form expressions for various network statistics of GLPMs (such as the average degree of a node and its neighbors) making them easier to theoretically analyze than logistic LPMs (Rastelli et al., 2016). Moreover, the lighter tails of the Gaussian link function are conducive to proving consistency of the maximum likelihood estimator of the latent positions (Spencer and Shalizi, 2019).

In this chapter, we identify and explore yet another advantage of GLPMs—the Gaussian shape of K facilitates faster posterior inference techniques. Our work considers an extension of the GLPM to accommodate categorical covariates. Specifically, we consider

$$K(\|z_i - z_j\|, x_{ij}) = \tau_{x_{ij}} \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right), \tag{4.4}$$

with parameters $\tau \in [0,1]^C$ and $\gamma^2 > 0$. Here, the effect of the covariate is encoded in the vector τ, allowing for subnetworks corresponding to some levels of the covariate categories to be far sparser than others. This single covariate formulation can be extended without loss of generality to multiple discrete covariates, with or without interactions, by a suitable mapping of the joint range space of covariates into $[C]$. For notational conciseness, we occasionally omit the dependence of K on x_{ij}. Together, the Gaussian shape and factorizability of K can be exploited to speed up Bayesian inference.

4.2.2 Bayesian Inference for LPMs

Fitting a LPM to a graph A can be separated into two interdependent tasks: (1) inferring the latent positions $Z \in \mathbb{R}^{n\times d}$ of the nodes, and (2) inferring the parameters of the link function K (e.g. τ, γ^2 for the Gaussian link). Depending on the modeling objective of the problem at hand, either (1) or (2) could be the primary inferential target. For instance, Z is primary inferential target when controlling for causal confounders (Shalizi and McFowland III, 2016), but τ is the primary target when estimating the effect of a covariate on edge probabilities. Regardless, both inference tasks are typically carried out simultaneously in a Monte Carlo single algorithm; independent priors are placed on both the parameters and the latent positions, and their posterior distribution is approximated with samples draw according to Markov chain Monte Carlo.

To simplify exposition, we will present our strategies for the two inference tasks (inferring Z and inferring K) separately in Sections 4.3.1 and 4.3.3. Here, we review existing computational strategies from the literature for inferring the latent positions Z conditional on K. All discussion of inference of the parameters of K for the GLPM is deferred until Section 4.3.3.

Given the full functional form of the link function K (as well as any covariates x it depends on), Bayesian inference of the latent positions Z depends on two additional inputs: an observed network (encoded by an adjacency matrix $A \in \{0,1\}^{n\times n}$), and a prior on $Z \in \mathbb{R}^{n\times d}$. The standard prior choice for Z in the literature has been an independent isotropic d-dimensional Gaussian on each row (i.e. latent position) of Z (usually $d = 2$ is used for visualization purposes). Here, we generalize this prior to $Z_{\cdot k} \sim N(0, \Omega^{-1})$—defined independently for each $k \in [d]$. That is, the nodes' positions are independent across dimensions, but can be dependent within a dimension. Without loss of generality, any Gaussian prior exhibiting dependence of a nodes' position across dimensions can be transformed into an equivalent prior with independence across dimensions via a rotation[*].

This more general set-up for the prior allows for known structural information—such as feature-informed node clustering or temporal dependence—to be included as non-zero entries in the precision matrix $\Omega \in \mathbb{R}^{n\times n}$. Other priors, such as a mixture of Multivariate Gaussians (Handcock et al., 2007; Krivitsky et al., 2009), are beyond the scope of this work, but would involve a straightforward extension of the methods presented here.

Given the prior $Z_{\cdot k} \sim N(0, \Omega^{-1})$ for $k \in [d]$, the posterior distribution on Z is given by

$$\mathbb{P}(Z|A) \propto \prod_{(i,j)\in E_A} K(\|z_i - z_j\|) \prod_{(i,j)\notin E_A} (1 - K(\|z_i - z_j\|)) \exp\left(-\frac{1}{2}\sum_{k=1}^{d} Z_{\cdot k}^T \Omega Z_{\cdot k}\right). \tag{4.5}$$

The normalization constant for this density is a $(n \times d)$-dimensional integral that cannot be computed analytically. Instead, we must rely on approximate methods for calculating expectations with respect to the posterior. Here, we discuss two related Monte Carlo strategies already proposed in the LPM literature, and use their short-comings to motivate our new Hamiltonian Monte Carlo strategy.

In their seminal LPM paper, Hoff et al. (2002) proposed for the posterior computation of $\mathbb{P}(Z|A)$ to be carried out via Markov chain Monte Carlo (MCMC). They obtained a Markov Chain $Z^1, Z^2, \ldots, Z^T$ with stationary distribution $\mathbb{P}(Z|A)$ by repeatedly applying a random walk Metropolis update to all latent positions Z simultaneously. As is the case for most MCMC algorithms, ensuring an adequate Metropolis-Hastings acceptance rate requires that the standard deviations of these random walk updates be appropriately tuned using a series of short pilot runs. However, these joint random walk proposals are known to be inefficient when the posterior is high-dimensional (e.g. for networks with many nodes) because the random walk standard deviation required to obtain reasonable acceptance rates is simply too small to explore the space efficiently. For example, when fitting GLPMs to synthetically generated 500 node networks ($C = 1$, $\tau = 0.5$, $\gamma^2 = 0.5$), we found that random walk standard deviations below 0.01 are typically required to obtain reasonable acceptance rates.

In an effort to alleviate this slow mixing, the subsequent LPM literature (e.g. (Handcock et al., 2007; Raftery et al., 2012)) use a Metropolis within Gibbs strategy for updating the latent positions instead. In *Metropolis within Gibbs*, the latent positions $(z_i)_{i\in[n]}$ are updated one at a time in sequence according to a random walk via a symmetric kernel q centered at its current position (e.g. a scaled isotropic Gaussian or multivariate uniform). This approach is still widely-used today (e.g. Fosdick et al. (2018); Aliverti and Durante (2019); Sweet and Adhikari (2020)); it is also implemented in the popular R package `latentnet` (Krivitsky and Handcock, 2008).

[*]In this sense, the dimensions of Z behave like principal components in principal component analysis

A sweep of the Metropolis within Gibbs algorithm can be summarized as follows. For each $i \in [n]$,

1. Propose $z_i' \sim q_\delta(z_i, z_i')$.

2. Accept this proposal with probability equal to

$$\frac{\mathbb{P}(Z'|A)}{\mathbb{P}(Z|A)} = \frac{\exp\left(\sum_{k=1}^{d} Z_{\cdot k}^T \Omega Z_{\cdot k}\right)}{\exp\left(\sum_{k=1}^{d} (Z'_{\cdot k})^T \Omega Z'_{\cdot k}\right)} \prod_{j:(i,j)\in E_A} \frac{K(\|z_i' - z_j\|)}{K(\|z_i - z_j\|)} \prod_{j:(i,j)\notin E_A} \frac{1-K(\|z_i' - z_j\|)}{1-K(\|z_i - z_j\|)} \tag{4.6}$$

   Otherwise reject and keep z_i.

Above, the matrix Z' in (4.6) is constructed such that $Z_i' = z_i'$ and $Z_j' = Z_j$ for all other $i \neq j$. The notation $z_i' \sim q_\delta(z_i, z_i')$ denotes drawing z_i' from a symmetric distribution centered at z_i with $\delta > 0$ denoting a tuning parameter for the width, or *step size* of the proposal. Computing (4.6) involves only the prior for z_i and the (at most n) likelihood terms corresponding to dyads containing i—all other terms are equivalent for Z and Z'. For a fully observed network A, each full sweep updating Z thus requires $O(n^2)$ computations.

Like for random walk Metropolis, it is standard practice to tune δ using preliminary tuning runs to achieve a desired acceptance rate[†]. The required value of δ typically shrinks as n grows, meaning that chains must be run much longer to achieve mixing when fitting larger networks. For example, Figure 4.6 demonstrates the decreasing relationship between the number of nodes and the tuned Metropolis within Gibbs step size δ for the variety of different synthetic networks considered in Section 4.4.2.

For large enough n, approximating the posterior using Metropolis within Gibbs thus also becomes intractable (Raftery et al., 2012)—the step-size is too small to efficiently explore the space given the complexity of computing the Metropolis-Hastings acceptance ratios. There have been multiple recent proposals that approximate the LPM likelihood (Raftery et al., 2012; Rastelli et al., 2018) to alleviate the computational burden of the accept-reject step. But as noted in the introduction, these approximations introduce non-vanishing bias in the subsequent inference.

Our goal in this work is to avoid such bias completely by developing an MCMC algorithm that outperforms Metropolis within Gibbs without sacrificing exactness of the sampler. To do so, we rely on a Monte Carlo algorithm known as Hamiltonian Monte Carlo (HMC). Recently, HMC has gained traction in the literature for fitting LPMs within large hierarchical models (e.g. Salter-Townshend and McCormick (2017) as implemented in Stan and Linderman et al. (2016) directly). However, we are the first (to our knowledge) to both develop a HMC algorithm that is specifically tooled for inference in LPMs, as well as the first to quantify their superior performance to other algorithms in the literature.

[†]Empirically, we have found that for LPMs, a Metropolis within Gibbs acceptance rate somewhere between 20 and 30 percent gives optimal results—this is consistent with related optimal scaling theory (Roberts et al., 2001)

4.2.3 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is an auxiliary variable MCMC algorithm that uses the gradient of the log posterior to inform an efficient Markov proposal kernel. Inspired by Hamiltonian dynamics, HMC augments the posterior distribution with a "momentum" variable for each target parameter, framing the task of proposing the next state of the Markov chain as that of simulating Hamiltonian motion of an object along a high-dimensional surface.

An HMC chain consists of a sequence of snapshots of the object sliding along a high-dimensional surface (without friction). The object's momentum is randomly refreshed each time a snapshot is taken, thus resulting in a sequence of stochastic draws. By using the negative log posterior as the energy function to inform its Hamiltonian motion, HMC provides larger step sizes that nevertheless maintain high Metropolis-Hastings acceptance rates. Moreover, these ratios have closed forms due to the properties of Hamiltonian dynamics (namely reversibility and volume preservation). The algorithm is thus efficient for exploring high-dimensional posteriors.

We now provide a description of HMC, placing emphasis on the components relevant to the development of our algorithm for the LPM. For more detailed reviews of the theory and practice of MCMC, Neal (2011) or Betancourt (2017) are fantastic references.

Consider a target density $p(Z)$ that is differentiable with respect to its real-valued parameters $Z \in \mathbb{R}^d$. HMC targets an augmented[‡] version of this density $p(Z, U) = p(Z)q(U)$ where $U \in \mathbb{R}^d$ is a vector of auxiliary momentum variables—each corresponding to an entry in Z. Let $q(U)$ be a zero mean multivariate Gaussian density with covariance matrix $M \in \mathbb{R}^{d \times d}$, and let $H(Z, U) = -\log(p(Z)) - \log(q(U))$. This function $H(Z, U)$ plays the role of energy in the Hamiltonian dynamics of HMC, with the covariance M (sometimes referred to as the *Mass matrix* (Neal, 2011) or *Euclidean metric* (Betancourt, 2017)) controlling the effect of the momentum on the dynamics.

Hamiltonian motion over $H(Z, U)$ is governed by the following differential equations:

$$\frac{\mathrm{d}Z_i}{\mathrm{d}t} = \frac{\partial H(Z,U)}{\partial U_i} = (M^{-1}U)_i \tag{4.7}$$

$$\frac{\mathrm{d}U_i}{\mathrm{d}t} = \frac{-\partial H(Z,U)}{\partial Z_i} = \frac{\partial \log(p(Z))}{\partial Z_i}. \tag{4.8}$$

Here, $(M^{-1}U)_i$ denotes the ith coordinate in the vector $M^{-1}U$, and t represents the artificial "time" for which the Hamiltonian trajectory is computed. That is, the derivatives of Z and U with respect to t reflect the rate of change in these quantities along Hamiltonian trajectory. Starting at an initial state (Z^0, U^0), HMC generates a Markov chain of snapshots $(Z^j, U^j)_{j \in \mathbb{N}}$ with stationary distribution $p(Z, U)$ by iterating between simulating Hamiltonian motion for a fixed amount of time $T > 0$, then refreshing the momentum

[‡]Note that because the density $p(Z, U)$ admits $p(Z)$ as a marginal, discarding the U's from a Markov chain targeting $p(Z, U)$ yields draws from $p(Z)$.

according to its conditional distribution. The value of T is typically specified by the user to control the length of time between momentum updates.

Given (Z^j, U^j), the next draw (Z^{j+1}, U^{j+1}) is obtained using the following steps

1. Gibbs update the momentum variables via Gibbs $U^{j'} \sim \text{MVN}(0, M)$.
2. Simulate Hamiltonian motion $(Z^j, U^{j'}) \to (Z^{j''}, U^{j''})$ for T time units.
3. Accept the move $(Z^{j+1}, U^{j+1}) = (Z^{j''}, -U^{j''})$ with probability

$$\min\left(\frac{p(Z^{j''}, U^{j''})}{p(Z^{j'}, U^{j'})}, 1\right). \tag{4.9}$$

Otherwise reject the move, letting $(Z^{j+1}, U^{j+1}) = (Z^j, U^{j'})$.

The negation of $U^{j''}$ in Step 3 ensures the proposal is reversible[§].

The performance of HMC as described above depends on two parameters chosen by the user: the mass matrix M and the amount of time T between momentum updates. Before discussing strategies for choosing these parameters, we must first address how to simulate Hamiltonian motion.

If the Hamiltonian motion in Step 2 above were to be simulated exactly, the Metropolis-Hastings correction in Step 3 would be unnecessary because the ratio is exactly one (Neal, 2011). This property is guaranteed by the conservation of energy in Hamiltonian motion—Step 2 simply moves along a density contour of the augmented distribution. Unfortunately, exact simulation of the Hamiltonian motion is not possible for most posterior densities that arise in Bayesian inference—there is no known analytic way to move along the contours.

In practice, simulation of the trajectory of Hamiltonian motion is typically carried out using approximate numerical integrators of the differential equations, the most popular of which is the *leapfrog integrator* (Neal, 2011) (sometimes referred to as the Stormer-Verlet integrator: e.g. (Chao et al., 2015)). The leapfrog integrator discretizes the Hamiltonian motion via alternating linear updates of U and Z until the trajectory of length T has been simulated. The following steps are iterated $L \in \mathbb{N}$ times:

$$U \leftarrow U - \frac{\epsilon}{2}\frac{\partial H}{\partial Z}(Z, U) \tag{4.10}$$

$$Z \leftarrow Z + \epsilon M^{-1} U \tag{4.11}$$

$$U \leftarrow U - \frac{\epsilon}{2}\frac{\partial H}{\partial Z}(Z, U) \tag{4.12}$$

Together, the user-specified parameters $\epsilon > 0$ (sometimes called the step-size) and $L \in \mathbb{N}$ (the number of steps) define leapfrog trajectory of length $T = L\epsilon$. Smaller values of ϵ provide more accurate approximations

[§]In practice, the marginal distribution of Z (and not the joint distribution of Z, U) is the target of HMC, so this negation step can be omitted because it is immediately changed by subsequent Gibbs update of U (Neal, 2011).

of the Hamiltonian motion (and thus the higher the Metropolis-Hastings acceptance rate), but also require correspondingly larger values of L—and thus more computation—to simulate a given trajectory length T. It is thus important to strike a balance between the two to obtain adequately high acceptance rates for reasonably correlated draws without wasting computational resources.

Choosing the user-specified parameters M and T for HMC via leapfrog amounts to choosing three parameters: the mass matrix M, the step size ϵ and the number of steps L. The original time parameter $T = L\epsilon$ is a byproduct of the choices of ϵ and L.

Like tuning the step-size for traditional Metropolis algorithms, standard practice for choosing ϵ and L is to conduct a series of preliminary tuning runs with various levels of the parameters, converging toward values that maximize the chain's efficiency. Similarly, the matrix M can also be chosen according to these tuning runs. A theoretically motivated heuristic (Betancourt, 2017) is to set M equal to an estimator of the precision matrix of the posterior, estimated from the empirical posterior covariance in the preliminary chains. In practice, the true precision matrix may be dense (and thus expensive to compute), meaning that a diagonal approximation (Carpenter et al., 2017) or a low-rank approximation (Bales et al., 2019) of the precision may also be a suitable choice. The former is implemented in the software package Stan.

Unfortunately, efficiently tuning all three of L, ϵ, and M can itself be a computationally burdensome because the three parameters are interdependent (the optimal choice of ϵ relies particularly heavily on the choice of M). Indeed, under the standard settings, it is typical for Stan to devote more time to adapting L, ϵ and M than running the final Markov chain. When the computational problem is already straining the computational budget at hand—such as in the case of large LPMs—these tuning costs can be prohibitive. We thus seek a simpler, more easily tunable algorithm that is specialized to large LPMs.

Before proceeding, it is worth noting that strategies exist for which L, ϵ and M are not necessarily held constant through all regions of the posterior. For instance, sophisticated algorithms exist for selecting L (e.g. the No-U-Turn sampler (NUTS (Hoffman and Gelman, 2014))), or M (e.g. Riemannian HMC Girolami and Calderhead (2011)) adaptively depending on the current state of the chain. However, these algorithms can be incredibly computationally intensive, as they can require many additional density, gradient, or Hessian evaluations. This is evident when we compare Stan and NUTS to our strategy in Section 4.4.2.

In the sequel, we derive an alternative HMC integration strategy that exploits the function form of the GLPM posterior for efficient exploration without costly tuning runs for L and M. Moreover, it locally adapts M to the link function parameter γ—providing a natural locally-adaptive choice of M that does not require computation of expensive Hessian calculations in Girolami and Calderhead (2011).

4.3 New Sampling Methodology

Our new MCMC algorithm for the GLPM is composed of three novel components: a split Hamiltonian Monte Carlo (Shahbaba et al., 2014) integrator to update the latent positions (Section 4.3.1), a Firefly Monte Carlo (FlyMC (Maclaurin and Adams, 2015)) auxiliary variable scheme to sub-sample non-edge dyads (Section 4.3.2), and Gibbs sampling strategies to update the parameters τ and γ^2 of the link probability function K (Section 4.3.3). We present each of these contributions in sequence.

4.3.1 Split Hamiltonian Monte Carlo

Though it is certainly the most popular for HMC, the leapfrog integrator is just one of many options for integrating Hamiltonian dynamics (e.g. Leimkuhler and Reich (2004); Chao et al. (2015); Mannseth et al. (2016)). Here, we consider an alternative called *split Hamiltonian Monte Carlo* (Shahbaba et al., 2014). Split HMC is a variant on the leapfrog strategy that efficiently integrates Hamiltonian's equations by leveraging a Gaussian component of the posterior. It works best when the Gaussian component is a good approximation of the posterior. As an added bonus, split HMC provides a natural and effective choice for the mass matrix M.

Note that the standard leapfrog update described in Section 4.2.3 is equivalent to decomposing the energy into three terms:

$$H(Z, U) = -\frac{1}{2}\log(p(Z)) - \log(q(U)) - \frac{1}{2}\log(p(Z)). \tag{4.13}$$

then cycling through isolated updates according (4.7) and (4.8) for each of the components individually. This "split" of the energy ensures that only one of Z or U is being updated at any given time, causing each isolated operation to be straightforward. In split Hamiltonian Monte Carlo, we consider a different split of the energy function, decomposing it to exploit partial analytic solutions of Hamiltonian equations.

Hamilton's equations usually lack an analytic solution, but one of a few notable exceptions occurs when the energy is defined as the negative logarithm of a multivariate Gaussian density (Pakman and Paninski, 2014) (other exceptions include the univariate exponential and uniform distributions (Bloem-Reddy and Cunningham, 2016)). For the Gaussian case, the motion and momentum updates can be simulated exactly along an ellipse (i.e. a contour of the Multivariate Gaussian distribution). Alone, this fact would have limited utility for Bayesian computation; exact algorithms for inference involving Gaussian posteriors are readily available. As part of an energy splitting strategy, however, these analytic solutions can be remarkably useful.

The split Hamiltonian integrator (Shahbaba et al., 2014) alternates between joint position momentum updates based on the analytical solution of the Gaussian component of the posterior and updates to the momentum to correct for the remaining portion of the posterior. When the exact part is a good approximation for the entire posterior, this allows for a coarser ϵ to maintain a high acceptance rate.

To make things more concrete, we now present the decomposition of the GLPM posterior into its Gaussian and non-Gaussian components. Specifically, the likelihood of the edges, the prior density, and the momentum forming the Gaussian component, and the likelihood of non-edges forms the remainder.

Recall the LPM posterior's functional form as provided in (4.5). Treating the τ and γ^2 as known, (4.5) takes the form

$$\mathbb{P}(Z|A,\tau,\gamma^2) = \prod_{\{i,j\}\in E_A} \tau_{x_{ij}} \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \left(\Omega + \frac{1}{\gamma^2}L^A\right) Z_{\cdot\ell}\right) \prod_{(i,j)\notin E_A} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{\|z_i - z_j\|^2}{2\gamma^2}\right)\right) \tag{4.14}$$

where L^A denotes the Laplacian of A. This posterior can thus be decomposed into two components $\mathbb{P}(Z|A,\tau,\gamma^2) = \mathbb{P}_1(Z|A,\tau,\gamma^2)\mathbb{P}_0(Z|A,\tau,\gamma^2)$ where

$$\mathbb{P}_1(Z|A,\tau,\gamma^2) = \left(\prod_{\{i,j\}\in E_A} \tau_{x_{ij}}\right) \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \left(\Omega + \frac{1}{\gamma^2}L^A\right) Z_{\cdot\ell}\right) \tag{4.15}$$

corresponds to the contribution of the prior and likelihood of the observed edges and

$$\mathbb{P}_0(Z|A,\tau,\gamma^2) = \prod_{ij\notin E_A} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right)\right) \tag{4.16}$$

corresponds to the contribution to the likelihood of the non-edges.

Using the shorthand

$$\Sigma = \left(\Omega + \frac{1}{\gamma^2}L^A\right), \tag{4.17}$$

we can now split the corresponding energy as

$$H(Z,U) = \left[-\frac{1}{2}\log(\mathbb{P}_0(Z|A,\tau,\gamma^2))\right] + \left[\frac{1}{2}\sum_{\ell=1}^{d}\left(Z_{\cdot\ell}^T\Sigma Z_{\cdot\ell} + U_{\cdot\ell}^T M^{-1} U_{\cdot\ell}\right)\right] - \left[\frac{1}{2}\log(\mathbb{P}_0(Z|A,\tau,\gamma^2))\right], \tag{4.18}$$

ignoring additive constants. The center term in this split is Gaussian.

In the above, we have departed from the typical notation in our definition of the mass matrix M and momentum variables U. In standard presentations of HMC (including our presentation of HMC is Section 4.2.3), the target parameters and momentum variables are naturally represented as vectors. For a LPM, the parameters Z are more suitably represented a $n \times d$ matrix. We have thus chosen to also represent the momentum variables U as a $n \times d$ matrix. Since there are $n \times d$ momentum variables, the standard notation/definition of the mass matrix would require that $M \in \mathbb{R}^{nd \times nd}$. We have opted to instead define the full mass matrix block diagonally, using d repetitions of the same matrix $M \in \mathbb{R}^{n \times n}$. This choice facilitates the more compact representation in (4.18) without altering the validity of the algorithm.

On top of being notationally and computationally convenient, the use of an identical mass matrix across all dimensions is justified by symmetry in the target posterior—the marginal distribution of each column of Z is the same (Shortreed et al., 2006).

The above decomposition thus suggests a natural choice of M. Recall from Section 4.2.3 that the precision matrix of the posterior is an efficient choice for M. Accordingly, we suggest that Σ is a reasonable choice for M, as it should be a good approximation of the posterior precision matrix provided that $\mathbb{P}_0$ is a good approximation of the full posterior. Moreover, the choice $M = \Sigma$ is also particularly amenable to simulating the split HMC trajectories because it leads to arithmetic cancellations that simplify computation. Finally, the mass matrix M depends on the parameter γ^2—when combined with a Monte Carlo strategy for inferring γ^2 (such as the we present in Section 4.3.3), setting $M = \Sigma$ allows for the mass matrix to evolve adaptively with the state of γ^2 in the chain.

The following is a complete recipe for split HMC for LPMs, using the block diagonal mass matrix we have just defined. Note that the intermediate variable $V \in \mathbb{R}^{n \times d}$ introduced in Step 2 is a change of variable for efficiently parametrizing the contour of the multivariate Gaussian, and Step 4 inverts the change of variable to recover U. For more details on the exact simulation of HMC for Multivariate Gaussians, see Pakman and Paninski (2014).

Suppose that A denotes an observed adjacency matrix, $\tau \in [0,1]^C$ and $\gamma^2 > 0$ denote the values of the parameters of the Gaussian link function, and $\epsilon > 0$, $L \in \mathbb{N}$ and Σ (as defined in (4.17)) denote the user-specified tuning parameters for split HMC. Given (Z^j, U^j), the next split HMC draw (Z^{j+1}, U^{j+1}) is obtained via the following steps:

1. Gibbs update the momentum variables via Gibbs $U^{j'} \sim \mathrm{MVN}(0, \Sigma)$.

2. Define intermediate variables $V^{j'} = \Sigma^{-1} U^{j'}$ and $Z^{j''} = Z^j$.

3. Integrate Hamiltonian motion $(Z^j, U^{j'}) \to (Z^{j''}, U^{j''})$ for $T = L\epsilon$ time units by iterating the following updates L times:

$$V^{j'} \leftarrow V^{j'} + \frac{\epsilon}{2}\Sigma^{-1}\frac{\partial \log(\mathbb{P}_0(Z|A,\tau,\gamma^2))}{\partial Z}(Z^{j''}) \tag{4.19}$$

$$(Z^{j''}, V^{j'}) \leftarrow \Big(\sin(\epsilon)V^{j'} + \cos(\epsilon)Z^{j''}, \cos(\epsilon)V^{j'} - \sin(\epsilon)Z^{j''}\Big) \tag{4.20}$$

$$V^{j'} \leftarrow V^{j'} + \frac{\epsilon}{2}\Sigma^{-1}\frac{\partial \log(\mathbb{P}_0(Z|A,\tau,\gamma^2))}{\partial Z}(Z^{j''}) \tag{4.21}$$

4. Let $U^{j''} = \Sigma V^{j'}$.

5. Accept the move $(Z^{j+1}, U^{j+1}) = (Z^{j''}, -U^{j''})$ with probability

$$\min\left(\frac{\mathbb{P}(Z^{j''}|A,\tau,\gamma^2)}{\mathbb{P}(Z^j|A,\tau,\gamma^2)}\exp\left(\frac{1}{2}\sum_{\ell=1}^{d}\left(U'_{\cdot\ell} - U''_{\cdot\ell}\right)^T \Sigma^{-1}\left(U'_{\cdot\ell} - U''_{\cdot\ell}\right)\right), 1\right). \tag{4.22}$$

   Otherwise, the move is rejected and $(Z^{j+1}, U^{j+1}) = (Z^j, U^{j'})$.

In Step 2 above, the gradient functions return $n \times d$ matrices defined by

$$\left(\frac{\partial \log(\mathbb{P}_0(Z|A,\tau,\gamma^2))}{\partial Z}(Z)\right)_{ik} = \frac{\partial \log(\mathbb{P}_0(Z|A,\tau,\gamma^2))}{\partial Z_{ik}} \tag{4.23}$$

$$= \sum_{j:ij\notin E_A} \frac{(Z_{ik} - Z_{jk})}{\gamma^2} \frac{\tau_{x_{ij}} \exp\left(-\frac{\|z_i - z_j\|^2}{2\gamma^2}\right)}{1 - \tau_{x_{ij}} \exp\left(-\frac{\|z_i - z_j\|^2}{2\gamma^2}\right)}. \tag{4.24}$$

Computing the gradients in (4.24) and the acceptance ratio in Steps 2 and 4 represent the main computational bottlenecks of our split HMC algorithm for LPMs, as they both require an operation be performed for each non-edge. Note that large sparse networks possess many non-edges. For such networks, it is especially important that we compute the gradients as efficiently as possible because the optimal value of L may be large. Motivated by this computational bottleneck, we now develop an exact subsampling strategy to reduce the number of non-edges that must be considered for each gradient computation.

4.3.2 Firefly Sampling of Non-Edges

Recall from Section 4.3.1 that the obstacle preventing exact simulation of the Hamiltonian motion is the presence of the non-edge terms in the likelihood. Moreover, the computational bottleneck for running the proposed split HMC algorithm is the density and gradient operations that involved the non-edges. Thus, it could be beneficial to eliminate some of non-edge terms of the model likelihood at each iteration of split HMC. Here, we propose such a strategy.

Consider the following data augmentation scheme inspired by the Firefly Monte Carlo (FlyMC (Maclaurin and Adams, 2015)). For each $i, j \in [n]^2$, we define auxiliary independent binary random variables θ_{ij} such that $\mathbb{P}(\theta_{ij} = 1|\tau_{x_{ij}}) = \tau_{x_{ij}}$. Using these auxiliary variables, we can re-express the edge probabilities as

$$\mathbb{P}(A_{ij} = 1|\theta_{ij} = 1, \tau_{x_{ij}}, \gamma^2) = \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right) \tag{4.25}$$

$$\mathbb{P}(A_{ij} = 1|\theta_{ij} = 0, \tau_{x_{ij}}, \gamma^2) = 0, \tag{4.26}$$

while maintaining the same marginal likelihood. Now,

$$\mathbb{P}(\theta_{ij} = 0 | A_{ij} = 0, Z, \tau_{x_{ij}}, \gamma^2) = \frac{1 - \tau_{x_{ij}}}{1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right)}, \tag{4.27}$$

$$\mathbb{P}(\theta_{ij} = 0 | A_{ij} = 1, Z, \tau_{x_{ij}}, \gamma^2) = 0. \tag{4.28}$$

Note that for all $(i, j) \in E_A$, $\theta_{ij} = 1$ must hold. Thus,

$$\mathbb{P}(Z|A, \theta, \tau, \gamma^2) = \mathbb{P}_1(Z|A, \tau, \gamma^2) \prod_{ij:\theta_{ij}=1, A_{ij}=0} \left(1 - \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right)\right), \tag{4.29}$$

meaning that

$$\mathbb{P}_0^*(Z|A, \theta, \tau, \gamma^2) = \prod_{ij:\theta_{ij}=1, A_{ij}=0} \left(1 - \exp\left(-\frac{1}{2\gamma^2}\|z_i - z_j\|^2\right)\right) \tag{4.30}$$

can replace $\mathbb{P}_0(Z|A, \tau, \gamma^2)$ in Split HMC once the θ variables are instantiated. If many of the θ_{ij} are 0, computing $\mathbb{P}_0(Z|A, \tau, \gamma^2)$—and its gradients—is far cheaper than computing the marginal $\mathbb{P}_0(Z|A, \tau, \gamma^2)$ analogs. Thus, combining the above with split HMC can represent a major computational improvement, provided that we can instantiate and update the θ_{ij} values efficiently.

To do so, we propose a Metropolis-Hastings step using proposal $q(\theta_{ij} = 1) = \tau_{ij}$. Let $\mathrm{MH}(\theta_{ij} = 0 \to \theta_{ij} = 1)$ denote the Metropolis-Hastings ratio associated with a proposed move from $\theta_{ij} = 0$ to $\theta_{ij} = 1$, and let $\mathrm{MH}(\theta_{ij} = 1 \to \theta_{ij} = 0)$ denote the Metropolis-Hastings ratio associated with a proposed move from $\theta_{ij} = 1$ to $\theta_{ij} = 0$. The values of these ratios are given by

$$\mathrm{MH}(\theta_{ij} = 0 \to \theta_{ij} = 1) = \left(1 - \exp\left(-\frac{1}{2\gamma_{ij}^2}\|z_i - z_j\|^2\right)\right), \tag{4.31}$$

$$\mathrm{MH}(\theta_{ij} = 1 \to \theta_{ij} = 0) = \frac{1}{1 - \exp\left(-\frac{1}{2\gamma_{ij}^2}\|z_i - z_j\|^2\right)} > 1. \tag{4.32}$$

Thus, out of the four possible moves $0 \to 0$, $0 \to 1$, $1 \to 0$, $1 \to 1$, the accept reject step need only be performed for $0 \to 1$. Therefore, this strategy is more computationally efficient than updating each θ_{ij} according to its full conditional using a Gibbs update.

Going forward, we refer to the parameter augmentation and update strategy described above as FlyMC. The reduction in computational cost of evaluating the posterior density and gradients under FlyMC is most prevalent when most of the θ_{ij} are 0. Because the $\mathbb{P}(\theta_{ij} = 0|\tau_{x_{ij}}) = 1 - \tau_{x_{ij}}$, the computational gains from FlyMC are thus largest when $\tau_{x_{ij}}$ is small. On the other hand, when $\tau_{x_{ij}}$ are relatively large (close to 1), most values of the θ_{ij} will be one, meaning the computational improvements in evaluating the gradient may not justify the computational expense of instantiating and updating the θ variables. In the extreme case of $\tau_{x_{ij}} = 1$, no subsampling will occur at all, so FlyMC should not be included.

For sparse networks, however, $\tau_{x_{ij}}$ may be very small for some values of x_{ij}, leading to substantial computational gains. In addition to providing a computational speed-up, the new FlyMC posterior facilitates the inference of τ via Gibbs steps. We now discuss MCMC updates for τ and γ^2 in Section 4.3.3.

4.3.3 Bayesian Inference of the Parameters of the Link Function

Thus far, our posterior computation strategy for Z has held τ and γ^2—the parameters of the link function—at fixed values. In most applications, τ and γ^2 are unknown—they need to be inferred along with Z. As we noted in Section 4.2.2, τ may even be the primary inferential target when the main scientific question involves measuring the effect of a covariate. For these reasons, it is important that our posterior computation strategy be able to compute the full joint posterior of τ, γ^2, and Z. Here, we describe efficient Gibbs updates for both τ (Section 4.3.3) and γ^2 (Section 4.3.3) that can be alternated with our split HMC + FlyMC strategy (Sections 4.3.1 and 4.3.2) for updating Z to define a complete MCMC algorithm.

Before proceeding, it is worth making a couple of notes. The updates we describe here apply to specific families of priors on τ and γ^2: independent Beta priors being used for each entry in τ and an inverse Gamma prior γ^2. Moreover, the update for τ is only applicable in conjunction with the FlyMC strategy outlined in Section 4.3.2. If FlyMC is not used (and thus the θ are not available), we recommend a simple random walk Metropolis-Hastings update for τ instead. Finally, our update strategy for γ^2 depends on a re-parameterization of the model that shifts direct inference of γ^2 in the link function to inference of a constant multiple on the variance Ω of the latent positions Z. This facilitates a conversion to a centered parameterization (Papaspiliopoulos et al., 2007) of Z, allowing a Gibbs update where one was previously intractable. This Gibbs update of γ^2 is applicable whether or not FlyMC is used.

Updating τ given Z, θ, γ^2

Suppose that each of the C entries in the vector τ are assigned independent priors such that $\tau_c \sim \text{Beta}(\alpha_c, \beta_c)$ for $c \in [C]$, $\alpha, \beta \in \mathbb{R}_+^C$. Then, if the FlyMC strategy outlined in Section 4.3.2 is used, each τ_c can be Gibbs updated according to its conditional posterior distribution given the FlyMC variables θ and the covariates x. Indeed, this posterior distribution is actually conjugate to the Beta prior because inferring $\tau_c|\theta, x$ for $c \in [C]$ is equivalent to inferring the probability parameter of a sequence of Bernoulli trials (specifically the θ_{ij} for which $x_{ij} = c$). Thus,

$$\tau_c \mid \theta, x \sim \text{Beta}(\alpha_c + \Theta_c^1, \beta_c + \Theta_c^0) \tag{4.33}$$

for each $c \in [C]$, where $\Theta^0, \Theta^1 \in (\{0\} \cup \mathbb{N})^C$ are defined according to

$$\Theta^0_c = | \left\{ \{i,j\} \in [n]^2 : \theta_{ij} = 1 \text{ and } x_{ij} = c \right\} | \tag{4.34}$$

$$\Theta^1_c = | \left\{ \{i,j\} \in [n]^2 : \theta_{ij} = 0 \text{ and } x_{ij} = c \right\} |. \tag{4.35}$$

This update can be efficiently updated by keeping track of the values of Θ^0 and Θ^1 when performing the FlyMC updates.

Updating γ^2 given Z, τ, θ

Suppose we place an InverseGamma(a, b) prior on γ^2, where $a, b > 0$. Then, the posterior density of $\gamma^2 | A, Z, \theta, \tau, x$ is proportional to

$$p(\gamma^2 \mid A, Z, \theta, \tau, x) = \text{IG}(\gamma^2 | a, b) \exp\left(-\frac{1}{2\gamma^2} \sum_{\ell=1}^{d} Z_{\cdot\ell}^T L^A Z_{\cdot\ell} \right) \prod_{ij : \theta_{ij}=1, A_{ij}=0} \left(1 - \exp\left(-\frac{1}{2\gamma^2} \|z_i - z_j\|^2 \right) \right) \tag{4.36}$$

where $\text{IG}(\gamma^2|a,b)$ denotes the probability density function of the InverseGamma(a, b) distribution evaluated at γ^2. This density provides no closed-form Gibbs update and is expensive to evaluate so working directly with it would be cumbersome as part of an MCMC strategy.

However, we can remedy this situation via a re-parameterization. Recall that Z has a Gaussian prior defined by $Z_{\cdot\ell} \sim N(0, \Omega)$ for $\ell \in [d]$. Thus, the re-parameterized random variable $Z^* = \gamma^{-1} Z$ has a conditionally multivariate Gaussian prior defined by $Z^*_{\cdot\ell} | \gamma^2 \sim N(0, \gamma^{-1}\Omega)$ for $\ell \in [d]$. The conditional distribution of γ^2 given A, Z^*, θ, τ, x is given by

$$p(\gamma^2 \mid A, Z^*, \theta, \tau, x) = \text{IG}(\gamma^2|a,b) \exp\left(-\frac{\gamma^2}{2} \sum_{\ell=1}^{d} \left(Z^*_{\cdot\ell}\right)^T \Omega^{-1} Z^*_{\cdot\ell} \right) \tag{4.37}$$

$$= \text{IG}\left(\gamma^2 | a + \frac{nd}{2}, b + \frac{1}{2} \sum_{\ell=1}^{d} \left(Z^*_{\cdot\ell}\right)^T \Omega^{-1} Z^*_{\cdot\ell} \right). \tag{4.38}$$

Therefore, by conditioning on Z^* instead of Z, the posterior dependence of γ^2 on θ and A have been removed—it depends solely on Z^*. Moreover, the prior on γ^2 is now conjugate, and can be updated via a simple Gibbs update.

Fortunately, this re-parameterization and the corresponding Gibbs update can be easily incorporated with the updates of Z, θ, and τ described in Sections 4.3.1, Section 4.3.2, and Section 4.3.3, respectively. Doing so requires just a small re-tooling of the model/notation. Instead of viewing the Gaussian LPM as having latent positions Z with prior $Z_{\cdot\ell} \sim N(0, \Omega)$ and a two parameter link function defined in (4.4), we can re-express the Gaussian LPM with Z^* taking the role of the latent positions, γ^2 being a hyperparameter of their prior $Z^*_{\cdot\ell} \sim N(0, \gamma^{-1}\Omega)$, and the edges generated according to a one parameter link function defined by

$$K(\|z_i^* - z_j^*\|, x_{ij}) = \tau_{x_{ij}} \exp\left(-\frac{1}{2}\|z_i^* - z_j^*\|^2\right). \tag{4.39}$$

Because the updates of Z (Section 4.3.1), θ (Section 4.3.2), and τ (Section 4.3.3) treated γ^2 as fixed, they are equally applicable after this re-parametrization—we just fix the γ^2 in the link function at 1, and have Z^* play the role of Z (with a new hyperparameter γ^2 on the prior). After performing MCMC sampling this re-parametrized setting, one can recover the original parameterization of Z by simply undoing the change-of-variable transformation.

Thus, we have now defined a full MCMC strategy for posterior computation of Z, τ, γ^2 that also introduces and updates auxiliary FlyMC variables θ. Each sweep of the chain iterates through a split HMC update of Z, a Metropolis update of each θ, a Gibbs update for each entry in τ, and a Gibbs update of γ^2. Alternatively, if FlyMC is not incorporated, posterior computation of Z, τ, and γ^2 can be performed by alternating the split HMC update of Z, a Metropolis update of each entry in τ, then a Gibbs update of γ^2. For completeness, the functional form of the posterior being computed in both the split HMC and split HMC +FlyMC algorithms is outlined in Section 4.6.2.

4.4 Empirical Studies

To explore and understand the relative strengths and weaknesses of our new posterior inference algorithm for GLPMs, we conduct two empirical studies. Study 1 (Section 4.4.2) consists of a "bake-off" between various inference algorithms, comparing the efficiency of our method to that of plausible competitors from in literature. These comparisons span a variety of synthetically generated networks of different sizes and sparsity levels. Study 2 (Section 4.4.3) is a real data example, demonstrating efficiency of split HMC and split HMC + FlyMC compared to traditional Metropolis within Gibbs for modeling information-sharing among elementary school teachers and staff in a school district. It considers multiple model set-ups, including the use of use categorical covariates and longitudinal network data. Details of the computational software and hardware used to run the experiments is provided in Section 4.6.1.

Before delving into Studies 1 and 2, Section 4.4.1 motivates and describes our metric for comparing the relative efficiency of various Monte Carlo algorithms for LPMs.

4.4.1 Measuring relative efficiency of MCMC Algorithm for LPMs

There are two principal criteria by which to judge the efficiency of a Markov chain for approximating a posterior distribution.

1. How efficiently does the MCMC sequence approximate the posterior expectation of a desired function of the parameters?

2. What is the computational expense of generating this chain?

Simple metrics exist for gauging each of these criteria, which we describe below.

For criterion 1, it is well-known that that mean estimates stemming from a geometrically ergodic (Roberts et al., 1997) Markov chain are subject to a Markov chain central limit theorem (CLT) (Tierney, 1994). Analogous to the standard (independent samples) central limit theorem, for which the variance of the estimator is inversely proportional to the raw sample size, the variance in the Markov chain CLT is inversely proportional to the *Effective sample size* (Kass et al., 1998; Ripley, 2009).

Letting $\theta^1, \theta^2, \ldots, \theta^N$ denote N draws from a Markov chain and f denote an arbitrary function, the effective sample size ESS_f for estimating the expectation of $f(\theta)$ is defined as

$$\mathrm{ESS}_f(\theta^1, \ldots, \theta^N) = \frac{N}{1 + 2\sum_{t=1}^{\infty} \rho_{t,f}} \tag{4.40}$$

where

$$\rho_{t,f} = \frac{\int f(\theta^{i+t}) f(\theta^i) p(\theta^i) \mathrm{d}\theta^i}{\int f(\theta^i)^2 p(\theta^i) \mathrm{d}\theta^i} \tag{4.41}$$

denotes the t-lag autocorrelation of the function $f(\theta)$ in the Markov chain.

The effective sample size thus takes into account all possible lags of autocorrelations, recognizing that highly autocorrelated chains represent fewer bits of independent information than an uncorrelated analog. In practice, estimating the effective sample size of a chain is done by estimating the autocorrelations. For our purposes, we use the effective sample size estimator implemented in the R package `coda` (Plummer et al., 2006).

We should note that we have not formally proved that our proposed HMC + FlyMC algorithm (as well the other algorithms considered here) produces a geometrically ergodic Markov chain (see Mangoubi and Smith (2017); Livingstone et al. (2019); Mangoubi and Smith (2019) for recent progress on related problems). However, the effective sample size remains an intuitive metric for the efficiency of a Markov chain approximation, due to its penalization of autocorrelations. Moreover, conservative confidence intervals based on the effective sample size can still be constructed even when geometric ergodicity does not necessarily hold Rosenthal (2017). For this reason, we feel it is still a suitable metric to use when comparing chains.

To assess criterion 2 (the computational expense of generating a chain), we simply measure the runtime[¶] of the algorithm. Although there is opportunity for parallelization in some of the algorithms (e.g. simultaneous updating of FlyMC variables), we do not take advantage of such opportunities—all our implementations perform the various steps in series.

Overall, both criterion 1 and criterion 2 should be considered simultaneously when judging the efficiency of a Markov chain—a fast MCMC algorithm is not necessarily accurate, and an accurate MCMC algorithm is not necessarily fast. The most popular metric for combining these criteria is the effective sample size per second (Gamerman and Lopes, 2006), defined by

$$\text{ESS per second} = \frac{\text{ESS}_f(\theta^1, \dots, \theta^N)}{\text{time (in seconds) taken to compute the chain } \theta^1, \dots, \theta^N}, \tag{4.42}$$

sometimes referred to as Markov chain efficiency (de Valpine, 2018). This metric provides a straightforward way to compare the performance of two MCMC algorithms. If one MCMC algorithm produces twice as many effective samples per second as another, that means it is twice as efficient, obtaining an equally accurate approximation of a desired posterior expectation in roughly half the time. This begs the question—what posterior expectation do we desire?

Typically, the posterior mean of the various parameters is the natural choice (Carpenter et al., 2017). However, for the LPM, the posterior mean of each latent positions is inappropriate. Due to the invariance of the likelihood under isometric transformations (e.g. rotations and reflections (Shortreed et al., 2006)), all latent positions are guaranteed to have posterior mean of 0. If our target were the posterior means of the positions, MCMC would be unnecessary.

A reasonable target should instead be well-identified in the model, such as the probabilities of edges between the nodes. These values depend both on the latent positions and the parameters of the link function. However, there are $n(n-1)/2$ such probabilities in total. For large networks, computing the effective sample size for all of them is computationally burdensome. Moreover, many of these expected probabilities will be very close to 0 in sparse networks, creating numerical underflow problems in practice.

To avoid having to compute the Markov efficiency for all $n(n-1)/2$ unique pairs in $[n]^2$, we instead consider a uniformly random subset of 500 dyads (sampled without replacement) for each of our empirical studies in Sections 4.4.2 and 4.4.3. Subsampling drastically reduces the amount of computation required while still providing a summary of how well the chain is mixing for all nodes. To avoid the underflow problems associated with estimating the raw probabilities, we calculate the effective sample size of estimating the log

[¶]None of the algorithms we consider here are memory-intensive. If any were, it might also make sense to report the memory requirements.

probabilities instead. That is, for a given $i, j \in [n]^2$, we consider

$$f(Z, \gamma, \tau, X) = \log(\tau_{x_{ij}}) - \frac{\|z_i - z_j\|^2}{2}. \tag{4.43}$$

as the function for which we calculate the posterior expectation. These functions are much more numerically tractable than the raw probabilities, whilst preserving strong ties with other quantities of interest (i.e. the distance between nodes and the density parameter τ).

Finally, we can now bring all of this together to define an interpretable quantity for reporting the relative efficiency of MCMC algorithms. Metropolis within Gibbs is currently the most popular MCMC algorithm for posterior computation of LPMs, making it the natural choice for the baseline against which to compare other algorithms in our empirical studies (Sections 4.4.2 and 4.4.3). Accordingly, we report each algorithm's efficiency relative to Metropolis within Gibbs for each subsampled dyad. That is, for a chain $\theta^1, \ldots, \theta^N$, we calculate

$$\frac{\mathrm{ESS}_f(\theta^1, \ldots, \theta^N)}{\mathrm{ESS}_f(\theta^1_m, \ldots, \theta^N_m)} \times \frac{\text{time (in seconds) taken to compute the chain } \theta^1_m, \ldots, \theta^N_m}{\text{time (in seconds) taken to compute the chain } \theta^1, \ldots, \theta^N} \tag{4.44}$$

for each dyad, where $\theta^1_m, \ldots, \theta^N_m$ denotes draws according to a well-tuned Metropolis within Gibbs algorithm exploring the same posterior.

4.4.2 Study 1: Synthetic Data

In this empirical study, we investigate the efficiency of split HMC (Section 4.3.1) and split HMC + FlyMC (Sections 4.3.1 and 4.3.2) compared to nine other exact MCMC algorithms from the literature. We are particularly interested in efficiently fitting LPMs to the large, sparse networks that have become increasingly common in modern applications, so we have tooled the study's design to showcase how the algorithms perform as the networks become larger and more sparse.

Our investigation involves fitting Gaussian LPMs to 16 different synthetically generated networks. These networks—stochastically generated according to a GLPMs with pre-specified values of τ, γ^2, and Ω—demonstrate a variety of different sizes and sparsity. Specifically, they represent a full factorial design on the combinations of settings of the parameters $\tau \in \{0.2, 0.8\}$, $\gamma^2 \in \{0.2, 1.0\}$, and the number of nodes $n \in \{50, 100, 200, 500\}$. To generate each of these $2 \times 2 \times 4 = 16$ networks, the latent positions of the nodes are drawn independently from a two-dimensional isotropic Gaussian (i.e. $\Omega = I_n$) prior, and the edges were generated according to the link function with the designated τ, γ parameters. We do not include any observed covariates in this study (i.e. $C = 1$ for $\tau \in [0, 1]$).

Integrating the link probability function K in (4.4) with respect to the isotropic Gaussian prior on Z indicates that the expected probability of an edge between any two nodes i and j in the synthetically generated networks is given by

$$\mathbb{E}(A_{ij}) = \tau \left(1 + 2\gamma^{-2}\right)^{-1}. \tag{4.45}$$

Thus, the parameter configurations described above exhibit a range of sparsity levels: For the $\tau = 0.2, \gamma^2 = 0.2$ networks, roughly 3 percent of dyads have edges, 10 percent of dyads have edges in the $\tau = 0.2, \gamma^2 = 1.0$ networks, 13 percent of dyads have edges in the $\tau = 0.8, \gamma^2 = 0.2$ networks, and 40 percent of dyads have edges in the $\tau = 0.8, \gamma^2 = 1.0$ networks. Though the $\tau = 0.8, \gamma^2 = 0.2$ and $\tau = 0.2, \gamma^2 = 1.0$ configurations correspond to similar edge counts (and thus sparsity levels), sparsity driven by small τ has a different structure than sparsity driven by γ^2, so it is worthwhile investigating both.

For each of the 16 synthetically generated networks, we use (4.44) from Section 4.4.1 to compare the relative performance of 11 different MCMC algorithms. These algorithms primarily vary along two criteria: the proposal used to update Z, and whether or not FlyMC (Section 4.3.2) is used to subsample the non-edges. We consider five different strategies for updating Z: Metropolis within Gibbs (Section 4.2.2), elliptical slice sampling (Murray et al., 2010), elliptical slice sampling within Gibbs (Hahn et al., 2019), split HMC (Section 4.3.1) with $T \approx 2$, and an alternative implementation of Split HMC that uses NUTS (Hoffman and Gelman, 2014) to adaptively choose the integration time T. For each of these five strategies for updating Z, we consider both a standard implementation and FlyMC implementation, amounting to ten different algorithms. Finally, we include HMC as implemented Stan (version 2.18.2 (Carpenter et al., 2017)) as an additional competitor algorithm, bringing the total number of algorithms to 11. Because Stan does not support discrete latent variables, a FlyMC version of Stan is not possible.

We included these algorithms for the following reasons: Metropolis within Gibbs is a baseline, as mentioned above. Both of the elliptical slice sampling algorithms were included as competitors because they represent alternative ways to exploit a Gaussian component in the posterior. The NUTS version of Split HMC was included to investigate whether extra computation cost of adaptively setting the integration time in HMC is worth it for LPMs. Stan was included to ensure our methods represented a worthwhile improvement over out-of-the-box general software.

In addition to inferring Z, all 11 algorithms are tooled to infer τ and γ^2 as well, using a standard uniform prior on τ an inverse gamma (IG(1,1)) prior on γ^2. In Stan, all parameters (Z, γ^2, and τ) are updated as part of a single HMC update. For the five FlyMC algorithms, we alternate between updates of Z as prescribed above, updates of the FlyMC variables θ according to the Metropolis strategy outlined in Section 4.3.2, updates of τ according to the Gibbs strategy outlined in Section 4.3.3, and an update of γ^2 to the Gibbs strategy outlined in Section 4.3.3. For the five remaining standard algorithms, we alternate between updates of Z as prescribed above, τ using a random walk Metropolis algorithm, and γ^2 to the Gibbs strategy outlined in Section 4.3.3.

Where appropriate, we tuned the parameters of the algorithms to promote efficient computation. Stan has a sophisticated (and computationally intensive) tuning strategy for choosing its step size ϵ along with a mass matrix M. For details, see Carpenter et al. (2017). For all of the non-Stan algorithms, we used a light tuning strategy based on a sequence of short (100 iteration) preliminary runs to iteratively select reasonable values for the parameters. For the updates of Z, the Metropolis and Metropolis + FlyMC step sizes were chosen to target an acceptance rate in the range $[0.2, 0.3]$. For the split HMC and split HMC + FlyMC algorithms, the value of ϵ was determined by targeting an acceptance rate within $[0.8, 0.85]$. The value of L was chosen simultaneously to ensure $T = L\epsilon \approx 2$. We have found through a wide array of preliminary experiments that these values tend to give results that are close to optimal without taking too much tuning time. For the NUTS algorithms, the value of ϵ chosen for the analogous split HMC algorithm was used. The elliptical slice algorithms have no tuning parameters. The step-size for the Metropolis τ updates was also chosen based on short preliminary runs, targeting an acceptance rate in the range $[0.2, 0.3]$. The step sizes used to update Z for split HMC, split HMC + FlyMC, Metropolis within Gibbs, and Metropolis within Gibbs + FlyMC are provided in Figure 4.6 in Section 4.6.3.

For each of the 16 synthetic networks, we performed a single run of each algorithm for 10000 iterations (aside from Stan which we ran for just 2000 iterations due to its much longer runtime[‖]). Each algorithm was initialized identically, using the maximum likelihood estimate of Z and the true values of τ and γ^2 as starting points. Figures 4.1 and 4.2 illustrate the relative efficiency of the eleven MCMC algorithms used to compute the posterior for the 16 different networks, with Figure 4.1 showing the standard implementations and Figure 4.2 showing the FlyMC implementations. For each posterior inference algorithm applied to each network, we calculated the Markov chain efficiency described in Section 4.4.1 for a random subset of 500 dyads in the network. To facilitate comparison between algorithms and parameter settings, we then determined the ratio in (4.44) for each algorithm and dyad to compare it to the Metropolis within Gibbs baseline. Figures 4.1 and 4.2 report the medians (across all edges) of these ratios for each algorithm and parameter setting. They are ordered according to the number of nodes to demonstrate how the relatively efficiency changes as the number of nodes increases.

The results shown in Figure 4.1 and Figure 4.2 demonstrate several phenomena. The first thing to note is that Split HMC and and Split HMC + FlyMC are the standout performers across all networks considered. Split HMC clearly outperforms Metropolis within Gibbs for all networks, and Split HMC + FlyMC clearly outperforms Metropolis within Gibbs for all networks except the 50 and 100 node networks in the sparsest regime. Notably, both implementations of Split HMC with $T \approx 2$ outperform their NUTS counterparts and Stan, demonstrating that the extra computational cost of these strategies adaptively updating L may be unwarranted for LPMs.

[‖]The use effective sample size per second as our metric means that Stan is not penalized for running for fewer iterations

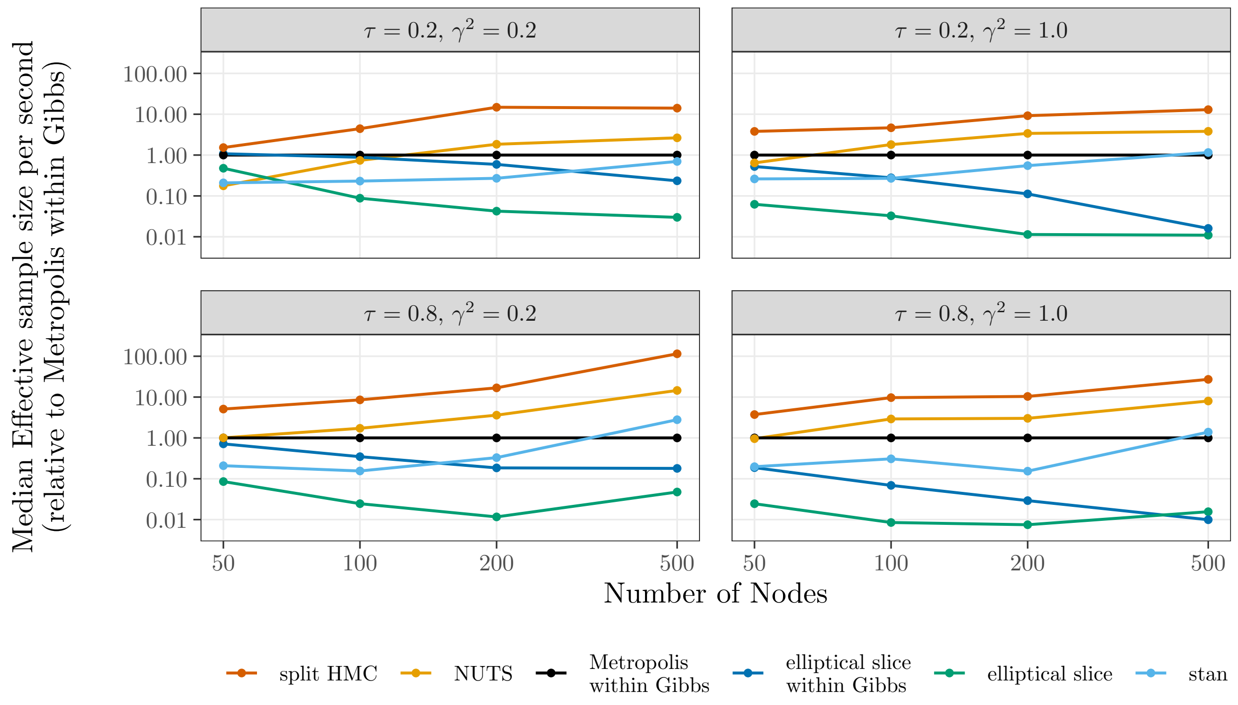


Figure 4.1: A depiction of the relationship between the number of nodes in the synthetically generated networks ($\tau = 0.2, 0.8, \gamma^2 = 0.2, 1.0$) for Empirical Study 1 and the relative efficiency (compared to Metropolis within Gibbs) of the five posterior computation algorithms. For each algorithm, relative efficiency (y axis) is quantified as the median across 500 dyads in the synthetic network of the relative Markov chain efficiency compared to Metropolis within Gibbs. For readability, the results from the analogous FlyMC algorithms are presented separately as Figure 4.2, using the same colors (but dashed instead of solid lines).

All methods based on elliptical slice sampling perform poorly, demonstrating that HMC is a better method for exploiting the near-Gaussianity of the posterior than elliptical slice sampling. Indeed, the elliptical slice sampling algorithms performed worse than Metropolis within Gibbs. The poor performance of the elliptical slice within Gibbs algorithms was due to its runtime—the conditional means and variances of each latent position at each iteration are very expensive to compute. The joint update elliptical slice algorithms performed poorly for the opposite reason. They had much faster runtimes, but the corresponding chains mixed very slowly because the draws exhibited very high autocorrelation.

It is also worth noting that the dominance of the Split HMC methods are more pronounced for larger networks— Split HMC and Split HMC + FlyMC are on the order of 50 to 100 times more efficient than Metropolis within Gibbs for 500 node networks. For the denser $\tau = 0.8$ networks, standard Split HMC performs remarkably well, showing a distinct upward trend, indicating that its dominance over Metropolis within Gibbs would be even more pronounced for larger networks. For the sparser $\tau = 0.2$ networks, Split HMC + FlyMC demonstrates a similar upward trend. The tuned values of ϵ and δ shown in Figure 4.6 in

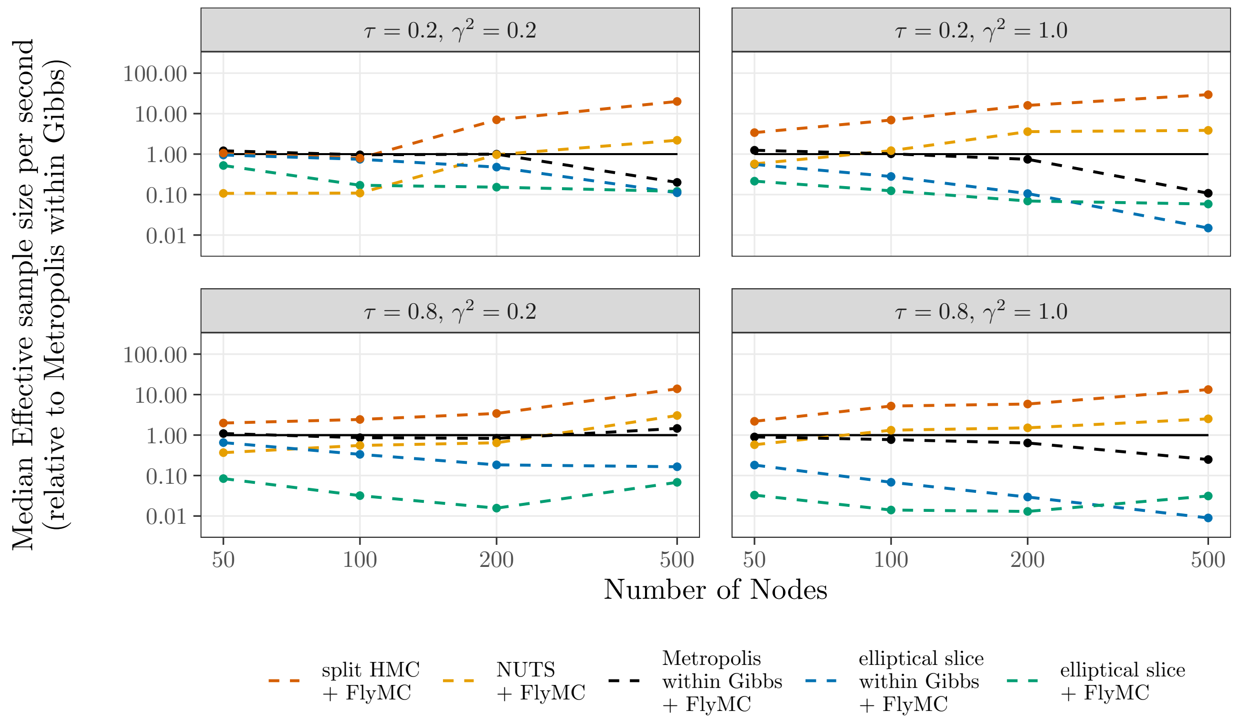


Figure 4.2: A depiction of the relationship between the number of nodes in the synthetically generated networks ($\tau = 0.2, 0.8, \gamma^2 = 0.2, 1.0$) for Empirical Study 1 and the relative efficiency (compared to Metropolis within Gibbs) of the five FlyMC posterior computation algorithms. This figure is a companion to Figure 4.1, presenting the same metric.The solid black baseline is included for easy comparison to Metropolis within Gibbs.

Section 4.6.3 demonstrate that split HMC is more robust to large network sizes: while the tuned step sizes for the Metropolis within Gibbs decay as the number of nodes increases, the tuned values of ϵ for split HMC remain more stable.

To facilitate comparison between the FlyMC and standard implementations of split HMC, we have also included Figure 4.3. Instead of just reporting the median, it summarizes the entire distribution of (4.44) across the 500 sampled dyads. From the side-by-side boxplots, we can see that standard split HMC clearly outperforms split HMC + FlyMC for the $\tau = 0.8$ networks of all sizes. For the $\tau = 0.2$ networks, the comparison is less clear cut. For the very sparse network $\tau = 0.2, \gamma^2 = 0.2$, standard split HMC outperforms the FlyMC version for the smaller networks, but FlyMC edges it out for the 500 node network. It is worth noting that in this very sparse regime for smaller networks, the extreme lack of edges can lead to ambiguity in whether the extreme sparsity is driven by small τ, small γ^2, or both. The joint posteriors of τ, γ^2 for different synthetic networks shown in Figure 4.7 demonstrates this phenomenon—there is a remarkable amount of uncertainty in the posterior of τ for the smaller sparse networks.

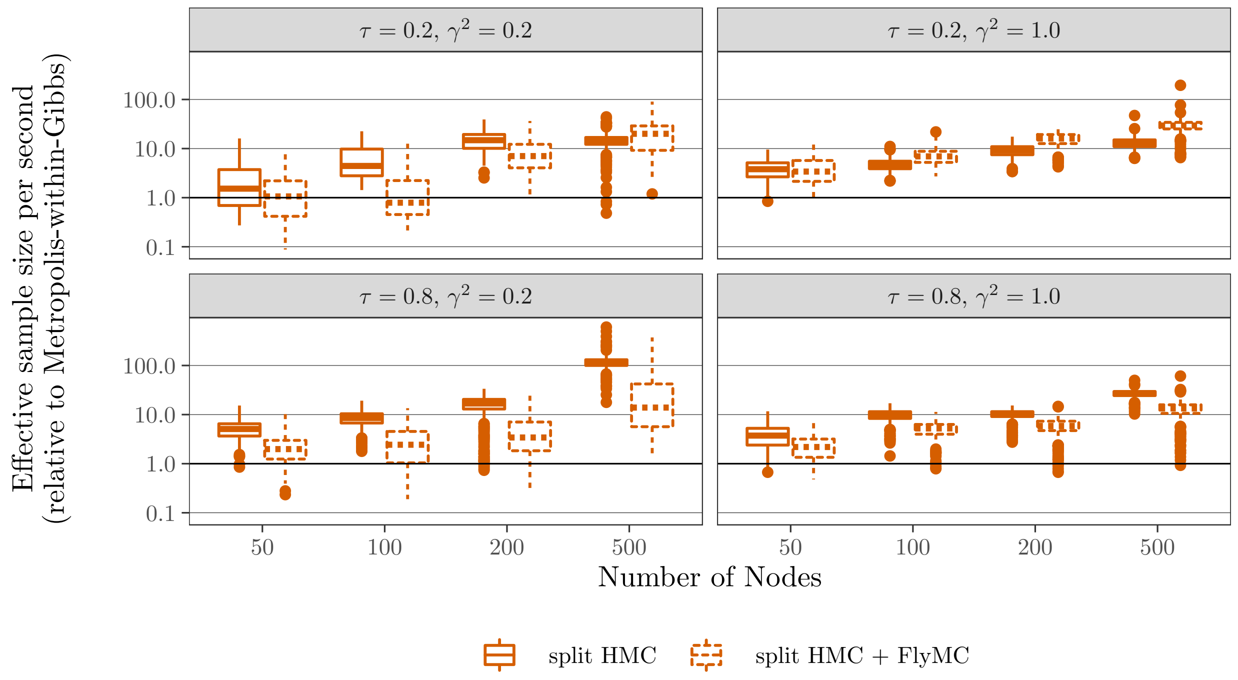


Figure 4.3: Boxplots summarizing the distribution of relative efficiency of Split HMC + FlyMC and standard Split HMC relative to Metropolis within Gibbs across 500 dyads in each network.

We have noticed that the FlyMC updates of τ and θ tend to mix slowly in these uncertain situations, thus leading to slow exploration of the joint distribution of (τ, γ^2). Unencumbered by slow-mixing θ variables, standard FlyMC tends to perform better in these under-identified settings, suggesting that FlyMC should only be used when the τ variable is better identified. This seems to be the case in the $\tau = 0.2, \gamma^2 = 1.0$ networks, where the FlyMC clearly outperforms the standard version.

The full distribution of the relative efficiencies highlights another observation—although the split HMC algorithms tend to outperform Metropolis within Gibbs for the vast majority of dyads, there is often a small minority of dyads for which Metropolis within Gibbs performs better (seen as the lower tails of the boxplots sometimes extending below 1). A thorough investigation into these dyads revealed no apparent pattern for which dyads tend to perform relatively poorly in a given network, suggesting that the primary explanation is simply the high-dimensionality of the posterior—with so many dimensions along which to mix, there will often be a small minority that mix more slowly. Regardless, the underperformance is not drastic for the large networks in which we are interested—split HMC still performs at the same order of magnitude as Metropolis within Gibbs.

From this empirical study, we have demonstrated that split HMC and split HMC + FlyMC tend to outperform competitors in the literature on synthetically generated data. For denser networks, or networks for which τ and γ^2 are poorly identified (i.e. smaller sparse networks), standard split HMC tends to be the better choice. For larger sparse networks, split HMC + FlyMC seems to be the top performer. In all cases, using these strategies will perform far better than simple Metropolis within Gibbs.

However, there is no guarantee that good performance on these synthetically generated networks (i.e. perfectly specified models) will necessarily translate to good performance on the approximately specified models that occur in real data applications. For this reason, we now turn to Study 2, where we use our algorithms to fit LPMs to real data with categorical covariates and structured priors.

4.4.3 Study 2: Network of Information-sharing in a School District

To demonstrate the efficacy of our split Hamiltonian Monte Carlo strategies applied to real data, we now showcase several applications of LPMs to information-sharing networks of teachers and staff in a school district. These applications involve several model/network configurations commonly encountered in practice: networks with categorical covariates that encode known group memberships of the nodes, longitudinally-observed networks with models enforcing serial dependence of the latent positions, and combinations of the two. The data we use in these applications were collected as part of the Distributed Leadership Studies at Northwestern University, a comprehensive program of research involving several longitudinal studies of workplace and social interactions among school staff and school systems. For more details about this particular dataset, see Spillane and Hopkins (2013); Spillane et al. (2018).

The networks we use pertain to Auburn Park, a pseudonym for a mid-sized suburban school district in the Midwestern United States. In five separate years, elementary school teachers and staff within this district were surveyed about who in the district they went to for advice, as well as the school in which they worked and other relevant covariates (e.g. what subjects they taught). Over these years, 661 distinct individuals responded to this advice-seeking survey in at least one year. 129 of them were present for all five surveys (some left or entered the district during the survey years).

For the purposes of this empirical study, we have compiled the survey responses into a series of five undirected *information-sharing* networks—one for each year of data. These undirected information-sharing relationships were obtained by symmetrizing the information in the advice-seeking survey. That is, for each network, an edge is present between two individuals if either of them reported going to the other for advice in that year. In addition to the edge information, we have are covariates indicating which individuals worked in the same school in each year (covariate a), and whether or not they had shared information in the previous survey year (covariate b).

From this sequence of networks, we have extracted four different subnetworks:

- *One school, one year*: the information-sharing network of 32 teachers and staff working within the same school (school ID 4) in the first survey year.

Model	**Number of Schools**	**Number of Years**	**Number of Nodes**	**Number of Edges**	**Covariate**
one year one school	1	1	32	150	No covariate
one year, all schools	14	1	326	1363	No covariate
one year, all schools covariate a	14	1	326	1363	Indicator for whether or not individuals work at the same school
all years, one school covariate b	1	5	14 per year	79	Indicator for whether or not individuals shared info in the previous year
all years, all schools covariate b	14	5	129 per year	1038	Indicator for whether or not individuals shared info in the previous year
all years, all schools a, b	14	5	129 per year	1038	Indicator for whether or not individuals work at the same school *and* Indicator for whether or not individuals shared info in the previous year

Table 4.1: A summary of the model and data configurations for Empirical Study 2

- *All schools, one year*: the information-sharing network of 326 teachers and staff across all fourteen schools in the district in the first survey year.
- *One school, all years*: a series of five information-sharing networks of fourteen teacher and staff working within the same school (school ID 4). Each network corresponds to a different survey year.
- *All schools, all years*: a series of five information-sharing networks of 129 teachers and staff working across all fourteen schools in the district. Each network corresponds to a different survey year.

Using these subnetworks, we fit six different models (one model for each of the *one school, one year* and *one school, all years* networks, and two models for each of the *all schools, one year* and *all schools, all years* networks). These models represent six separate settings with which to assess the performance of standard split HMC and split HMC with FlyMC. The specifics of how these four datasets were constructed, as well as the details of the six models we fit to them, are provided below (for quick reference, a summary is available in Table 4.1). For each model fit, the efficiency of Split HMC and Split HMC with FlyMC are reported in Figure 4.4, relative to the baseline Metropolis within Gibbs.

The *one school, one year* network corresponds to all information shared within a specific school in the district (shown as school ID 4 in Figure 4.5) in survey year one. We chose this particular school due because it contains a large number of teachers and staff who were interviewed in all five years. To this network, we fit one model—a Gaussian LPM with no covariates, an independent two-dimensional isotropic Gaussian priors for each z_i, a uniform prior on τ, and an inverse gamma (IG(1,1)) prior on γ^2.

The *all schools, one year* network represents all information shared across all schools in the district (both between and within-schools) in survey year one. To this network, we fit two separate LPMs: one without covariates, and another in which the binary covariate a is used to indicate whether or not the two individuals were working in the same school that year. For both model configurations, we used independent two-dimensional isotropic Gaussian priors for each z_i, a bivariate uniform prior on τ, and an inverse gamma (IG(1,1)) prior on γ^2.

For the *one school, all years* network, we considered the same school (school ID 4) as in the *one school, one year* network. However, we included only those 14 teachers and staff that were surveyed in that school in all five survey years—any individuals that missed at least one survey year or changed schools during the study were excluded. As a result, this data consists of a series of five networks on the same 14 nodes across the five survey years. We fit one model to these five years data—a single longitudinal latent position model (Kim et al., 2018). Doing so is straightforward within our LPM framework; we pooled the five distinct fourteen node networks into a single 70 node network, then treated any of the impossible "across year" dyads (e.g. a year 1 node cannot share information with a year 3 node) as unobserved, omitting them from the likelihood. We model the temporal dependence in the network in two ways. First, we place an autoregressive prior on each nodes' sequence of two-dimensional latent positions—a Gaussian prior where subsequent years are autocorrelated at a 0.95 level. Second, we use the binary covariate b to allow the model to account for "persistent edges". That is, b indicates whether or not the edge being considered was present in the previous year. As before, we place independent uniform priors on each value of $\tau \in [0,1]^2$, and an inverse gamma (IG(1,1)) prior on γ^2.

Finally, the *all schools, all years* is sequence of five networks similar to the *one school, all years* network but with all schools in the district considered. Like for the *one school, all years* network, this network includes only those that were surveyed in all five years. However, we do keep those who changed schools within the district (updating their covariates year-to-year). For this data, we fit two separate models for *all schools, all years.* They are analogous to the two models considered for *all schools, one year* but they also incorporate longitudinal dependence as in the *one school, all years* model. That is, both *all schools, all years* models involve a two-dimensional Gaussian prior to enforce 0.95 serial correlation among an individuals's latent position in adjacent survey years. The two models differ in how their covariates are structured. One model, analogous to the "no covariate" model for *all schools, one year*, uses just the binary covariate b to indicate whether the edge was present in the previous year (as in the *one school, all years* model). The second model, analogous to the covariate model for *all schools, one year*, incorporates the covariate information from both a and b: x_{ij} takes on four separate values. The values depend on whether or not the teachers/staff work in the same school (a) and whether or not the edge was present in the previous survey year (b).

Note that we chose and structured the datasets and models above to ensure simplicity; they allow us to consider different LPM set-ups in the study without having to account for irregularly missing data or any potential artifacts thereof.

For all six models, we employed the same preliminary run tuning strategy as in Study 1 (Section 4.4.2) to choose ϵ for the split HMC algorithms (with $T \approx 2$), as well as the random-walk step size for each parameter in the Metropolis within Gibbs and the update of τ in standard split HMC. Each Metropolis within Gibbs chain was run for 20000 iterations, and each standard HMC and HMC + FlyMC chain was run for 10000 iterations. For each algorithm and model, the distribution of relative speed-up (as measured by (4.44)) is summarized using a boxplot for 500 randomly selected dyads in Figure 4.4 (all 496 unique dyads for the one year one school network are shown).

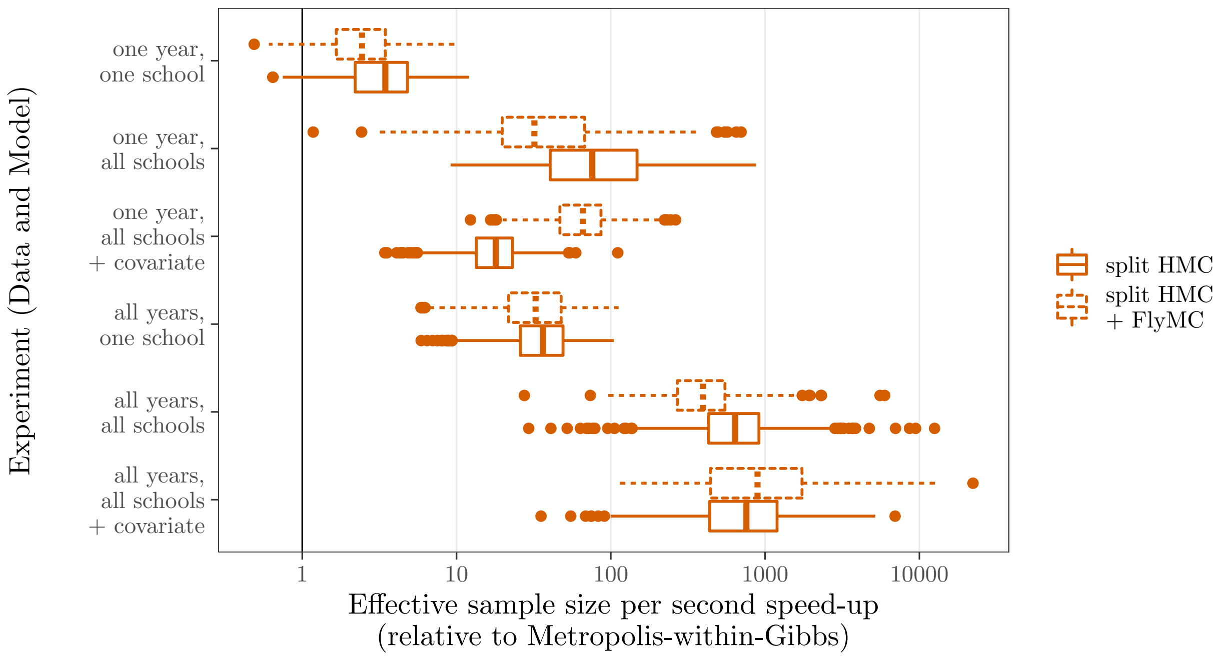


Figure 4.4: Boxplots depicting the relative efficiency (in terms of effective sample size per second) of split Hamiltonian Monte Carlo (both with and without FlyMC) compared to Metropolis within Gibbs. The six different data/model configuration described above are considered. For each configuration, the relative speed-up in effective sample size per second is provided for computing the posterior log probability of a random subset of 500 dyads in the network. Note that the x-axis is provided on the log scale.

In each of the six settings, both the standard and FlyMC versions of split HMC vastly outperform Metropolis within Gibbs, even more so than in Study 1. The speed-up is most pronounced in the all years, all schools with covariates a and b setting, where both algorithms are almost 1000 times more efficient than Metropolis within Gibbs. For the most part, the two HMC algorithms (with and without FlyMC) perform comparably across the different model/data settings. The most noticeable difference occurs in the two models fit to the one year, all schools network. Without the covariate a, standard split HMC performs better. When a is included, the FlyMC version is the better performer. This disparity between the two models provides a good case study for when FlyMC is most useful, so we will now dig deeper into how the inclusion/exclusion of the covariate affects the model fits and Monte Carlo algorithms.

Figure 4.5 summarizes the fits of the two separate models on the one year, all schools network. For both models, the nodes are arranged according to point estimates of their latent positions. These point estimates were obtained by computing the expectation of the matrix of squared distances between the latent positions, then using multi-dimensional scaling to extract the optimal two-dimensional embedding based on these distances. Contours of the posterior distribution for three nodes at different schools are included to demonstrate the uncertainty in the corresponding posteriors. These contours were obtained by applying a Procrustes transformation to all of the posterior samples to best align them with the point estimates of the latent positions.

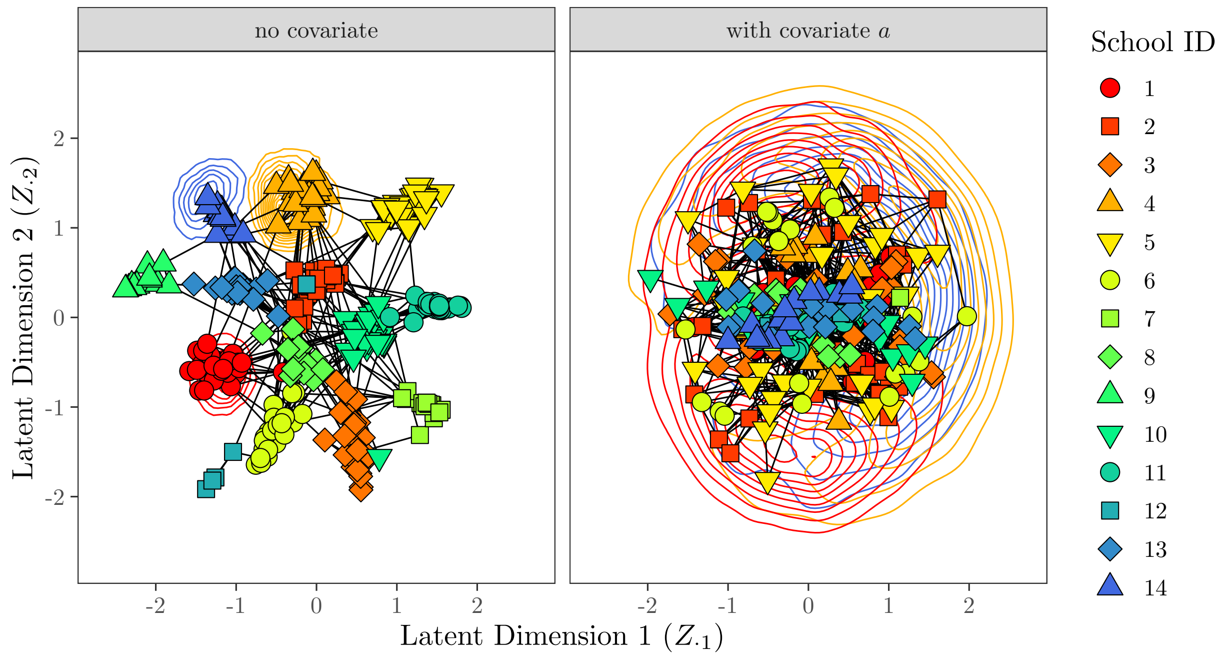


Figure 4.5: The left panel depicts point estimates of the nodes' latent positions in the one year, all schools model. The right panel depicts point estimates of the nodes' latent positions in the one year, all schools, covariate a model. Both sets of point estimates are obtained via 2-dimensional multi-dimensional scaling on the corresponding posterior expectation of the matrix of squared latent distances. In both plots, each node's shape/color combination is assigned according to the individuals's school. Uncertainty contours of the latent positions are depicted for three individuals at different schools (one from school 1, one from school 4, one from school 14—the contours are colored according to school). Finally, the information-sharing relationships are included as edges.

Figure 4.5 demonstrates that information-sharing tends to be mostly concentrated within schools. The school information is unavailable in the no covariate model, but the LPM still manages to capture the strength of this effect using the latent positions. Their estimated values effectively act as a proxy for the held-out covariate information, clustering the individuals in space according to the schools at which they work. The posterior expectation of γ^2 for this model is thus quite small (a posterior expected value of approximately 0.15) ensuring the link probability function decays rapidly to capture the rarity of edges between clusters. With most nodes' latent positions tending to be close to others within the same school, the value of τ in this model effectively captures the rate of within-school ties—its posterior expected value in the model is approximately 0.43. The few between-school ties that are present in the network determine the relative positioning of the different school clusters in the posterior fit.

Explicitly including the school covariate a in the model leads to a much different fit of the latent positions. Because the estimated value of τ directly captures the large disparity between the within-school edge density and between-school edge density (τ has a posterior mean of 0.86 for within-school ties versus 0.01 for between-school ties), the latent positions are free to capture any residual structure in the network. The corresponding posterior shown in Figure 4.5 is diffuse and unstructured—the uncertainty contours for the three nodes covers most of the nodes' estimated latent positions. This suggests that the majority of the network's structure is captured by the covariate. Accordingly, the posterior expected value of γ^2 in this setting is approximately 1.19—much larger than the no covariate setting—showing that the edge probabilities decay more gradually with latent distance.

The superior performance of the FlyMC algorithm in the covariate a setting is due to its ability exploit the sparsity (modeled by τ) between schools. The corresponding θ variables drastically downsample the number of dyads involved in each likelihood and gradient calculation, speeding up computation. This illustrates that FlyMC is especially useful when applied to dyads for which the value of $\tau_{x_{ij}}$ is expected to be small.

Shifting focus to the *all years, all schools* models, an inspection of their fits reveals that the models with and without a differ in a similar way to the one year models. For the model without covariate a, the posterior means of the τ variables are 0.27 (when the previous year is a non-edge) and 0.96 (when the previous year is an edge). The posterior mean of γ^2 is 0.18, reflecting the same tight separation of the latent positions into within-school clusters. For model with covariate a, the posterior means of the τ variables are 0.01 (for a between-school dyads when the previous year is a non-edge), 0.74 (for a within-school dyad when the previous year was a non-edge), 0.32 (for a between-school edge when the previous year was an edge), and 0.86 (for a within-school edge when the previous year was an edge). Again, the τ variable captures most of the structure in the network. The posterior mean of γ^2 is 1.09. For both of these longitudinal models, the particularly strong performance of the split HMC algorithms compared to Metropolis within Gibbs can be attributed to two factors: the size of the network, and the extra structure imposed by the temporal autocorrelation in latent positions. The joint gradient-informed update of all nodes in split HMC is particularly well-suited to account for the autocorrelation in the prior, where the uninformed random walk update of Metropolis within Gibbs is not.

Surprisingly, despite the small value of the between-schools τ for the model with covariate a, we do not see much of a disparity between the performance of the standard and FlyMC implementations of split HMC when fitting the all years, all schools, covariates a, b model. This is due partially to the slower mixing of the τ variables in the FlyMC implementations than in the standard implementation—although they facilitate a closed-form Gibbs update, the auxiliary θ variables can also lead to extra autocorrelation in the chains for τ as they slowly evolve. Another contributor to the smaller disparity is a larger gap between tuned values of ϵ. Our approach for tuning ϵ and L yielded $\epsilon = 0.31$, $L = 7$ for the standard implementation, and $\epsilon = 0.14$ and $L = 15$ for the FlyMC implementation. In the corresponding one year models, the gap was not as large— $\epsilon = 0.22$, $L = 10$ for the standard implementation compared to $\epsilon = 0.14$ and $L = 15$ for the FlyMC implementation.

4.5 Concluding Remarks

In this chapter, we proposed a new algorithm based on a split HMC for inferring the latent positions in Gaussian LPMs, as well as strategies for updating the parameters of the link function (4.4). Moreover, we described an auxiliary FlyMC algorithm for subsampling the non-edge dyads while keeping the Monte Carlo algorithm exact. We conducted two empirical studies to investigate the performance of the split HMC and split HMC + FlyMC approaches: one on synthetic data (Section 4.4.2) and one on real data concerning information-sharing among teachers and staff (Section 4.4.3).

Our synthetic data study demonstrated that when split HMC and split HMC + FlyMC are tuned to have an acceptance rate of around 0.8-0.85 (for $T \approx 2$), these algorithms vastly outperform similarly well-tuned competitors in the literature, especially for large networks. The competitors we considered included standard algorithms such as Metropolis within Gibbs and elliptical slice sampling, as well as more sophisticated HMC algorithms such as the NUTS and Stan that adaptively set T. Across the board, our implementations of split HMC and split HMC + FlyMC outperformed all competitors, even the more sophisticated HMC algorithms. Of the two new algorithms proposed in this chapter, standard split HMC seemed to be the better performer for denser networks, as well as smaller sparse networks for which the link function parameters were poorly identified. Split HMC + FlyMC performs best on large, sparse networks that contain sufficient information to identify the link function parameters. Given these results, we would expect split HMC + FlyMC to perform especially well in settings for which τ is very small, such as the sparse graphon version of GLPMs (Borgs et al., 2014; Spencer and Shalizi, 2019).

At first, we were surprised by the extent to which the HMC integration time T at approximately 2 outperformed the adaptive strategies for setting T used in NUTS and Stan. Adaptive strategies have been shown to outperform fixed integration strategies across a variety of models and perform at least comparably for others (Hoffman and Gelman, 2014; Betancourt, 2016). However, the relatively poor performance of these algorithms on LPMs is less surprising when one considers the criteria used by Stan and NUTS to choose T. Both algorithms are configured to optimize the mixing of the latent positions. However, as we discussed in Section 4.4.1, the latent positions themselves are under-identified in the posterior—the edge probabilities or latent distances are more appropriate targets. This insight suggests a potential avenue of future research—perhaps an adaptation of the NUTS criterion that optimizes the mixing of user-specified functions of the parameters would be a worthwhile extension.

The real data studies involving covariates in Section 4.4.3 demonstrated FlyMC also performs well when τ is allowed to vary by a categorical covariate, taking on small values for just some categories of the covariate. Another potential avenue of future research would be to consider a hybrid of standard split HMC and split HMC + FlyMC. FlyMC would be applied to a subset of the dyads, such as those possessing covariate categories for which τ is expected to be small. This would for FlyMC to be exploited where it is most effective, while limiting any potential mixing problems due to the introduction of additional auxiliary variables.

In Study 2, we also considered the application of our algorithm to fit Gaussian LPMs for longitudinal networks. Of the variety of different ways to configure longitudinal LPMs (Kim et al., 2018), we used a simple model structure that used a structured Gaussian prior to promote serial dependence of the nodes' latent positions across time points, as well as covariates to promote persistent edges across time points. For these models, our split HMC algorithm performed particularly well because gradient-informed proposals naturally accommodate the extra structure in the prior. Therefore, we anticipate that our insights could also be applied to achieve substantial computational gains in other more structured priors for LPMs, such as latent cluster model (Krivitsky et al., 2009) or the multi-resolution network model Fosdick et al. (2018). Recently, Turnbull (2020) explored the use of sequential Monte Carlo (Doucet and Johansen, 2009) for Bayesian inference of longitudinal latent position models. They found that the scaling of the algorithm was poor for large networks due to the computational complexity of evaluating the likelihood. These findings suggest that a combination of their approach with the algorithms we present here could be fruitful.

Finally, it is worth briefly commenting on our motivation for focusing on the Gaussian link function instead of the traditional logistic link function. When proposing the GLPM, Rastelli et al. (2016) argued that the Gaussian LPM yields results that are analogous and comparable to those of the logistic LPM, whilst providing interpretable link function parameters and making it easier to derive theoretical results. Since then, Spencer and Shalizi (2019) proved consistent estimation results for the Gaussian link function—analogous results under the logistic link function remain an open problem. In this chapter, we have demonstrated additional benefits of the Gaussian link function over the logistic link function. The closed-form Gaussian decomposition underlying our split HMC algorithm is not possible with a logistic link, and the FlyMC strategy cannot be applied because it relies on a factorization of the sparsity parameter τ from the link function. Moreover, the likelihood of the logistic link LPM is not differentiable when two nodes have the same position, which complicates the application of HMC at all. For all of these reasons, we propose that the Gaussian LPM would be a more suitable "default" choice of the link function—the logistic link should only be used when its particular functional form is justified by the application.

4.6 Additional Details

4.6.1 Computational Details of Experiments

The implementations of all of the algorithms we used for the empirical studies in Section 4.4.2 and Section 4.4.3 have been made public as part of the R package `LatentPositionNetworks` (Spencer, 2020). The Metropolis within Gibbs algorithms use a multivariate uniform distribution over $[z_i - \delta, z_i + \delta]$ as the proposal distribution q_δ, with δ tuned to target an acceptance rate within 20 and 30 percent. Similarly, the value of ϵ for the split HMC algorithms was tuned to target an acceptance rate between 80 and 85 percent. The tuned values of ϵ and δ for all of the configurations in Study 1 are available in Figure 4.6.3.

All experiments (except those involving Stan) were run using the Bridges High Performance Computing System (Nystrom et al., 2015) at the Pittsburgh Supercomputing Center. The computing costs were supported by XSEDE Integrated Advanced Digital Services (Towns et al., 2014). Because of a software version incompatibility issue, we had to instead run the Stan experiments on a Hydra computing cluster supported by the Department of Statistics and Data Science at Carnegie Mellon University. Timing tests revealed that the Bridges supercomputer runs roughly 3.3 to 3.6 times slower than analogous runs on the Hydra computing cluster. To facilitate direct comparisons between the Hydra and Bridges experiments, all run times of the Stan algorithms were multiplied by a factor of 3.3 when determining the comparisons shown in Figure 4.1. Version 2.18.2 of `rstan` was used to run the experiment, and `coda` version 0.19-2 was used when calculating effective sample sizes.

4.6.2 Full Conditional Distributions

Sections 4.6.2 and 4.6.2 provide the details of thetargeted joint distribution as well as the relevant conditional distributions and algorithmic steps for split HMC + FlyMC and standard split HMC, respectively.

Split HMC + FlyMC

The full joint distribution of all observed data and parameters for our split HMC + FlyMC strategy (after applying the re-parametrization described in Section 4.3.3) can be decomposed as

$$p(A, U, Z, \theta, \tau, \gamma^2 \mid x, a_*, b_*, \alpha, \beta, \Omega) = p(U|\gamma^2, A, \Omega) p(A \mid Z, \theta, \tau, \gamma^2, x) p(Z \mid \gamma^2, \Omega) \tag{4.46}$$

$$\times\, p(\theta \mid \tau, x) p(\tau \mid \alpha, \beta) p(\gamma^2 \mid a_*, b_*), \tag{4.47}$$

where $x \in [C]^{n\times n}$ denotes the observed covariates, $a_*, b_* \in \mathbb{R}_+$ denote the hyperparameters for the inverse gamma prior on γ^2, $\alpha, \beta \in \mathbb{R}_+^C$ denote the hyperparameters for the beta prior(s) on τ, and Ω denotes the prior covariance for the latent positions Z before the re-parametrization. The full expression for each of the components in the decomposition is

$$p(A \mid Z, \theta, \tau, \gamma^2, x) = \left(\prod_{\{i,j\}\in E_A} \tau_{x_{ij}}\right) \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T L_A Z_{\cdot\ell}\right) \prod_{\substack{\theta_{ij}=1\\ A_{ij}=0}} \left(1 - \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)\right) \tag{4.48}$$

$$p(Z \mid \gamma^2, \Omega) = \frac{1}{(2\pi)^{nd/2}\gamma^{nd}\det(\Omega)^d} \exp\left(-\frac{1}{2\gamma^2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right) \tag{4.49}$$

$$p(\gamma^2 \mid a_*, b_*) = \frac{b_*^{a_*}}{\Gamma(a_*)} (\gamma^2)^{-a_*-1} \exp\left(-\frac{b_*}{\gamma^2}\right) \tag{4.50}$$

$$p(\theta \mid \tau, x) = \prod_{\{i,j\}\in A} \tau_{x_{ij}}^{\theta_{ij}} \left(1 - \tau_{x_{ij}}\right)^{1-\theta_{ij}} \tag{4.51}$$

$$p(\tau \mid \alpha, \beta) = \prod_{c=1}^{C} \frac{\Gamma(\alpha_c + \beta_c)}{\Gamma(\alpha_c)\Gamma(\beta_c)} \tau_c^{\alpha_c - 1} \left(1 - \tau_c\right)^{\beta_c - 1} \tag{4.52}$$

$$p(U|\gamma^2, A, \Omega) = \frac{\det\left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^d}{(2\pi)^{nd/2}} \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} U_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^{-1} U_{\cdot\ell}\right) \tag{4.53}$$

where $\det(\cdot)$ denotes the determinant of a matrix and $\Gamma(\cdot)$ denotes the Gamma function. To perform MCMC on this distribution, we alternate through the following conditional updates

1. Use split HMC described in Section 4.3.1 to update (Z, U) according to the conditional posterior density $p(U, Z \mid A, \theta, \gamma^2, \Omega)$ defined by

$$p(U, Z \mid A, \theta, \gamma^2, \Omega) \propto \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} Z_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right) Z_{\cdot\ell} + U_{\cdot\ell}^T \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right)^{-1} U_{\cdot\ell}\right) \tag{4.54}$$

$$\times \prod_{\substack{\theta_{ij}=1\\ A_{ij}=0}} \left(1 - \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)\right) \tag{4.55}$$

Note that in this case, the mass matrix for HMC is given by

$$M = \left(\frac{1}{\gamma^2}\Omega^{-1} + L_A\right) \tag{4.56}$$

which amounts to having it adaptively updated according to γ^2.

2. Apply the Metropolis-Hastings strategy described in Section 4.3.2 to update each of θ_{ij} for which $A_{ij} = 0$ according to the conditional posterior density $p(\theta \mid \tau, x)$ defined by

$$p(\theta_{ij} = 0 \mid A_{ij} = 0, \tau_{x_{ij}}) = \frac{1 - \tau_{x_{ij}}}{1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2}\|z_i - z_j\|^2\right)}. \tag{4.57}$$

Recall that because $p(\theta_{ij} = 0 \mid A_{ij} = 1, \tau_{x_{ij}}) = 0$, the θ_{ij} for which $A_{ij} = 1$ need not be updated—they are known to be fixed at one.

3. Apply the Gibbs updates described in Section 4.3.3 to update each entry in τ according to the beta conditional posterior densities

$$p(\tau_c|\theta, \alpha_c, \beta_c) = \frac{\Gamma(\alpha_c + \beta_c + \Theta_c^0 + \Theta_c^1)}{\Gamma(\alpha_c + \Theta_c^0)\Gamma(\beta_c + \Theta_c^1)} \tau_c^{\alpha_c + \Theta_c^1 - 1} (1 - \tau_c)^{\beta_c + \Theta_c^0 - 1} \tag{4.58}$$

where Θ_0^c and Θ_1^c are defined as in (4.34) and (4.35), reproduced below for easy access.

$$\Theta_c^0 = |\left\{\{i,j\} \in [n]^2 : \theta_{ij} = 1 \text{ and } x_{ij} = c\right\}|$$
$$\Theta_c^1 = |\left\{\{i,j\} \in [n]^2 : \theta_{ij} = 0 \text{ and } x_{ij} = c\right\}|.$$

4. Apply the Gibbs update described in Section 4.3.3 to update γ^2 according to the inverse gamma conditional density

$$p(\gamma^2|a_*, b_*, Z, \Omega) = \frac{\left(b_* + \frac{1}{2}\sum_{\ell=1}^d Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)^{a_* + \frac{nd}{2}}}{\Gamma(a_* + \frac{nd}{2})\,(\gamma^2)^{a_* + \frac{nd}{2} + 1}} \exp\left(-\frac{\left(b_* + \frac{1}{2}\sum_{\ell=1}^d Z_{\cdot\ell}^T \Omega^{-1} Z_{\cdot\ell}\right)}{\gamma^2}\right). \tag{4.59}$$

Note that this expression above arises only after marginalizing the momentum variables U. Typically, after such a marginal update in MCMC, the U parameter would need to be updated according to its conditional distribution. In practice, this is not necessary, as U is not one of the target parameters (moreover, the Gibbs update is immediately applied again in the following Step 1).

Split HMC

The full joint distribution of all observed data and parameters for our split HMC strategy (after applying the re-parametrization described in Section 4.3.3) can be decomposed as

$$\begin{aligned} p(A, U, Z, \tau, \gamma^2 \mid x, a_*, b_*, \alpha, \beta, \Omega) = p(U|\gamma^2, A, \Omega) p(A \mid Z, \tau, \gamma^2, x) p(Z \mid \gamma^2, \Omega) \quad (4.60)\\ \times p(\tau \mid \alpha, \beta) p(\gamma^2 \mid a, b), \quad (4.61) \end{aligned}$$

where $x \in [C]^{n \times n}$ denotes the observed covariates, $a_*, b_* \in \mathbb{R}_+$ denote the hyperparameters for the inverse gamma prior on γ^2, $\alpha, \beta \in \mathbb{R}_+^C$ denote the hyperparameters for the beta prior(s) on τ, and Ω denotes the prior covariance for the latent positions Z before the re-parametrization. The full expression for each of the components in the decomposition is

$$p(A \mid Z, \tau, \gamma^2, x) = \left(\prod_{\{i,j\} \in E_A} \tau_{x_{ij}} \right) \exp\left(-\frac{1}{2} \sum_{\ell=1}^{d} Z_{\cdot \ell}^T L_A Z_{\cdot \ell} \right) \prod_{\{i,j\} \notin E_A} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2} \|z_i - z_j\|^2 \right) \right) \quad (4.62)$$

$$p(Z \mid \gamma^2, \Omega) = \frac{1}{(2\pi)^{nd/2} \gamma^{nd} \det(\Omega)^d} \exp\left(-\frac{1}{2\gamma^2} \sum_{\ell=1}^{d} Z_{\cdot \ell}^T \Omega^{-1} Z_{\cdot \ell} \right) \quad (4.63)$$

$$p(\gamma^2 \mid a_*, b_*) = \frac{b_*^{a_*}}{\Gamma(a_*)} (\gamma^2)^{-a_* - 1} \exp\left(-\frac{b_*}{\gamma^2} \right) \quad (4.64)$$

$$p(\tau \mid \alpha, \beta) = \prod_{c=1}^{C} \frac{\Gamma(\alpha_c + \beta_c)}{\Gamma(\alpha_c) \Gamma(\beta_c)} \tau_c^{\alpha_c - 1} (1 - \tau_c)^{\beta_c - 1} \quad (4.65)$$

$$p(U|\gamma^2, A, \Omega) = \frac{\det\left(\frac{1}{\gamma^2} \Omega^{-1} + L_A \right)^d}{(2\pi)^{nd/2}} \exp\left(-\frac{1}{2} \sum_{\ell=1}^{d} U_{\cdot \ell}^T \left(\frac{1}{\gamma^2} \Omega^{-1} + L_A \right)^{-1} U_{\cdot \ell} \right) \quad (4.66)$$

where $\det(\cdot)$ denotes the determinant of a matrix and $\Gamma(\cdot)$ denotes the Gamma function. To perform MCMC on this distribution, we alternate through the following conditional updates

1. Use split HMC described in Section 4.3.1 to update (Z, U) according to the conditional posterior density $p(U, Z \mid A, \gamma^2, \Omega)$ defined by

$$\begin{aligned} p(U, Z \mid A, \gamma^2, \Omega) \propto \exp\left(-\frac{1}{2} \sum_{\ell=1}^{d} Z_{\cdot \ell}^T \left(\frac{1}{\gamma^2} \Omega^{-1} + L_A \right) Z_{\cdot \ell} + U_{\cdot \ell}^T \left(\frac{1}{\gamma^2} \Omega^{-1} + L_A \right)^{-1} U_{\cdot \ell} \right) \quad (4.67)\\ \times \prod_{\{i,j\} \notin E_A} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2} \|z_i - z_j\|^2 \right) \right) \quad (4.68) \end{aligned}$$

Note that in this case, the mass matrix for HMC is given by

$$M = \left(\frac{1}{\gamma^2} \Omega^{-1} + L_A \right) \tag{4.69}$$

which amounts to having it adaptively updated according to γ^2.

2. Apply a random walk Metropolis to update each entry in τ using its posterior conditional distribution

$$p(\tau_c \mid A, x_{ij}, \alpha_c, \beta_c) \propto \tau_c^{\alpha_c + \zeta_c^1 - 1} (1 - \tau_c)^{\beta_c - 1} \prod_{\substack{x_{ij} = c \\ A_{ij} = 0}} \left(1 - \tau_{x_{ij}} \exp\left(-\frac{1}{2} \|z_i - z_j\|^2 \right) \right) \tag{4.70}$$

where ζ_c^1 is defined as

$$\zeta_c^1 = |\left\{ \{i, j\} \in [n]^2 : A_{ij} = 1 \text{ and } x_{ij} = c \right\}|.$$

We recommend updating each entry individually, using a uniform proposal centered at its current value with step-size tuned to obtain an acceptance rate within 20 and 30 percent. This is the strategy we used throughout the chapter.

3. Apply the Gibbs update described in Section 4.3.3 to update γ^2 according to the inverse gamma conditional density

$$p(\gamma^2 | a_*, b_*, Z, \Omega) = \frac{\left(b_* + \frac{1}{2} \sum_{\ell=1}^d Z_{\cdot \ell}^T \Omega^{-1} Z_{\cdot \ell} \right)^{a_* + \frac{nd}{2}}}{\Gamma(a_* + \frac{nd}{2}) \, (\gamma^2)^{a_* + \frac{nd}{2} + 1}} \exp\left(- \frac{\left(b_* + \frac{1}{2} \sum_{\ell=1}^d Z_{\cdot \ell}^T \Omega^{-1} Z_{\cdot \ell} \right)}{\gamma^2} \right). \tag{4.71}$$

Note that this expression above arises only after marginalizing the momentum variables U. Typically, after such a marginal update in MCMC, the U parameter would need to be updated according to its conditional distribution. In practice, this is not necessary, as U is not one of the target parameters (moreover, the Gibbs update is immediately applied again in the following Step 1).

4.6.3 Additional Figures and Tables

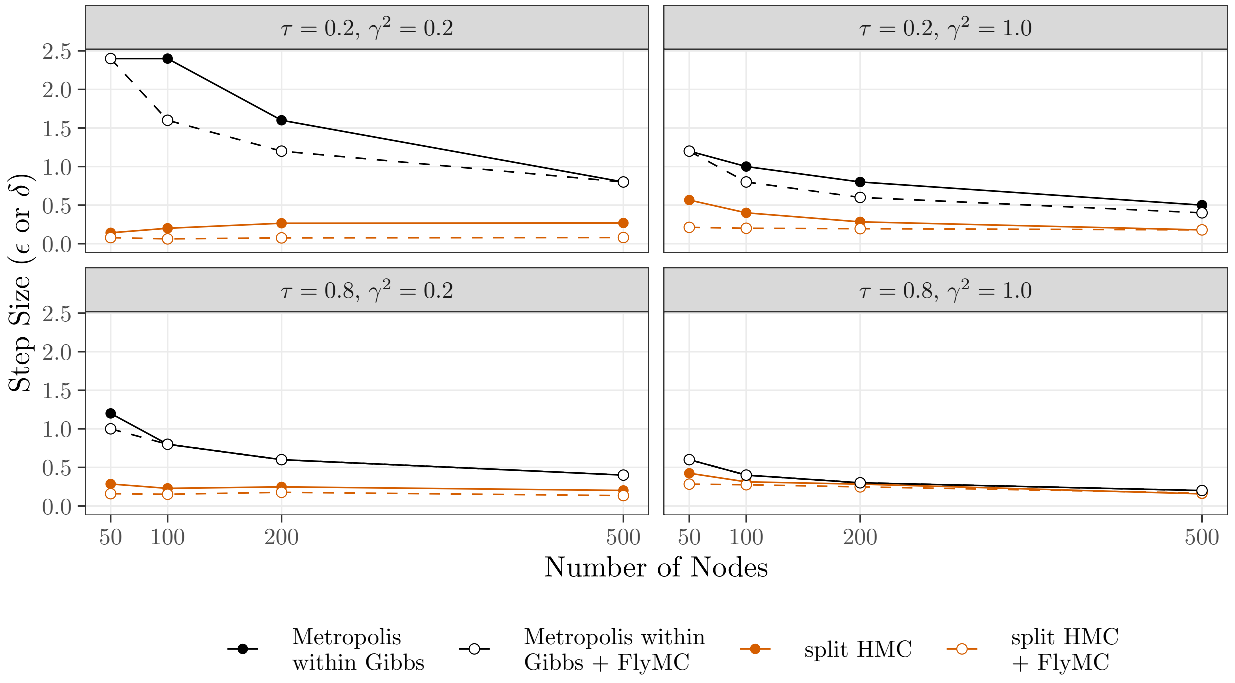


Figure 4.6: Four panels depicting the tuned step size parameters used for Metropolis within Gibbs, Metropolis within Gibbs + FlyMC, split HMC, and split HMC + FlyMC algorithms used to fit the 16 different networks considered in Study 1 (Section 4.4.2). Each panel displays the step size parameter (δ for Metropolis methods and ϵ for HMC methods) used for the 50, 100, 200, and 500 node networks generated using the parameter values featured in the panel heading. Point color, point shape, line color, and line shape are used to distinguish between the four algorithms.

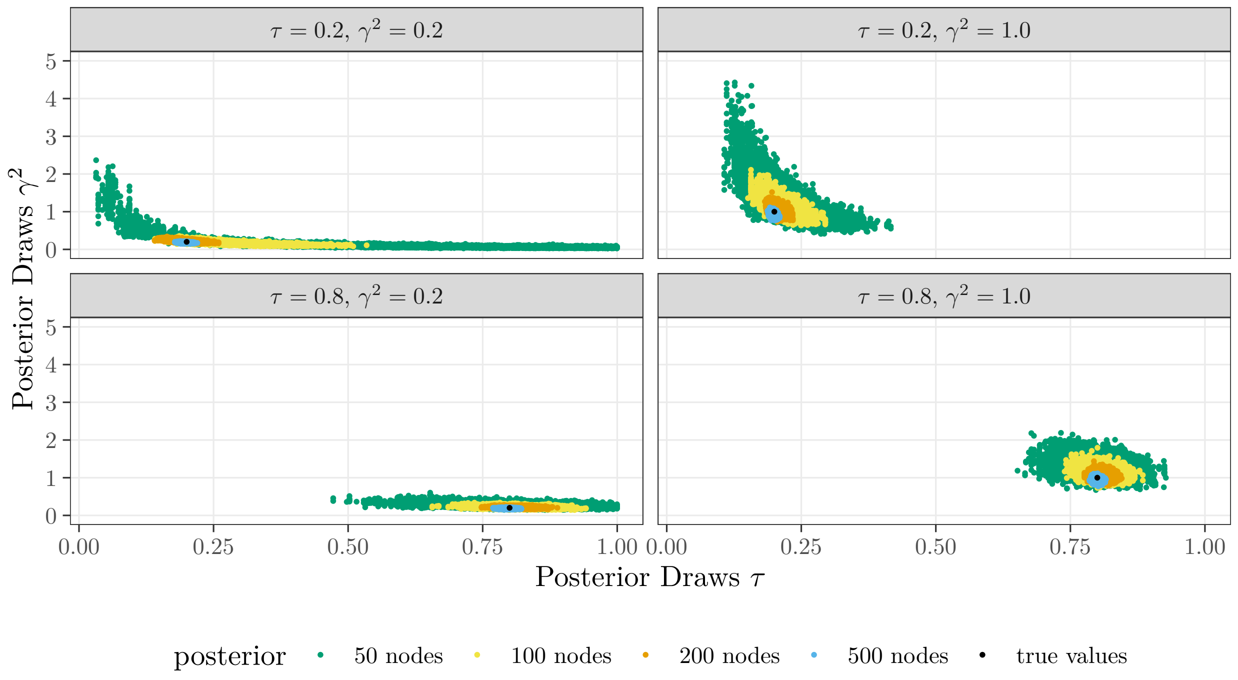


Figure 4.7: Four panels depicting the marginal joint posterior of τ and γ^2 for the 16 different networks considered in Study 1 (Section 4.4.2). Each panel displays draws from the joint posterior for the 50, 100, 200, and 500 node networks generated using the parameter values featured in the panel heading. The draws for the networks of different sizes are differentiated by color, with a black point used to indicate the true value of τ and γ^2 used to generate each synthetic network.

Chapter 5

Bayesian Inference for Neural Spike Train Models via the Spike and Slab Zigzag Process

5.1 Introduction

Brains contain networks of interconnected neurons transmitting information simultaneously in short bursts of electrical activity called spikes (Rieke et al., 1999; Kass et al., 2018). A spike train (Brown et al., 2004; Kass et al., 2018) is a sequence of times at which a neuron is recorded spiking. Historically, spike train recordings monitored just a few neurons at once (Brillinger, 1992; Kass et al., 2018), but current technologies can now record thousands of neurons across multiple brain areas (e.g. Jun et al. (2017)). The breadth of this new data represents an unprecedented opportunity for understanding the brain's function and structure.

Developing and fitting statistical models to such massive neural data presents several new computational and methodological challenges. In particular, standard multivariate spike models based on generalized linear models (GLMs (Truccolo et al., 2005; Kass et al., 2018)) are challenging to fit because each neuron's spiking is modeled as depending on the spike history of all other recorded neurons; the number of coefficients to estimate increases quadratically in the number of neurons being recorded. This large number of coefficients often leads to a high degree of uncertainty in their fitted values, especially for those corresponding to neurons that rarely spike.

For this reason, Bayesian inference is an especially useful framework for fitting spike train GLMs because it provides an intuitive way to quantify uncertainty in model fits. It also allows for additional structure to be promoted in the model coefficients through hierarchical and/or sparsity promoting priors (e.g. Linderman et al. (2016); Vinci et al. (2018); Wu et al. (2019)) that reduce the effective dimension of the parameter space.

Despite these advantages, fully Bayesian inference is notorious for being computationally intensive to perform. The GLMs used for spike train models are typically non-conjugate, so the corresponding posterior expectations are typically computed using Markov chain Monte Carlo (MCMC (Geyer, 1992; Kass et al., 1998)). Unfortunately, popular generic MCMC algorithms such as Metropolis within Gibbs are known to scale poorly with both the dimension of the model and the number of data points (Beskos and Stuart, 2009). This poor scaling has necessitated the development of alternative computational approaches for fitting large GLMs. These alternatives take one of two forms: specialized MCMC algorithms that exploit the structure of the particular problem to drastically improve the computational efficiency of the sampling procedure (e.g. Neal (2011), Polson et al. (2013), Bouchard-Côté et al. (2018), Nishimura and Suchard (2018), Hahn et al. (2019), Sen et al. (2019)), or approximate algorithms that modify/approximate some aspect of the problem to make it more tractable (e.g. Ramirez and Paninski (2014), Barber et al. (2016), Huggins et al. (2017), Blei et al. (2017), Bardenet et al. (2017), Campbell and Broderick (2018), Zoltowski and Pillow (2018), Trippe et al. (2019)).

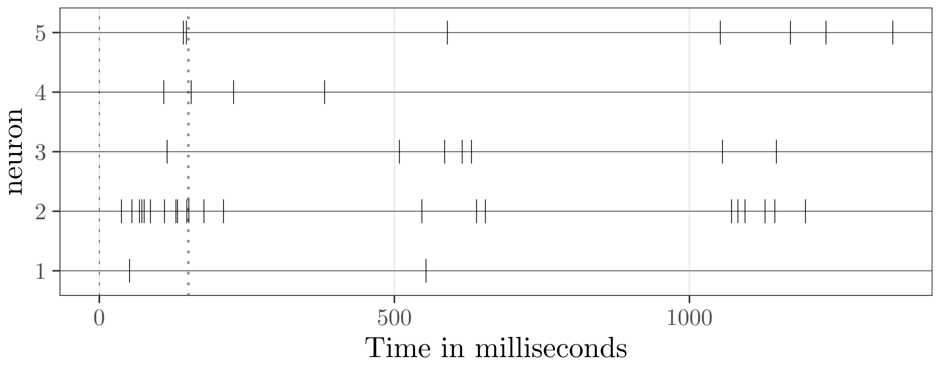
Though the latter approximation-based algorithms are often far more computationally efficient than the former MCMC algorithms, they typically lack the suite of theoretical guarantees and diagnostic tools that accompany MCMC. As a result, these "shortcut" algorithms may yield unreliable results due to inaccurate approximations. To make matters worse, most shortcut methods lack formal diagnostic tests, meaning it can be impossible to judge their accuracy without running the expensive inference procedure that was being avoided in the first place. Some exceptions exist for certain cases (e.g. Yao et al. (2018) and Huggins et al. (2019) for forms variational inference), but as a general rule, shortcut algorithms must be applied with caution—especially in high-dimensions.

For this reason, the focus of this chapter is to develop a specialized MCMC algorithm that is especially well-suited for doing posterior computation of high-dimensional multivariate spike train GLMs. The workhorse of our algorithm is the zigzag process, a recent advance in the realm of non-reversible continuous time Markov chains (Bierkens et al., 2019a) that can efficiently fit massive logistic regressions. Building on the exact data subsampling tricks recently proposed in Sen et al. (2019), we develop a method for simulating the zigzag process that explicitly exploits the sparsity and structure of the design matrix, coefficient vector, and response vector for spike train GLMs. Moreover, we promote sparsity in the coefficient vector through a spike-slab-prior (Mitchell and Beauchamp, 1988) consisting of a point mass at zero and Laplace distribution for each coefficient. We introduce a new parametrization that accommodates the mixture of discrete and continuous components in the posterior, allowing our zigzag process to seamlessly move through the mixed distribution without the need for discrete auxiliary variables.

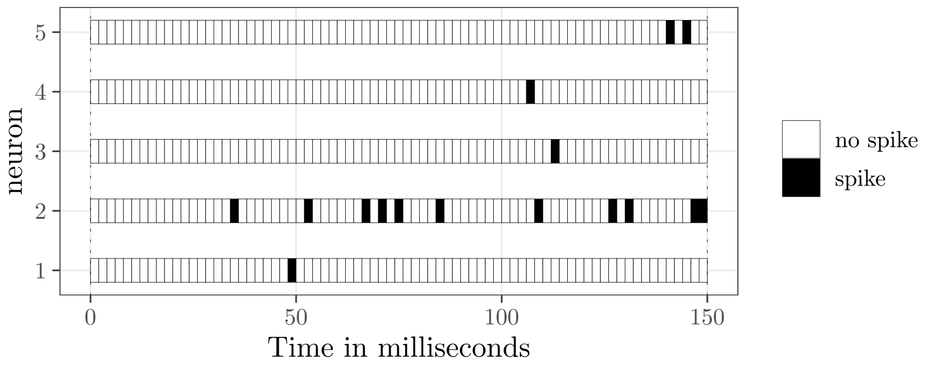
The remainder of this article is organized as follows. Section 5.2 provides background and notation on logistic regression models for spike trains, as well as a general statement of and background on the zigzag process. Section 5.3 outlines our implementation of the zigzag process that is specifically designed to exploit the properties of neural spike train GLMs, as well as the special spike-and-slab prior structure we have developed to obtain sparsity in the regression coefficients. Section 5.4 analyzes the efficiency of our algorithm and outlines a series of synthetic experiments to demonstrate the scenarios in it outperforms the spike-and-slab Polya-Gamma augmentation algorithm of Linderman et al. (2016). Finally, Section 5.5 contains some concluding remarks.

5.2 Background

We consider the problem of modeling the spiking behavior (i.e. spike trains) of an ensemble of N neurons that are being recorded simultaneously. Figure 5.1(a) provides an example of such data following $N = 5$ neurons for 1500 milliseconds. Because neural spikes are instantaneous events whose frequencies and times are known to vary stochastically, spiking behavior is typically modeled as following a multivariate point process (Truccolo et al., 2005). Specifically, each neuron's instantaneous spiking rate is modeled as depending on its own spiking history (e.g. to allow for a refractory period), the spiking history of other neurons (e.g. neurons may excite or inhibit each other), and any extrinsic covariates (e.g. a stimulus that may prompt the neuron to spike). In practice, fitting a point process model that accounts for the relationship between instantaneous spiking rates, spike histories, and covariates can be framed as a logistic regression task.



(a)



(b)

Figure 5.1: (a) depicts all recorded spike times for five neurons—labeled 1 through 5—over a 1500 millisecond time interval. Vertical bars are used to show the time of each spike. (b) depicts the spike counts binned at 2 millisecond intervals for the first 150 milliseconds of the same spike trains shown in (a). For each neuron, a bin is colored black if a spike occurs in its interval, white otherwise. Dotted lines delineating the first 150 milliseconds are shown in both (a) and (b) to facilitate comparison.

5.2.1 Notation

The following notation will be used throughout the chapter. We use $\mathbb{R}$ to denote the set of real numbers, $\mathbb{R}_+$ to denote the set of non-negative real numbers, $\mathbb{N}$ to denote the set of natural numbers, and $[n]$ to denote the set $\{1, \ldots, n\}$ of natural numbers less than or equal to n. For a set S, we use S^d to denote the collection of all d-length vectors with entries from S and $S^{n \times d}$ to denote collection of possible $n \times d$ matrices with entries from S. For two sets S_1, S_2, $S_1 \times S_2$ denotes their Cartesian product.

For a vector $z \in \mathbb{R}^d$, z_i denotes its ith entry and $\|z\|$ denote its Euclidean norm. For a matrix $B \in \mathbb{R}^{n\times d}$, $B_{i\cdot}$ denotes its ith row, $B_{\cdot i}$ denotes its ith column, B_{ij} denote its (i,j)th entry, $B^T \in \mathbb{R}^{d\times n}$ denotes its transpose, $\|B\|$ denotes its Frobenius norm, $\|B\|_1$ denotes its ℓ_1 norm, and B^{-1} denotes its inverse.

5.2.2 Spike Trains as Logistic Regression

To frame the modeling of multivariate spike trains as a logistic regression task, we follow Truccolo et al. (2005) and summarize each neuron's spiking behavior using the spike counts within discrete time bins. Specifically, we separate the duration of an experiment trial into $T \in \mathbb{N}$ separate bins of equal width, then count the number of times each neuron spikes in each bin. Assuming our data come from an experiment consisting of K separate trials of equal duration, our entire spiking dataset can thus be summarized as a a 3-dimensional array Y, where $Y^k_{n,t}$ denotes the spike count for neuron $n \in [N]$ in time bin $t \in [T]$ for trial $k \in [K]$. After discretization, modeling the instantaneous spiking rate of a neuron reduces to modeling the spiking behavior within each bin, conditional on the ensemble spiking history and any covariate information.

We consider the spike train generalized linear model as popularized by (Truccolo et al., 2005). In this model, the expected spike count $\mathbb{E}(Y^k_{n,t})$ of neuron n at time bin t of trial k is modeled as a nonlinear function (we will use the logistic function) of the instantaneous rate $\psi^k_{n,t}$ defined according to

$$\psi^k_{n,t} = b^k_{n,t} + \sum_{u=1}^{N} \sum_{\Delta=1}^{\Delta=t_{\max}} \sum_{f=1}^{F} \beta^f_{un} \cdot Y^k_{u,t-\Delta} \cdot g_f(\Delta). \tag{5.1}$$

Here, $b^k_{n,t} \in \mathbb{R}$ captures the neuron's baseline rate across the trial (potentially varying across k and t), $g_1(t), \ldots, g_F(t)$ are basis functions modeling the time delayed coupling of activity between neurons (e.g. splines (Kass and Ventura, 2001), Laguerre bases (Song et al., 2013), exponential functions (Linderman et al., 2016), or raised cosine bumps (Pillow et al., 2008)), $t_{\max}$ is the maximum time delay at which the model allows coupling to occur, and β^f_{un} models the sign and magnitude of basis function f for the time-delayed coupling between neurons $u \in [N]$ and $n \in [N]$. For our purposes, we assume the basis functions $g_1, \ldots g_F$ are non-negative, upper bounded by 1 on the interval $[0, t_{\max}]$, and 0 elsewhere. Many basis functions satisfy these criteria—in Section 5.4, we use the raised cosine bumps of Pillow et al. (2008).

By choosing the value of T to be sufficiently large (i.e. making the bins sufficiently small), we can ensure each neuron spikes at most once within each time bin. Thus, $Y^k \in \{0,1\}^{N\times T}$ for all $k \in [K]$. Figure 5.1(b) illustrates the binned representation of the first 150 milliseconds of the spike trains shown in Figure 5.1(a). This set-up facilitates the modeling of Y using logistic regression—the 0–1 encoded spiking behavior plays the role of the response variable, with past spiking and covariate information playing the role of the predictors.

Before delving into the details of how to frame (5.1) as a logistic regression for binary response Y, it is worth briefly commenting on other possible GLM set-ups—namely Poisson regression and negative binomial regression—that also fall within the Truccolo et al. (2005) framework. Unlike logistic regression, these regression models assign positive probability to all natural number-valued responses, allowing for any number of spikes to occur in any given time bin. These models can thus accommodate coarser discretizations of the response Y, and correspondingly smaller values of T. Depending on the inference algorithm being used, coarsening the discretization can provide considerable computational advantages—the corresponding reduction in the number of observations will speed up Bayesian inference drastically. However, these computational gains come at the cost of leaving the model insensitive to small time scales; an overly coarse discretization will not capture fine-structure such as the refractory period of a neuron. We thus recommend discretizing Y into a binary response to promote adequate sensitivity of the model.

Once the discretization of Y is fine enough to ensure the probability of more than one spike occurring in a given bin is negligible, both Poisson regression and negative binomial regression are effectively modeling Bernoulli responses. Therefore, any difference between the Poisson, negative binomial, and logistic model fits will come down to subtleties of the nonlinearity in the link function being used. In this work, we have chosen to focus on the case of logistic regression because it ultimately leads to a posterior distribution that admits a log-density with a globally bounded gradient, thus facilitating the fast posterior inference strategy via the zigzag process we outline in Section 5.3. In Section 5.5, we briefly comment on how to choose link function for Poisson regression and negative binomial regression to also obtain a globally bounded gradient.

We can now proceed with the specifics of spike train logistic regression. To streamline our exposition, we make two simplifications. We assume a fixed intercept term b_n for each neuron $n \in [N]$ that is independent of the trial number k and the time. That is, we parametrize the model as $b_{n,t}^k = b_n$ for all $k \in [K], t \in [T]$, ignoring the presence of extrinsic covariates, non-stationarity within a trial, or trial-specific effects. Note that the methods we present here are equally applicable for time-varying and trial-varying baselines, they just require extra bookkeeping and additional coefficients. Moreover, we focus on the case of $K = 1$ (a single trial), dropping the k superscript from Y and ψ to make the notation cleaner. Our approach extends immediately to multiple trials by simply stacking the design matrices and response vectors for all trials into a single dataset.

Using the notation in (5.1), our logistic regression model for the random matrix $Y\,\{0,1\}^{N \times T}$ takes the form:

$$Y_{n,t} \mid \psi_{n,t} \sim^{\text{iid}} \text{Bernoulli}((1 + e^{\psi_{n,t}})^{-1}). \tag{5.2}$$

At first glance, it may be unclear how the above expression corresponds to logistic regression. After all, $\psi_{n,t}$ in (5.1) depends on Y, the response variable. How can Y depend on itself in logistic regression? And where is the familiar design matrix X?

The key to answering these questions lies in recognizing that $\psi_{n,t}$ depends only on those $Y_{n,t-\Delta}$ for which $\Delta \geq 1$. That is, each neuron's current spiking behavior is modeled as depending on the *past* spiking behavior of all recorded neurons. This is akin to how serial dependence is modeled in vector autoregressive time series models (Hamilton and Press, 1994)—past observations serve as covariates for future observations, allowing for standard vector autoregressive models to be fit as straightforward linear regressions. Analogously, fitting our spike train model can be framed as fitting as a logistic regression model. The past spiking behavior of the neurons, combined with the basis functions $g_1, \ldots, g_F$, are used to construct the design matrix X as follows.

For a given set of basis function $g_1, \ldots g_F$, we can construct lower triangular matrices $G^1, \ldots, G^F \in [0,1]^{T \times T}$ according to $G^f_{ij} = g_f(i-j)$. Now,

$$Y_{n,t} \mid X_t, b_n, \beta^*_{\cdot n} \sim^{\text{iid}} \text{Bernoulli}((1 + e^{b_n + X_{t\cdot}\beta^*_{\cdot n}})^{-1}) \tag{5.3}$$

where $\beta^*_{\cdot n} \in \mathbb{R}^{NF}$ denotes the vector concatenation

$$\beta^*_{\cdot n} = \left(\beta^1_{\cdot n}, \beta^2_{\cdot n}, \ldots, \beta^F_{\cdot n}\right), \tag{5.4}$$

and the design matrix $X \in [0,1]^{T \times NF}$ is defined according to

$$X = \left[G^1 Y^T \; G^2 Y^T \; \cdots \; G^F Y^T\right]. \tag{5.5}$$

As such, modeling Y amounts to fitting N distinct logistic regressions—one for the activity $Y_{n\cdot}$ of each neuron. Note that because the design matrices X for each neuron are derived from the same spiking history Y, all of these regressions use an identical[*] design matrix X as defined in (5.5) . If one were only interested in doing inference via maximum likelihood estimation, fitting the model would thus reduce to N separate logistic regressions. Each could be optimized via iteratively re-weighted least squares in parallel.

However, recall from Section 5.1 that N may be very large—representing hundreds or even thousands of neurons, and the corresponding logistic regressions are thus very high-dimensional with a high degree of uncertainty. For this reason, we use Bayesian inference to both quantify this uncertainty in the estimator and promote additional structure in the coefficients (e.g. sparsity and network dependence). Doing so requires summarizing the full the posterior distribution of regression coefficients β, which is a very computationally intensive task for each regression when or N or T is large. Moreover, hierarchical priors that promote structure (e.g. Linderman et al. (2016); Vinci et al. (2018); Wu et al. (2019)) in the regression coefficients

[*]Note that if any neuron-specific effects or extrinsic covariates are appended to X, the design matrices will no longer be identical across all N regressions.

can induce dependence between the N separate regressions, further increasing the computational burden by requiring that the uncertainty in the coefficients of the N regressions be quantified simultaneously.

We are therefore motivated to develop as efficient and scalable an algorithm as possible for computing the posterior distribution of the regression coefficients β. In particular, we are interested in scaling to the case where the number of neurons N, and therefore the number of coefficients in the each logistic regression, is very large. For this task, we rely on the zigzag process (Bierkens et al., 2019a), a new family of posterior sampling algorithms that are known to perform particularly well for efficiently computing posterior distributions associated with logistic regression.

5.2.3 The Zigzag Process

In this section, we describe the key details of the zigzag processes for Bayesian computation. We begin with a brief non-technical overview of what zigzag processes are and what sorts of problems they are good at solving. This is followed by a more detailed explanation of how their implementations work. We use a notation that emphasizes our application to logistic regression on spike trains in Section 5.3. For additional discussion of the zigzag process for logistic regression, we recommend Bierkens et al. (2019a) and Sen et al. (2019).

Zigzag processes (Bierkens et al., 2019a) are a recently proposed family of continuous time Markov processes designed to target a desired distribution as their stationary measure. As depicted in Figure 5.2, a zigzag process explores the distribution using a continuous path of piece-wise linear segments, each segment ending by reversing direction along a dimension. These reversals—which we refer to as bounces—occur stochastically according a carefully defined inhomogeneous Poisson process. This ensures the path is likely to stay in regions of high probability. Simulating a zigzag process trajectory is thus a powerful tool for computing a posterior distribution in Bayesian inference (Bierkens and Duncan, 2017; Bierkens et al., 2019b).

Typically, the main computational bottleneck of simulating a zigzag process is generating the bounce times. For this reason, zigzag processes tend to be especially well-suited for exploring log densities that decompose into summands with bounded gradients—these are properties that allow for fast bounce time generation via Poisson thinning and superposition (Kingman, 1992). Zigzag processes have thus been shown to outperform traditional posterior computation algorithms on variety of relevant statistical problems (e.g. Bierkens et al. (2020b); Koskela (2020)) with high-dimensional logistic regression standing out as an especially effective example (Bierkens et al., 2019a; Sen et al., 2019; Sachs et al., 2020). In Section 5.3, we continue this tradition by developing a technique for especially fast bounce time simulation for spike train logistic regression.

Let us now delve into the specifics of the zigzag process. Let $\beta \in \mathbb{R}^d$ denote a continuous random variable with a differentiable probability density denoted $\pi(\beta)$. A zigzag process targeting $\pi(\beta)$ consists of two components: a position variable $\beta(\tau) \in \mathbb{R}^d$ and a velocity variable $\theta(\tau) \in \{-1, 1\}^d$. Here, τ denotes the artificial[†] time index along which the zigzag process trajectory evolves. Specifically, $\{\beta(\tau), \theta(\tau)\}_{\tau > 0}$ defines a continuous time Markov process with the position variable $\beta(\tau)$ moving through $\mathbb{R}^d$ in a series of linear trajectories, its velocity dictated by $\theta(\tau)$. Each entry in $\theta(\tau)$ occasionally swaps its sign as a random state-dependent event, thus causing $\beta(\tau)$ to reverse its direction along that coordinate. These stochastic sign-swapping events are the "bounces".

The following notation is useful for describing a bounce. For $i \in [d]$, let $S_i : \{-1, 1\}^d \to \{-1, 1\}^d$ denote a function that swaps the sign of the ith coordinate. That is, for all $\theta \in \{-1, 1\}^d$ and $i, j \in [d]$, define

$$(S_i(\theta))_j = \begin{cases} \theta_j & \text{if } i \neq j \\ -\theta_j & \text{if } i = j \end{cases} \tag{5.6}$$

The evolution of a zigzag process trajectory thus consists of two types of updates:

1. The deterministic linear updates of the positions $\beta(\tau)$ according to the velocity $\theta(\tau)$, and
2. The stochastic bounce updates that occasionally swaps the sign of an entry in velocity $\theta(\tau)$.

Figure 5.2 illustrates a short example trajectory for a two dimensional zigzag process. Here, the position variables $\beta(\tau)$ evolve piece-wise linearly in τ for a stochastic amount of time until a bounce occurs. When a bounce occurs, the trajectory of $\beta(\tau)$ reverses direction along the bounce dimension with the trajectory along the other dimensions remains unchanged. That is, $\theta(\tau) \to S_i(\theta(\tau))$ where i denotes the bounce dimension.

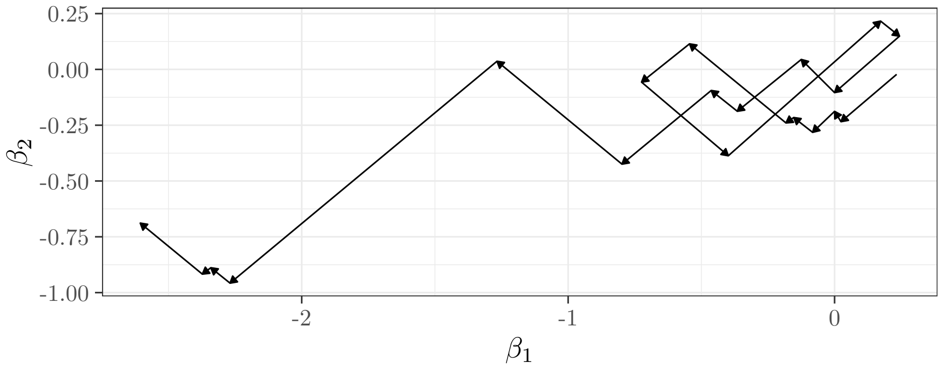


Figure 5.2: An illustration of the piece-wise linear trajectory of a zigzag process with a two-dimensional position variable β. Arrowheads indicate the direction of each piece-wise linear component as it progresses along with τ.

To ensure that a zigzag process targets the appropriate stationary distribution $\pi(\beta)$, the bounce rates must steer the trajectory towards regions of high probability. Doing so requires that zigzag bounce rate

[†]The index τ represents how long the chain is being run, serving as the continuous analog of the number of draws in traditional discrete time Markov chain Monte Carlo.

be a function of its current state (i.e. position $\beta(\tau)$ and velocity $\theta(\tau)$) of the process. Accordingly, we use $\lambda(\beta,\theta) : \mathbb{R}^d \times \{-1,1\}^d \to [0,\infty)^d$ to denote the vector of instantaneous bounce rates, with $\lambda_i(\beta,\theta)$ denoting the bounce rate for the ith coordinate. Bierkens and Duncan (2017) and Bierkens et al. (2019b) showed that for probability densities $\pi(\beta)$ that are strictly positive and continuously differentiable, the zigzag process will be ergodic with respect to π if and only if $\lambda(\beta,\theta)$ satisfies a condition involving the local gradient of the log posterior. Soon, we will get into the details of how to define such a $\lambda(\beta,\theta)$ in (5.11). For concreteness, we first outline how to simulate a zigzag process for a given $\lambda(\beta,\theta)$.

Recall that the velocity $\theta(\tau)$ remains fixed between bounces, so $\beta(\tau)$'s trajectory evolves deterministically according to $\theta(\tau)$. An entire zigzag trajectory $\{\beta(\tau),\theta(\tau)\}_{\tau>0}$ can thus be reconstructed from just its initial starting point $(\beta(0),\theta(0)) \in \mathbb{R}^d \times \{-1,1\}^d$, its bounce times $(\tau_j \in \mathbb{R}_+)_{j=1,\ldots}$, and the states $(\beta(\tau_j),\theta(\tau_j))$ immediately following each bounce. The remainder of the trajectory is a simple linear interpolation of these "skeleton points".

When simulating a zigzag process, we need only concentrate on generating these skeleton points. The following steps determine the time τ_{j+1} and state $(\beta(\tau_{j+1}),\theta(\tau_{j+1}))$ of the next bounce, given the current state $(\beta(\tau_j),\theta(\tau_j))$ and time τ_j:

- For each $i \in [d]$, generate a bounce time t_i distributed according to

$$\mathbb{P}(t_i > \tau) = \exp\left(-\int_{\tau_j}^{\tau} \lambda_i(\beta(\tau_j) + \theta(\tau_j)(s-\tau_j), \theta(\tau_j))\mathrm{d}s\right) \tag{5.7}$$

- Determine the coordinate $i_0 = \mathrm{argmin}_{i\in[d]}(t_i)$ that bounces first.

- The state $(\beta(\tau_{j+1}),\theta(\tau_{j+1}))$ and time τ_{j+1} of the next bounce are:

$$\tau_{j+1} = t_{i_0}, \tag{5.8}$$

$$\beta(\tau_{j+1}) = \beta(\tau_j) + \theta(\tau_{j+1})(\tau_{j+1} - \tau_j), \text{ and} \tag{5.9}$$

$$\theta(\tau_{j+1}) = S_{i_0}(\theta(\tau_j)) \tag{5.10}$$

Repeatedly iterating this process J times effectively simulates a zigzag trajectory with duration $\sum_{j=1}^{J} \tau_j$. Rather than simulate the process for a pre-specified number of bounces, we choose to simulate for a pre-specified duration, then thin it into a discrete Markov chain by determining its state across a set of equally spaced times $\mathcal{T}$ (e.g. $\mathcal{T} = \{\omega i : i \in [1000]\}$ for some $\omega \in \mathbb{R}_+$). This approach is analogous to the traditional, well-studied case of running a Markov chain Monte Carlo algorithm for a pre-specified number of iterations. We can thus use the existing sweet of tools for discrete time Markov chains when examining the results of the empirical studies in Section 5.4.

Notice that the only non-trivial step of simulating a zigzag process is the generation of the bounce times for each coordinate. As such, bounce time generation represents the sole computational bottleneck of simulating a zigzag trajectory. In Section 5.3.3, we describe an efficient strategy for generating bounce times for the logistic regression spike train model. Before presenting this strategy, however, we first define the bounce rate λ in a way that ensures $\pi(\beta)$ is targeted. The following is a summarization of the results of Bierkens et al. (2019a) to that effect.

Under regularity conditions[‡], a zigzag process with bounce rate $\lambda(\beta, \theta)$ targets the density $\pi(\beta)$ if and only if

$$\lambda_i(\beta, \theta) = \left[-\theta_i \frac{\partial \log(\pi(\beta))}{\partial \beta_i}\right]_+ + \gamma_i(\beta, \theta) \tag{5.11}$$

where the function $\gamma : \mathbb{R}^d \times \{-1, 1\}^d \to [0, \infty)$ is defined such that $\gamma_i(\beta, \theta) = \gamma_i(\beta, S_i(\theta))$ for all $\beta, \in \mathbb{R}^d$, $\theta \in \{-1, 1\}^d$, and $i \in [d]$, and $[x]_+ = \max(0, x)$ denotes a function that returns x if x is positive, 0 otherwise.

The expression (5.11) demonstrates that, in order to target a desired density, the instantaneous bounce rate of the zigzag process must be closely tied to the gradient of the logarithm that density—the additional summand γ providing some flexibility. Given that zigzag processes with higher bounce rates tend to explore the posterior distributions less efficiently (Bierkens et al., 2019a), the theoretically optimal choice for λ would thus be to minimize the effective bounce rate by setting $\gamma(\beta, \theta) = 0$ for all (β, θ). In practice, choosing an appropriate λ is not so clear cut. Because generating bounce times is the sole computational bottleneck of simulating a zigzag process, the computational efficiency of a zigzag process relies heavily on how rapidly the bounce times for each coordinate can be generated. It can thus be preferable to choose a bounce rate λ that corresponds to a nonzero value of γ, provided that is amenable to fast simulation of bounce times.

For non-trivial bounce rates (i.e. those for which the inverse cdf method (Devroye, 1986) is impractical), a common approach to generating the bounce times $(\tau_i)_{i \in [d]}$ is to think of them as point processes. Specifically, we can recognize that zigzag bounce times as defined in (5.7) are essentially first arrival times of temporal Poisson processes with rates $\lambda_i'(s) = \lambda_i(\beta(\tau_j) + \theta(\tau_j)(s - \tau_j), \theta(\tau_j))$. Framing the problem this way opens the door to leverage known techniques for generating Poisson processes such as super-position and thinning (Kingman, 1992), thus allowing for the generation of bounce times without necessarily having to calculate the full gradient of the posterior distribution (Bouchard-Côté et al., 2018; Bierkens et al., 2019a). Section 5.3 now describes such an approach specialized for neural spike train regression.

[‡]The statement of this result in Bierkens et al. (2019a) supposes that $\log(\pi(\beta))$ is a continuously differentiable function. In Section 5.3, we rely on a slightly more general formulation that allows for $\log(p(\beta))$ to piece-wise differentiable, provided that is continuous everywhere. Justification to support this generalization is provided in the proof of Proposition 1 in Koskela (2020).

5.3 A Zigzag Process for Neural Spike Trains

Having established the necessary background info regarding both spike train logistic regressions and zigzag processes, we can outline our new framework for efficient posterior computation of spike train logistic regressions via the zigzag process. The novelty of our approach stems from two main contributions.

First, we carefully design the zigzag bounce rates λ such that we can exploit the imbalance in the response Y and structured sparsity in the design matrix X to efficiently simulate the bounce times via Poisson thinning and superposition. Our approach is an extension of the approach proposed in Sen et al. (2019), specialized to leverage some additional structure in spike train logistic regressions. We separate our description into two separate parts: our definition of the bounce rate λ is defined in Section 5.3.1, and our strategy for simulating these bounce rates is outlined in Section 5.3.3.

Second, we describe a strategy with which the zigzag process can be efficiently combined with a spike-and-slab prior to promote sparsity the regression coefficients β. This approach leads to more efficient posterior inference whilst simultaneously encoding prior information that the coupling network of neurons within and across regions of the brain is typically sparse. We efficiently accommodate a spike-and-slab prior in a zigzag process—without introducing additional discrete variables—through a novel parametrization of the spike-and-slab prior that suggests a variant of the Poisson process. It amounts to temporarily changing the velocity $\theta_i(\tau)$ to 0—temporarily halting its trajectory—for a pre-specified amount of time each time $\beta_i(\tau)$ crosses the origin. We outlined this approach in Section 5.3.3.

5.3.1 Defining the Zigzag Bounce Rate

In Section 5.2.3, we described how a zigzag process with bounce rate λ will target a distribution $\pi(\beta)$ provided that condition (5.11) holds. Here, we define a specific bounce rate λ that satisfies (5.11) to target the posterior associated with the spike train regression framework outlined in Section 5.2.2. Our particular choice is made to promote efficient bounce time simulation.

Recall that Bayesian inference for the modeling task outlined in Section 5.2.2 requires computing the posterior distribution of N sets of regression coefficients—a separate logistic regression for each neuron being recorded. However, if the prior distribution on β assumes independence between regression coefficients for the spiking of distinct neurons, these N tasks can be carried out independently of each other. Going forward, we assume a prior distribution on β that satisfies this independence criterion. We can thus focus our exposition on the problem of fitting a single linear regression, understanding that this same procedure applies to all N regressions. We defer our remarks on how this approach can be extended to allow dependency in the prior on β to Section 5.5.

Now having reduced our problem to computing the posterior associated with a single regression task, we can further simplify the notation introduced in Section 5.2.2. Without loss of generality, we focus on the logistic regression associated with spiking of neuron $n = 1$. We let $Y \in \{0,1\}^N$ denote corresponding the response variable in the logistic regression (originally denoted $Y_{1\cdot}$ in Section 5.2.2), $\beta \in \mathbb{R}^{NF+1}$ denote the coefficient vector defined in(5.4) concatenated with the intercept term b_1, and $X \in [0,1]^{T\times(NF+1)}$ be the matrix defined in (5.5) with a column of ones added to the end. This intercept concatenation is standard in regression tasks—it allows for concise representation of both the log posterior and its gradients.

The log likelihood $\ell(\beta|Y,X)$ associated with our target posterior distribution can be expressed as

$$\ell(\beta \mid Y, X) = \sum_{t=1}^{T} \left(Y_t \left(X_{t\cdot}\beta \right) - \log\left(1 + e^{X_{t\cdot}\beta}\right)\right). \tag{5.12}$$

Assuming an independent prior $\pi_0(\beta)$ on β that decomposes entry-wise into the product of densities $\pi_0^1(\beta_1), \ldots, \pi_0^{NF+1}(\beta_{NF+1})$, the log posterior $\log(\pi(\beta \mid Y,X))$ is thus equal to

$$\log\left(\pi(\beta \mid Y, X)\right) = \sum_{t=1}^{T} \left(Y_t X_{t\cdot}\beta - \log\left(1 + e^{X_{t\cdot}\beta}\right)\right) + \sum_{j=1}^{NF+1} \log(\pi_0^j(\beta_j)) \tag{5.13}$$

$$\tag{5.14}$$

up to an additive constant, and the gradient of the posterior is given by

$$\begin{aligned} \frac{\partial \log\left(\pi(\beta \mid Y, X)\right)}{\partial \beta_j} &= \sum_{t=1}^{T} X_{tj} \left(Y_t - \left(1 + e^{-X_{t\cdot}\beta}\right)^{-1}\right) + \frac{1}{\pi_0^j\left(\beta_j\right)} \frac{\partial \pi_0^j(\beta_j)}{\partial \beta_j} && (5.15) \\ &= \sum_{t=1}^{T} X_{tj} \left(Y_t - \mathbb{P}(Y_t = 1 | X_t, \beta)\right) + \frac{1}{\pi_0^j\left(\beta_j\right)} \frac{\partial \pi_0^j(\beta_j)}{\partial \beta_j} && (5.16) \end{aligned}$$

for all $j \in [NF]$.

We choose our bounce rate λ_i for each $i \in [NF+1]$ to be

$$\begin{aligned} \lambda_i(\beta, \theta) &= \sum_{t=1}^{T} \left[-\theta_i \frac{\partial \log\left(\mathbb{P}(Y_t \mid \beta, X_{t\cdot})\right)}{\partial \beta_i} \right]_+ + \left[\frac{-\theta_i}{\pi_0^i\left(\beta_i\right)} \frac{\partial \pi_0^i(\beta_i)}{\partial \beta_i} \right]_+ && (5.17) \\ &= \sum_{t=1}^{T} \left[-\theta_i X_{tj} \left(Y_t - \mathbb{P}(Y_t = 1 | X_t, \beta)\right)\right]_+ + \left[\frac{-\theta_i}{\pi_0^i\left(\beta_i\right)} \frac{\partial \pi_0^i(\beta_i)}{\partial \beta_i} \right]_+ . && (5.18) \end{aligned}$$

where $[x]_+ = \max(0,x)$. In addition to satisfying (5.11), this choice is particularly amenable to efficient generation of bounce times (Sen et al., 2019). We will describe an efficient algorithm for generating bounce times in Section 5.3.3. First, however, we must prescribe our spike-and-slab prior for β along with a non-standard parametrization that allows the zigzag process to be applied in its otherwise discrete setting.

5.3.2 Spike-and-slab prior

Sparsity of the regression coefficients is a popular assumption in neural spike train regression (e.g. Park and Pillow (2011), Linderman et al. (2016), Zoltowski and Pillow (2018), Vinci et al. (2018), Wu et al. (2019)) and regression problems in general (Tibshirani, 1996; Park and Casella, 2008; Ročková and George, 2018). Sparsity naturally arises in neuroscience in the form of receptive fields, biological neuron connectivity, and neural network structure. Even in cases where the vector of coefficients is not known be sparse, it can still be beneficial to "bet on sparsity" in high-dimensional settings (Hastie et al., 2009, Sec. 16.2.2) because estimation in the non-sparse situation is intractable. Here, we detail a prior structure that promotes sparsity in β for the spike train logistic regression problem outlined in Sections 5.2.2 and 5.3.1.

We consider a prior π_0 on β that decomposes into independent spike-and-slab priors (Mitchell and Beauchamp, 1988) for each entry in β. Spike-and-slab priors consist of a point mass at 0 (i.e. the *spike*) and a diffuse continuous density that is symmetric around 0 (i.e. the *slab*). We define a canonical parametrization of π_0 in Section 5.3.2, then develop a re-parametrization in Section 5.3.2 that converts it into a continuous and piece-wise differentiable distribution to accommodate inference via the zigzag process.

Defining the Prior

Recall from Section 5.3.1 that we are interested in a prior $\pi_0(\beta)$ that decomposes into independent priors $\pi_0^1, \dots, \pi_0^{NF+1}$ for each of entries β_i, $i \in [NF+1]$. Here, we define such a spike-and-slab prior with two hyperparameters for each coefficient: $p_i \in [0,1]$ to prescribe the prior probability that β_i is nonzero (i.e. the slab probability), and $\alpha_i \in \mathbb{R}_+$ to control how diffuse the slab is around the origin. Together, each entry in the prior represents a two component mixture

$$\pi_0^i(\beta_i) := \sim \begin{cases} \beta_i \sim \text{Laplace}(\alpha_i) & \text{with probability } p_i \\ \beta_i = 0 & \text{with probability } 1 - p_i \end{cases} \tag{5.19}$$

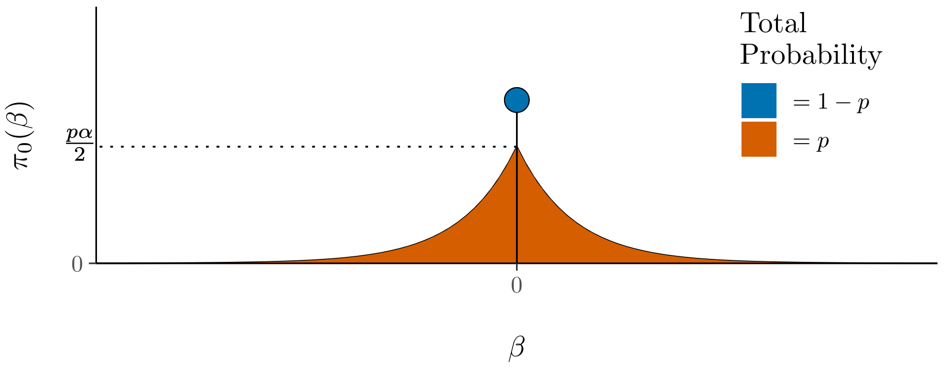
where Laplace(α_i) denotes a distribution with probability density function

$$f(\beta) = 2^{-1}\alpha_i e^{-\alpha_i|\beta|}. \tag{5.20}$$

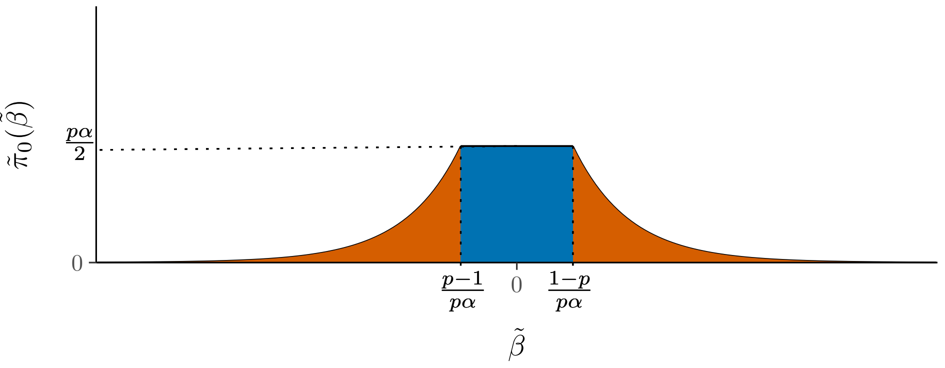
Figure 5.3(a) depicts $\pi_0^i(\beta_i)$ as a mixed probability mass function/probability density function representation.

As shown in Figure 5.3(a), the atom (i.e. spike) in the distribution at 0 ensures a prior probability of $1-p_i$ that β_i is zero, whereas the Laplace slab component ensures positive prior density over the entire real line. For a finite number of time bins T, any positive prior probability that $\beta_i = 0$ be pushed to the posterior, yielding sparsity in the posterior. The strength of sparsity will depend both on the hyperparameters $(p_i)_{i\in[NF+1]}$ and the observed spiking Y.

The maximum of the density component of π_0^i is $p_i 2^{-1}\alpha_i$. The smaller the value of α_i, the more diffuse the slab component of the distribution. As such, this prior provides an intuitive way to control over both the expected sparsity in the coefficients, as well as the expected magnitude of the nonzero coefficients using the two hyperparameters.



(a)



(b)

Figure 5.3: (a) depicts the canonical parametrization of the spike-and-slab prior π_0 as defined in (5.19). The continuous density component (the slab) of π_0 is is shown in orange. The discrete point mass at 0 (the spike) in blue. (b) depicts the reparametrized prior $\tilde{\pi}_0$ on the variable $\tilde{\beta}$. Following (a), the portion of the density corresponding to the slab component is shown in orange, and the uniform distribution encoding the original spike component is shown in blue.

Despite their providing a simple, intuitive way to encode sparsity, spike-and-slab priors are rarely used in fully Bayesian inferences of spike train regression outside of relatively low dimensional problems. The reason for this avoidance is primarily computational—posterior inference on spike-and-slab posterior distributions is unwieldy due to the effective dimension of the posterior varying along with the number of nonzero coefficients.

The standard computation approach to accommodate such irregular posteriors is to use binary auxiliary variables to indicate which coefficients are nonzero, then structure the problem in a way that facilitates either collapsed Gibbs updates (as in Linderman et al. (2016)) or reversible jump Markov chain Monte Carlo

(Green, 1995). Nevertheless, the introduction of these auxiliary variables typically leads to slow mixing of the chain in modern high dimensional problems. For this reason, there has been a recent push in the sparse Bayesian regression literature towards continuous relaxations of spike-and-slab, such as the horseshoe prior (Carvalho et al., 2009) and its variants (Johndrow et al., 2017; Piironen et al., 2017; Bhadra et al., 2017, 2019).

Such continuous relaxations lead to continuous posterior densities for which computation tends to be simpler and more reliable, mainly due to their being accommodating of modern gradient-based posterior inference methods such as Hamiltonian Monte Carlo (Neal, 2011). However, continuous relaxations also have a cost—the continuous prior density means that the posterior is no longer sparse. Many posterior draws will have coefficients that approximately 0, but no coefficient will be exactly 0. These posteriors are thus more difficult to summarize and interpret. Moreover, using a Markov chain Monte Carlo strategy to repeatedly update coefficients that are essentially zero seems computationally wasteful as compared to keeping them fixed at exactly 0 using spike-and-slab.

With this in mind, we have devised a way to get the best of both worlds via the zigzag process. Using a novel parametrization, we can formulate the spike-and-slab Bayesian inference problem without relying on auxiliary variables, allowing the zigzag process to rapidly mix.

New parametrization

Recall that zigzag process as outlined in Section 5.2.3 applies only to densities that are continuous and piece-wise differentiable. Because the canonical representation of our spike-and-slab prior has a discrete component, the corresponding posterior distribution is not amenable to inference via the zigzag process. Here, we develop a novel continuous parametrization of the spike-and-slab that is compatible with zigzag process inference. As we will soon show, we do not even need to explicitly apply this re-parametrization to reap its rewards—a slight modification of how the zigzag process behaves in the canonical parametrization will suffice.

Now, we outline our novel parametrization. For each coefficient $\beta_i \in \mathbb{R}$, we define a variable $\tilde{\beta}_i$ such that

$$\tilde{\beta}_i \sim \begin{cases} \tilde{\beta}_i = \beta_i - \frac{1-p_i}{\alpha_i p_i} & \text{if } \beta_i < 0 \\ \text{unif}\left(-\frac{1-p_i}{\alpha_i p_i}, \frac{1-p_i}{\alpha_i p_i}\right) & \text{if } \beta_i = 0 \\ \tilde{\beta}_i = \beta_i + \frac{1-p_i}{\alpha_i p_i} & \text{if } \beta_i > 0 \end{cases} \cdot \tag{5.21}$$

where p_i and α_i correspond to the hyperparameters of π_0^i.

The prior π_0^i on β_i in (5.19) thus translates to the prior $\tilde{\pi}_0^i$ on $\tilde{\beta}_i$ defined by

$$\tilde{\pi}_0^i(\tilde{\beta}_i) = \begin{cases} \frac{p_i\theta_i}{2} & |\tilde{\beta}_i| \leq \frac{(1-p_i)}{p_i\alpha_i} \\ \frac{p_i\theta_i}{2} e^{-\theta_i\left(|\tilde{\beta}_i| - \frac{1-p_i}{p_i\theta_i}\right)} & |\tilde{\beta}_i| > \frac{(1-p_i)}{p_i\theta_i} \end{cases}. \tag{5.22}$$

Figure 5.3(b) depicts this new density $\tilde{\pi}_0^i(\tilde{\beta}_i)$, which can be viewed as a continuous parametrization of the spike-and-slab prior π_0^i. The same colors are used in Figures 5.3(a) and 5.3(b) to showcase this connection. The atom at zero has been recast as uniform distribution centered at 0, and the slab has been split in two, translated, then appended to either end of the uniform distribution.

Note that (5.21) implies straightforward relationship between β and $\tilde{\beta}$. That is, the coefficient β_i can be easily recovered from its corresponding $\tilde{\beta}_i$ as $\beta_i = B_i(\tilde{\beta}_i)$, where

$$B(\tilde{\beta}_i) = \begin{cases} \tilde{\beta}_i + \frac{1-p_i}{\alpha_i p_i} & \tilde{\beta}_i < -\frac{1-p_i}{\alpha_i p_i} \\ 0 & |\tilde{\beta}_i| < \frac{1-p_i}{\alpha_i p_i} \\ \tilde{\beta}_i - \frac{1-p_i}{\alpha_i p_i} & \tilde{\beta}_i > \frac{1-p_i}{\alpha_i p_i} \end{cases}. \tag{5.23}$$

We can thus easily frame the problem of posterior inference of β in terms of this new parametrization—as posterior inference of $\tilde{\beta}$. Unlike our original parametrization, this parametrization is amenable to zigzag because it satisfies the continuity and piece-wise differentiability conditions outlined Section 5.2.3. The bounce rates λ defined in (5.17) translate bounce rates $\tilde{\lambda}$ defined as

$$\tilde{\lambda}_i(\tilde{\beta}, \theta) = \sum_{t=1}^{T} \left[-\theta_i \frac{\partial \log\left(\mathbb{P}(Y_t \mid B(\tilde{\beta}), X_{t\cdot})\right)}{\partial \beta_i} \frac{\partial B(\tilde{\beta}_i)}{\partial \tilde{\beta}} \right]_+ + \left[\frac{-\theta_i}{\tilde{\pi}_0^i\left(\tilde{\beta}_i\right)} \frac{\partial \tilde{\pi}_0^i(\tilde{\beta}_i)}{\partial \tilde{\beta}_i} \right]_+ . \tag{5.24}$$

Armed with this newly defined parametrization and bounce rate, it is now possible to define a new zigzag process that targets the posterior distribution $\tilde{\beta}$ directly. However, it turns out that we do not even need to explicitly switch to the new parametrization to use the zigzag process. Notice that the definition of $\tilde{\beta}$ has ensured that for the entire region

$$|\tilde{\beta}_i| < \frac{1-p_i}{\alpha_i p_i}, \quad \text{(i.e. } \beta_i = 0) \tag{5.25}$$

both

$$\frac{\partial \tilde{\pi}_0^i(\tilde{\beta}_i)}{\partial \tilde{\beta}_i} = 0 \text{ and } \frac{\partial B(\tilde{\beta}_i)}{\partial \tilde{\beta}_i} = 0. \tag{5.26}$$

Therefore, the bounce rate $\tilde{\lambda}_i(\tilde{\beta}, \theta) = 0$ for all values of $\tilde{\beta}$ such that $\beta_i = 0$. That is, the reparametrized zigzag trajectory is guaranteed to never bounce while passing through the region defined by (5.25). As shown in blue portion in Figure 5.3(b), this region has width $(1 - p_i)\alpha_i^{-1}p_i^{-1}$—each pass through the "no bounce zone" is guaranteed to last for $(1 - p_i)\alpha_i^{-1}p_i^{-1}$ time units, independently of what occurs in other dimensions.

Translating back to the canonical parametrization, this effect on the trajectory is exactly equivalent to following. For each coordinate $i \in [NF + 1]$, each time the trajectory $\beta(\tau)$ reaches point at which $\beta_i(\tau) = 0$, the position $\beta_i(\tau)$ gets temporarily "stuck" for a duration of $(1 - p_i)\alpha_i^{-1}p_i^{-1}$ time units. Once this "stuck" time has passed, it will resume its trajectory as normal, with direction once again dictated by value of θ_i from before getting stuck. That is, the velocity θ_i will temporarily switch to 0 each time β_i reaches 0, with the switch duration depending on the hyperparameters α_i and p_i of the prior. Intuitively, the larger the spike in the prior, the longer each stop at 0 will be. In the extremes, $p_i = 0$ will guarantee that $\beta_i(\tau)$ is fixed at the origin (all spike), and $p_i = 1$ will result in no stop at all (all slab).

It is thus straightforward to implement the spike-and-slab as part of the zigzag process in the canonical representation. We operate using the slab component of the prior as the entire prior, monitoring when the positions in $\beta(\tau)$ hit 0. When they do, the spike component of the prior causes them to temporarily get stuck. That is, we temporarily switch their velocity to 0 for a duration of $(1 - p_i)\theta_i^{-1}p_i^{-1}$ time units, then allow the velocity to pick back up where it left off when this duration ends. This approach is exactly equivalent to running a zigzag process in the continuous, piece-wise differentiable re-parametrized setting described above. It is thus guaranteed to target the desired stationary distribution due to the results of Bierkens et al. (2019a) and Koskela (2020).

With our strategy for accommodating the spike-and-slab prior defined, we can now return to discussing how to simulate bounce times.

5.3.3 Bounce Time Simulation

In this section, we describe how to apply the principles of Poisson superposition and Poisson thinning to efficiently generates bounce times with the rate λ prescribed (5.17). We also incorporate the concrete details for letting β_i gets "stuck" when crossing zero as described in Section 5.3.2. Our exposition will begin by focusing on bounce time generation, incorporating the "getting stuck" component only when we bring it all together at the end.

Note that each λ_i in (5.17) consists of $T + 1$ non-negative summands—the first T summands correspond to the likelihood, and the final one corresponding to the prior. Recall that generating a bounce time with rate λ_i is equivalent to determining the first arrival time of an inhomogeneous temporal Poisson process with rate λ_i. Invoking the superposition principle of Poisson processes (Kingman, 1992), a Poisson process with rate λ_i can thus expressed as the superposition of $T + 1$ independent Poisson processes with rates

$\lambda_i^1 \ldots, \lambda_i^T, \lambda_i^{T+1}$, defined as

$$\lambda_i^t(\beta,\theta) = \begin{cases} [-\theta_i X_{ti}\,(Y_t - \mathbb{P}(Y_t = 1|X_t,\beta))]_+ & t \in [T] \\ \left[\frac{-\theta_i}{\pi_0^i(\beta_i)}\frac{\partial \pi_0^i(\beta_i)}{\partial \beta_i}\right]_+ = [\alpha_i\theta_i \text{sign}(\beta_i)]_+ & t = T+1 \end{cases}, \tag{5.27}$$

respectively. Therefore, the first arrival time of a process with rate λ_i has the same distribution as the minimum of the first arrival times taken across $T+1$ processes with rates $\lambda_i^1, \ldots, \lambda_i^{T+1}$. We will use this representation to efficiently generate our bounce times.

At first glance, it may seem that invoking the superposition principle has only made the problem harder. Exchanging the generation of the first arrival time of a single Poisson process with that of generating first arrival times for $T+1$ separate Poisson processes? That does not feel like progress! However, this superposition representation ends up being helpful when combined with another common tool for generating from Poisson processes—Poisson thinning.

Poisson thinning is a technique for generating Poisson processes in which one generates a Poisson process with rate λ by first generating a Poisson process with rate $\Lambda \geq \lambda$, then preferentially subsampling the result to compensate for the difference between λ and Λ at the corresponding points. This approach is especially powerful when Λ is both a relatively tight bound on λ, and straightforward to simulate from directly (e.g. a uniform bound corresponds to exponential bounce times).

Poisson thinning allows for the generation of a first arrival time with rate λ via the following rejection sampling procedure. First, we generate a first arrival time τ^* with rate Λ. Then, with probability $\lambda(\tau^*)/\Lambda(\tau^*)$, we accept the value τ^* as the first arrival time. Otherwise, we advance the base start time to τ^* and repeat the procedure until acceptance occurs. This thinning approach is powerful because one need only be able to upper bound the rate λ and evaluate it point-wise in order to simulate a first bounce time from it—no evaluation of the inverse cdf is required.

Returning to the task of simulating the bounce rate λ_i as defined in (5.17), we could apply Poisson thinning directly by upper bounding the bounce rate λ_i. This is accomplished by individually upper bounding each[§] summand $(\lambda_i^t(\beta,\theta))_{t\in[T+1]}$. First, let us upper bound the likelihood components $\lambda_i^1, \ldots, \lambda_i^T$.

For a given $\theta_i \in \{-1, 1\}$, many of these components will be 0. Specifically, $\lambda_i^t(\beta,\theta) = 0$ if any of the following three conditions hold:

1. $X_{ti} = 0$,

2. $Y_t = 1$ and $\theta_i = 1$, or

3. $Y_t = 0$ and $\theta_i = -1$

[§]Recall that if $i \in [NF+1]$ is currently "stuck" at 0 due to the spike-and-slab prior, its bounce rate is guaranteed to be 0 (see Section 5.3.2) and can thus be ignored until it is no longer stuck. We thus presume that the $i \in [1 + NF + 1]$ under consideration is not currently stuck.

Therefore, we need only focus on those $t \in T$ not meeting any criteria above. Noting that $|Y_t - \mathbb{P}(Y_t = 1|X_t, \beta)| \leq 1$ and $X_{ti} > 0$, the remaining entries can be upper bounded by the constant X_{ti}. Moreover, the prior term λ_0^{T+1} is trivially upper bounded by α_{T+1} if $\theta_{T+1} = \text{sign}(\beta_{T+1})$, and is simply 0 otherwise.

Therefore, we have

$$\lambda_i(\beta, \theta) \leq \Lambda_i(\theta) \tag{5.28}$$

$$= \sum_{t \in R_{\theta_i}} X_{ti} \tag{5.29}$$

where

$$R_{\theta_i} = \begin{cases} \{t \in [T] : (Y_t + \theta_i) \in \{0, 1\} \text{ and } X_{ti} > 0\} \cup \{T+1\} & \text{if } \theta_{T+1} = \text{sign}(\beta_{T+1}) \\ \{t \in [T] : (Y_t + \theta_i) \in \{0, 1\} \text{ and } X_{ti} > 0\} & \text{otherwise} \end{cases} \tag{5.30}$$

denotes the set of $t \in [T+1]$ for which $\lambda_i^t(\beta, \theta)$ is not guaranteed to be zero by the three conditions above. To avoid having to deal with the intercept term separately, we use the slight abuse of notation $X_{(T+1)i} = \alpha_i$.

We can now finally demonstrate the value of the superposition principle. Suppose that our zigzag process is currently at the state (β, θ). Directly applying Poisson thinning to generate the first arrival τ time using the upper bound Λ_i would involve a rejection procedure of generating $\tau^* \sim \text{Exp}(\Lambda_i)$ where Exp denotes an exponential distribution with rate Λ_i, then accepting $\tau = \tau^*$ with probability $\lambda_i(\beta + \theta\tau^*)/\Lambda_i(\theta)$. Such a procedure would involve evaluating $\lambda_i^t(\beta + \theta\tau^*)$ for all $t \in R_{\theta_i}$. It is thus computationally intensive to apply this procedure for large T.

Alternatively, we could apply the superposition principle. To do so, we upper bound each $(\lambda_i^t)_{t \in R_{\theta_i}}$ individually with X_{ti}. Generating the first arrival time across the $T+1$ summands is equivalent to generating

$$\tau_1^* \sim \text{Exp}(X_{1i})$$

$$\vdots$$

$$\tau_{T+1}^* \sim \text{Exp}(X_{(T+1)i}),$$

(ignoring those $t \notin R_{\theta_i}$), followed by determining $j = \text{argmin}_{k \in R_{\theta_i}}(\tau_k^*)$, then accepting $\tau = \tau_j^*$ with probability $\lambda_i^j(\beta + \theta\tau_j^*)/X_{ji}$. This procedure requires evaluating at most one λ_i^t for each proposal, making it far more efficient than the non-superposition approach.

We can make this superposition procedure even more efficient by taking advantage of a property of the minimum of exponential distributions. Specifically,

$$\tau_j^* = \min\left(\tau_1^* \sim \mathrm{Exp}(X_{1i}), \ldots, \tau_{T+1}^* \sim \mathrm{Exp}(X_{(T+1)i})\right) \tag{5.31}$$

$$\Rightarrow \tau_j^* \sim \mathrm{Exp}\left(\sum_{t \in R_{\theta_i}} X_{ti}\right) \text{ and} \tag{5.32}$$

$$j \sim \text{Categorical}\left(\frac{X_{1i}}{\sum_{t \in R_{\theta_i}} X_{ti}}, \ldots, \frac{X_{(T+1)i}}{\sum_{t \in R_{\theta_i}} X_{ti}}\right). \tag{5.33}$$

This reduces the problem from drawing $|R_{\theta_i}| + 1$ exponential random variables to that of drawing a single exponential random variable along with a categorical variable.

Generically, generating a single draw from a high-dimensional categorical distribution can be computationally expensive. However, the linear combination structure of X provided in (5.5) makes it straightforward to express the categorical distribution in (5.33) as a mixture of at most $t_{\max}$ discrete uniform distributions, thus allowing for j to be simulated very efficiently.

Moreover, the efficiency gains that can be gleaned from Poisson superposition continue. The categorical sampling approach we apply in (5.33) can actually be extended to finding the minimum for all of $i \in [NF+1]$ at once. Recall from Section 5.2.3 that our procedure for generating the zigzag process depends only on determining $i_0 = \mathrm{argmin}_{i \in [NF+1]}(t_i)$ and t_{i_0} where t_i denotes the first bounce time in coordinate i. Let I denote the set of $i \in [NF+1]$ that are not currently "stuck" at 0 (as described in Section 5.3.2). Now, extending the same exponential trick as above, we obtain

$$t_{i_0} \sim \mathrm{Exp}\left(\sum_{i \in I} \sum_{t \in R_{\theta_i}} X_{ti}\right) \text{ and} \tag{5.34}$$

$$(i_0, t) \sim \text{Categorical}\left(\frac{X_{11}}{\sum_{i \in I} \sum_{t \in R_{\theta_i}} X_{ti}}, \ldots, \frac{X_{(T+1)(NF+1)}}{\sum_{i \in I} \sum_{t \in R_{\theta_i}} X_{ti}}\right). \tag{5.35}$$

Employing Poisson thinning, the draw (i_0, t_{i_0}) is accepted with probability $\lambda_{i_0}^t(\beta + \theta t_{i_0})/X_{ti_0}$, discarded otherwise. Note that here, we have set-up the categorical draw as being over two separate indices: i_0 represents the index $i \in [NF+1]$ of the coefficient being considered, and $t \in [T+1]$ denotes the index of a single time bin. This two-index procedure ensures that only a single λ_i^t needs to be evaluated.

We can efficiently make the categorical draw of (i_0, t) according to the following nested process:

$$i_0 \sim \text{Categorical}\left(\frac{\sum_{t\in R_{\theta_1}} X_{t1}}{\sum_{i\in I}\sum_{t\in R_{\theta_i}} X_{ti}}, \ldots, \frac{\sum_{t\in R_{\theta_{NF+1}}} X_{t(NF+1)}}{\sum_{i\in I}\sum_{t\in R_{\theta_i}} X_{ti}}\right) \tag{5.36}$$

$$t \mid i_0 \sim \text{Categorical}\left(\frac{X_{1i_0}}{\sum_{t\in R_{\theta_{i_0}}} X_{ti}}, \ldots, \frac{X_{(T+1)i_0}}{\sum_{t\in R_{\theta_{i_0}}} X_{ti}}\right) \tag{5.37}$$

This set-up is efficient because the sums $\sum_{t\in R_{\theta_i}} X_{t1}$ for each (i, θ_i) pair need only be computed once for the entire run of the algorithm, and the conditional draw of t allows the efficient mixture of discrete uniforms approach to be applied.

Combining this approach with an implementation of the "stickiness" of $\beta_i(\tau)$ at 0 due to the spike-and-slab prior, yields the following algorithm.

Suppose we are currently at state $(\beta(\tau_j), \theta(\tau_j))$ and time τ_j. Let $I \subset [NF+1]$ denote the set of coefficients that are currently not stuck at 0, and I^C denote the set of coefficients are stuck at 0. For each $i \in I$, we define R_{θ_i} as above. Then, the next bounce time τ_{j+1} and state $(\beta(\tau_{j+1}), \theta(\tau_{j+1}))$ of the next bounce can be obtained as follows.

1. Initialize $\tau^* = \tau_j$, $\beta(\tau^*) = \beta(\tau_j)$, and $\theta(\tau^*) = \theta(\tau_j)$.
2. Iterate the following steps:
   (a) For each $i \in I^C$, let u_i denote the time at which β_i is scheduled to become unstuck.
   (b) For each $i \in I$, let s_i denote the time at which β_i is scheduled to reach 0 along its current trajectory. Specifically, $s_i = |\beta_i|$ if $\theta_i \neq \text{sign}(\beta_i)$, $s_i = \infty$ otherwise.
   (c) Determine $i_1 = \text{argmin}_{i\in I^C}(u_i)$ and $i_2 = \text{argmin}_{i\in I}(s_i)$.
   (d) Generate $t^b \sim \text{Exp}\left(\sum_{i\in I}\sum_{t\in R_{\theta_i}} X_{ti}\right)$
   (e) If($u_{i_1} < s_{i_2}$ and $u_{i_1} < t^b$):
      - Append i_1 to I and remove it from I^C.
      - Increment $\tau^* \leftarrow \tau^* + u_{i_1}$ and $\beta(\tau^*) \leftarrow \beta(\tau^*) + \theta(\tau^*) \cdot u_{i_1}$,
      - Reset $\theta_{i_1}(\tau^*)$ to its unstuck value,
      - Return to Step (a).
   (f) If($s_{i_2} < u_{i_1}$ and $s_{i_2} < t^b$):
      - Remove i_2 from I and append it to I^C.
      - Increment $\tau^* \leftarrow \tau^* + s_{i_2}$ and $\beta(\tau^*) \leftarrow \beta(\tau^*) + \theta(\tau^*) \cdot s_{i_2}$,
      - Temporarily set $\theta_{i_2} = 0$,

- Return to Step (a).

(g) Otherwise ($t^b < u_{i_1}$ and $t^b < s_{i_2}$)

- Generate $i_0 \in I^C$ such that $\mathbb{P}(i_0 = i) \propto \sum_{t \in R_{\theta_i}} X_{ti}$.
- Generate $t \in R_{\theta_{i_0}}$ such that $\mathbb{P}(t = a) \propto X_{ai_0}$.
- If $t = T + 1$, proceed immediately to Step 3.
- If $\theta_{i_0} = -1$, proceed immediately to Step 3 with probability

$$\mathbb{P}(Y_t = 0 | X_t, \beta = \beta_{\tau^*} + \theta(\tau^*) t^b). \tag{5.38}$$

- If $\theta_{i_0} = 1$, proceed immediately to Step 3 with probability

$$\mathbb{P}(Y_t = 1 | X_t, \beta = \beta_{\tau^*} + \theta(\tau^*) t^b). \tag{5.39}$$

(h) Increment $\tau^* \leftarrow \tau^* + t^b$ and $\beta(\tau^*) \leftarrow \beta(\tau^*) + \theta(\tau^*) \cdot t^b$

(i) Return to Step (a).

3. The new bounce time and state are defined as:

- $\tau_{j+1} = \tau^* + t^b$,
- $\beta(\tau_j) = \beta(\tau^*) + \theta(\tau^*) \cdot t^b$,
- $\theta(\tau_j) = S_{i_0}(\theta(\tau^*))$.

Implicit in steps outlined above is the requirement to track how long each $i \in I^C$ will be stuck in 0, and save a reference to what velocity θ_i will return to once β_i is unstuck. This task can be efficiently in practice using an assortment of linked lists that track the states of each i, thus avoiding the need to re-determine i_1 and i_2 at every iteration. Moreover, the values R_{θ_i} and $\sum_{t \in R_{\theta_i}} X_{t1}$ can be precomputed allowing for an efficient update of the categorical probabilities each time θ_i swaps its sign. We have implemented this procedure in C++ code as part of our R package `BrainNetworks`.

With the details of our bounce procedure described, we will now conduct a series of simulations on synthetic data to determine the relative performance of our algorithm relative to that of a competitor (Linderman et al., 2016).

5.4 Efficiency Analysis

The dimensions of spike train logistic regression problems can vary widely depending on brain area, animal being studied, and recording technology used. Historically, only a small number of neurons N could be recorded at once, meaning the duration of the spike train (that is, the number of spike bins T and number of experimental trials) was the primary indicator of the computational burden when fitting models. However, new neural recording technologies (e.g. Jun et al. (2017)) have rapidly changed this paradigm. Thousands of spike trains across multiple areas of the brain can now be recorded simultaneously. As such, the number of neurons N can be computational bottleneck. As such, it is important we have computational tools that can handle both large N and large T. In this section, we assess how our zigzag algorithm performs in the large N and large T regimes.

5.4.1 How to Analyze Efficiency?

When assessing the efficiency of our algorithm, it is natural to use an existing algorithm from literature as a baseline. When it comes to fully Bayesian inference for the spike-and-slab logistic regression problem outlined in Section 5.2.2, the Markov chain Monte Carlo algorithm proposed by Linderman et al. (2016) is the current state of the art. This algorithm introduces two sets of auxiliary variables in order to facilitate closed-form Gibbs updates of the regression coefficients. Specifically, T Polya-Gamma (Polson et al., 2013; Pillow and Scott, 2012) auxiliary variables (one for each spike bin) are instantiated to bypass the difficulties of working directly with the logistic link function, and $NF + 1$ binary spike/slab indicators are instantiated to accommodate the spike-and-slab prior.

A full sweep of this MCMC algorithm involves Gibbs updates of the Polya-Gamma variables, the binary spike/slab indicators, and the coefficient vector β. In total, this amounts to computational complexity of $\Theta((NF)^3+NFT)$ per iteration—the $(NF)^3$ term stemming from the Gibbs updates of the NF β coefficients and binary indicators, the (NFT) term stems from the updates of the T Polya-Gamma variables. In terms of complexity, this algorithm seems better suited for the traditional setting of small N large T than modern applications in which N is large.

In contrast, the computational complexity of simulating our zigzag process for a fixed trajectory length τ is roughly $\Theta((NF)^2T_1)$. Here, T_1 denotes the number of time bins for which $Y_t = 1$, rather than the full number of time bins T. The approximate complexity $(NF)^2T_1$ stems from the following argument. For each proposed bounce time, deciding whether to accept requires evaluating $\mathbb{P}(Y_t = 1|X_{t\cdot}, \beta)$—this has complexity $\Theta(NF)$. The number of proposed zigzag bounces occurring within a fixed time interval τ is proportional to the sum of the upper bounds Λ_i on bounce rates across all dimensions, i.e. $\sum_{i=1}^{NF+1} \Lambda_i(\theta)$. By definition of $\Lambda_i(\theta)$ in (5.28), we see that this sum is roughly proportional to $\|X\|_1$. The definition of X in (5.5) indicates the rough proportionality $\|X\|_1 \propto T_1NF$, provided that $t_{\max}$ is small compared to T. Taking the product of the approximate number of bounces and the approximate complexity per bounce yields the rough approximation $\Theta((NF)^2T_1)$.

Comparing these two complexities—$\Theta((NF)^3 + NFT)$ for Polya-Gamma versus $\Theta((NF)^2 T_1)$ for zigzag, suggests that the zigzag algorithm will perform comparatively better when NF—the number of regression coefficients—is relatively large. Moreover, we would also expect better performance when T_1 is relatively small compared to T (i.e. spikes are a rare occurrence). Both of these situations are common in modern neural datasets.

Though the above complexities provide a compelling heuristic with which to compare the two algorithms, they do not amount to any sort of formal proof of efficiency. This is because the raw number of iterations generated is a proof metric with which to judge a MCMC algorithm. Instead, we should be more concerned with the number of effectively *independent* iterations that can generated in a given amount of time. That is, a MCMC algorithm is not necessarily good just because it is fast—it must also produce chains that mix quickly. As such, a more appropriate metric with which to judge MCMC algorithms is the *effective sample size* (Kass et al., 1998; Ripley, 2009) per second (Section 4.4.1 in Chapter 4 provides a detailed definition treatment of the effective sample size as a metric for MCMC performance).

Unfortunately, outside of simple algorithms applied to toy problems, it is typically impossible to analytically determine the effective sample size per second a given algorithm. For this reason, the standard practice is to compare algorithms by running empirical studies. In Section 5.4.2, we run a battery of synthetic experiments to compare the effective sample size per second between the two algorithms.

Before proceeding with these experiments, it is worth making one additional note about the efficiency of the Polya-Gamma augmentation. MCMC algorithms based on the Polya-Gamma augmentation have been shown mix poorly when there is a large class imbalance in the response variable Y (i.e. T_1/T is small) (Johndrow et al., 2018). Since class imbalance is quite common in neural datasets, we specifically design some experiments in Section 5.4.2 to investigate the relative performance of zigzag under class imbalance.

5.4.2 Synthetic Experiments

In this section, we compare the performance (as measured the effective sample size per second) of the zigzag process implementation outlined in Section 5.3 to that of the Polya-Gamma augmentation of Linderman et al. (2016) for synthetic datasets. Specifically, we use both algorithms to fit spike train logistic regressions to 36 different synthetically generated datasets encompassing a variety of values of T, N, and response imbalance (i.e. T_1/T). For each dataset, we run the zigzag process for a trajectory length of $\tau = 5000$, obtaining a Markov chain with 50000 observations by considering the discretization associated with $\mathcal{T} = \{0.1k : k \in [50000]\}$. Similarly, we run the Polya-Gamma augmentation for 5000 draws.

These 36 datasets represent a full factorial experiment on three separate parameters. To assess how each algorithm performs as the number of neurons N increases, we consider 4 different values of N: $N = 25$, $N = 50$, $N = 100$, and $N = 150$. To assess how each algorithm performs as the number of time bins T

increases, we consider 3 different values of T: $T = 1000$, $T = 10000$, and $T = 50000$. Finally, we also want to assess how each algorithm performs under varying degrees of class imbalance. When generating synthetic spike trains according to the logistic regression model outlined in Section 5.2.2, a straightforward way to globally control the number spikes in all neurons to vary the intercept terms b. Across the trials, we consider 3 different values for the intercept terms b: $b = -3$ (for all N neurons), $b = -4$ (for all N neurons), and $b = -5$ (for all N neurons). These values correspond to roughly 4 percent, 2 percent, and 1 percent of time bins having spikes, respectively.

Recall from Section 5.2.2 that in addition to N, T, and b, generating a full set of synthetic spike trains requires specification of the number of basis functions $F \in \mathbb{N}$, the max time of coupling $t_{\max} \in \mathbb{N}$, the basis functions $g_1, \ldots, g_F : [t_{\max}] \to [0, 1]$ and the coupling coefficients $\beta \in \mathbb{R}^{F \times N \times N}$.

For all 36 synthetically generated datasets, we used the following structure. We used $F = 3$ basis functions with $t_{\max} = 30$. For the basis functions, we used raised cosine bumps (Pillow et al., 2008). The values in β were randomly generated to be 90 percent sparse, except the self-coupling filters, which were parametrized as dense. The nonzero entries in β were generated from zero mean Laplace distributions such that the entries corresponding to the first basis function had parameter 2, the second basis function had parameter 5, and the third basis function had parameter 10. To ensure stability of the process, the distribution of coefficients of the first and second basis functions for the self-coupling filters were modified slightly—instead generated using negative exponential distributions with means -2 and -1/5, respectively. This prevented any neuron from having its own positive feedback loop in which it was always spiking.

For each of the 36 datasets, we used both Polya-Gamma and zigzag algorithms to compute the posterior of the spike train coefficients that modeled the spiking of the first neuron. Running these algorithms required priors for β. For both algorithms, we defined the spike-and-slab prior π_0 such that the non-self coupling filter coefficients had 90 percent prior probability p_i of being 0, and the three self-coupling filter coefficients, as well as the intercept b had 1 percent prior probability p_i of being 0. The parameters α_i were chosen to match[¶] those used to generate the coefficient values (i.e. (2, 5, and 10) respectively for the different basis functions of non-self coupling filters, (1/2, 5, and 10) for the self-coupling filters). We assigned $\alpha_{NF+1} = 1/3$ for the intercept b.

Boxplots demonstrating the relative performance of the zigzag process to that of Polya-Gamma algorithm for each of the 36 datasets are shown in Figure 5.4, Figure 5.5, and Figure 5.6. In all three figures, there is a boxplot summarizing the results of running the two posterior inference algorithms for each dataset. These boxplots are constructed from $NF + 1$ values—one corresponding to each coefficient being inferred in the posteriors. Each of the values themselves represent the ratio of two quantities pertaining to the

[¶]Since the Polya-Gamma algorithm requires that Gaussian distributions be used instead of Laplace distributions as the slab components of the priors, we simply chose Gaussian distributions that matched the standard deviations of corresponding Laplace distributions when implementing the Polya-Gamma algorithm. Since these priors are quite similar, the downstream effects on the posterior were minimal

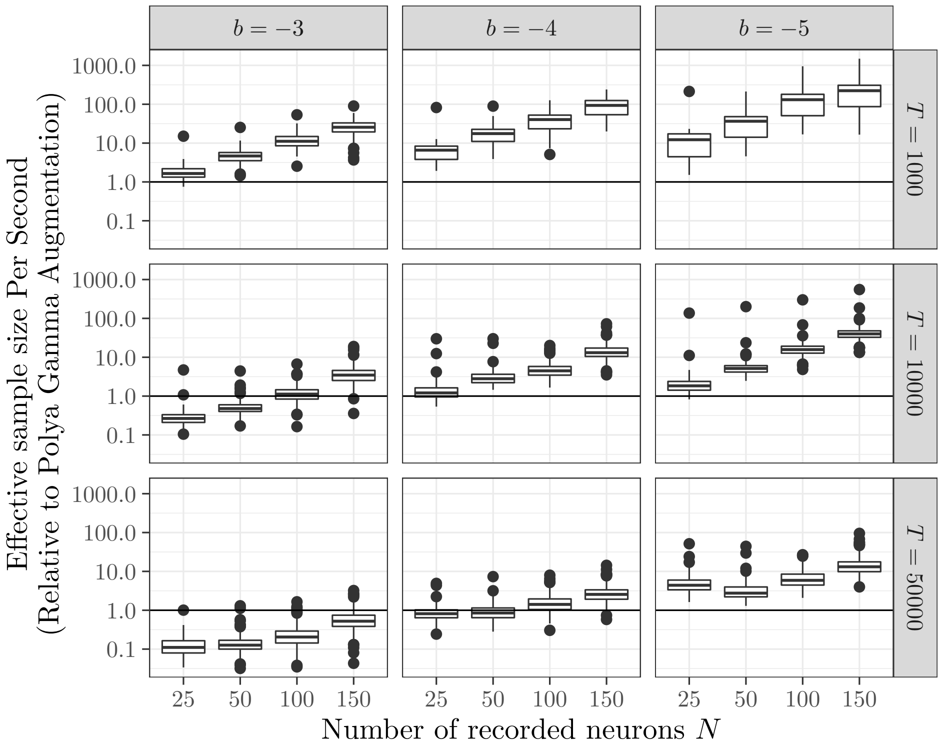


Figure 5.4: Boxplots depicting how the efficiency (in terms of effective sample size per second) of our zigzag process-based posterior inference algorithm compares to that of the Polya-Gamma augmentation algorithm of Linderman et al. (2016) for doing posterior inference on the regression coefficients as the number of neurons N increases. Each boxplot corresponds to a different synthetically generated dataset. The datasets differ in the numbers of neurons N considered, numbers of time bins T reported, and logistic regression intercept b of the model used to generated the synthetic data. Each boxplot is constructed using the ratio of effective sample sizes per second of both algorithms for inferring each coefficient in the regression. Note that the vertical axis is on a logarithmic scale, with the horizontal line at 1 showing the value at which the two algorithms are considered equally efficient.

same coefficient—specifically, the effective sample size per second of the zigzag algorithm for inferring the posterior of that coefficient, divided by the effective sample size per second of the Polya-Gamma algorithm for inferring the posterior of that coefficient. That is, a value of 5 on the vertical axis would indicate that zigzag is five times as efficient as Polya-Gamma for inferring that coefficient, whereas a value of 1/2 would indicate that it was only has as efficient. Therefore, the horizontal lines at 1 in each panel depict a value for which the two algorithms would be considered equally efficient.

Though Figures 5.4, 5.5, and 5.6 technically contain the same information, each of the three is arranged differently to emphasize a different relationship. Figure 5.4 is arranged to demonstrate that the zigzag process tends to outperform the Polya-Gamma augmentation to a higher and higher degree as the number of neurons being recorded grows, and this holds true across all numbers of time bins and all levels of category imbalance implied by the different values of b. Note that the lower lefthand panel shows that zigzag tends

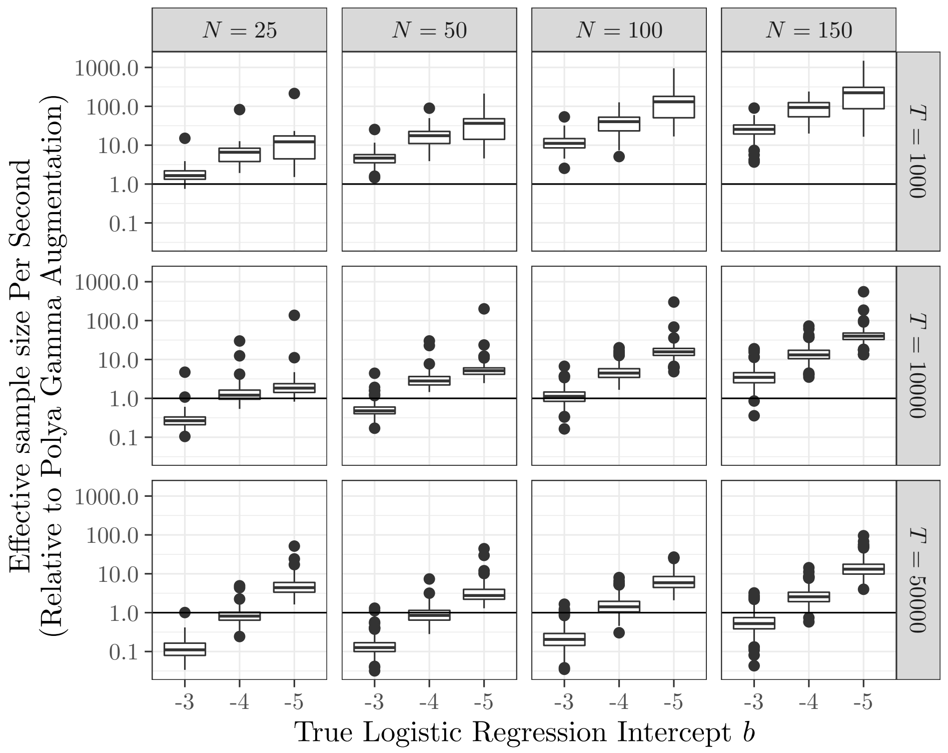


Figure 5.5: Boxplots depicting how the efficiency (in terms of effective sample size per second) of our zigzag process-based posterior inference algorithm compares to that of the Polya-Gamma augmentation algorithm of Linderman et al. (2016) for doing posterior inference on the regression coefficients as the value of the regression intercept b *decreases*. Each boxplot corresponds to a different synthetically generated dataset. The datasets differ in the numbers of neurons N considered, numbers of time bins T reported, and logistic regression intercept b of the model used to generated the synthetic data. Each boxplot is constructed using the ratio of effective sample sizes per second of both algorithms for inferring each coefficient in the regression. Note that the vertical axis is on a logarithmic scale, with the horizontal line at 1 showing the value at which the two algorithms are considered equally efficient.

to be at its relative worst for small values of N, large values of T, and lower levels of covariate imbalance. This is the behavior we expected to see based on our heuristic argument in Section 5.4.1.

Figure 5.5 demonstrates that zigzag process performs relatively better as the amount of class imbalance increases. The gains appear to be steepest when the number of time bins T is large. Once again, this supports the heuristic argument provided in Section 5.4.1.

Finally, Figure 5.6 is the more sobering of the three Figures. It demonstrates that as the number of time bins T increases, the zigzag process tends to perform relatively worse. This suggests that the zigzag process is not a panacea for all spike train logistic regression problems. In the more traditional large T small N settings, one is better off using Polya-Gamma unless the categorical imbalance is incredibly lopsided. Once again, this result is consistent with complexity arguments we provided in Section 5.4.1.

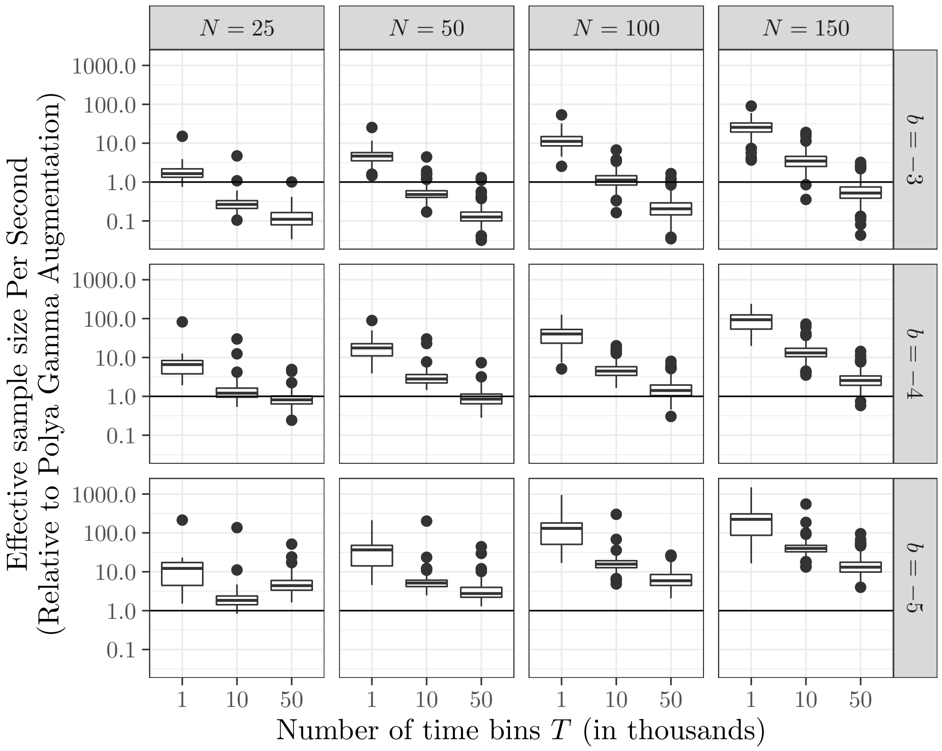


Figure 5.6: Boxplots depicting how the efficiency (in terms of effective sample size per second) of our zigzag process-based posterior inference algorithm compares to that of the Polya-Gamma augmentation algorithm of Linderman et al. (2016) for doing posterior inference on the regression coefficients as the number of observed time bins T increases. Each boxplot corresponds to a different synthetically generated dataset. The datasets differ in the numbers of neurons N considered, numbers of time bins T reported, and logistic regression intercept b of the model used to generated the synthetic data. Each boxplot is constructed using the ratio of effective sample sizes per second of both algorithms for inferring each coefficient in the regression. Note that the vertical axis is on a logarithmic scale, with the horizontal line at 1 showing the value at which the two algorithms are considered equally efficient.

Overall, the results of these synthetic experiments show that our algorithm is a promising new approach to fitting spike train logistic regressions to modern datasets. Even for a relatively modest-sized spike trains consisting of 150 neurons, one can expect our zigzag algorithm to be roughly 50 to 100 times faster than the start of the state of the Polya-Gamma algorithm. The trend as N increases in Figure 5.4, as well as our heuristic complexity arguments in Section 5.4.1 seem to indicate that the gains could be orders of magnitude higher for the massive thousands-of-neurons modern datasets.

In this section, our discussion of the scaling performance of our zigzag algorithm has concentrated on comparisons to the Polya-Gamma algorithm. However, applying the zigzag process to a neural dataset requires knowledge of the raw performance of the zigzag algorithm as well. For instance, how long does it take to fit a given dataset? As a reference, we have included Figure 5.7 and Figure 5.8 that report the raw computer runtime in seconds that it took for our zigzag implementation to generate the 50000 iterations used in each of our 36 synthetic datasets. These runtimes were obtained on the Bridges High Performance Computing System (Nystrom et al., 2015) at the Pittsburgh Supercomputing Center (the details of which are already discussed in Section 4.6.1). Each run was performed on a single core with no parallelization. Figures 5.7 and 5.8 contain the same information, but are arranged differently to illustrate the scaling behavior in terms of the number of neurons N and the number of time bins T, respectively. It is noteworthy that both increasing relationships shown support the heuristic scaling argument provided in Section 5.4.1. The longest runtime of 127315 seconds ($\approx$ 35 hours) corresponded to the case with the most neurons ($N = 150$), longest duration ($T = 50000$) and most spikes ($b = -3$). The shortest runtime of 19 seconds corresponds to the case with the fewest neurons ($N = 25$), shortest duration ($T = 1000$) and fewest spikes ($b = -5$).

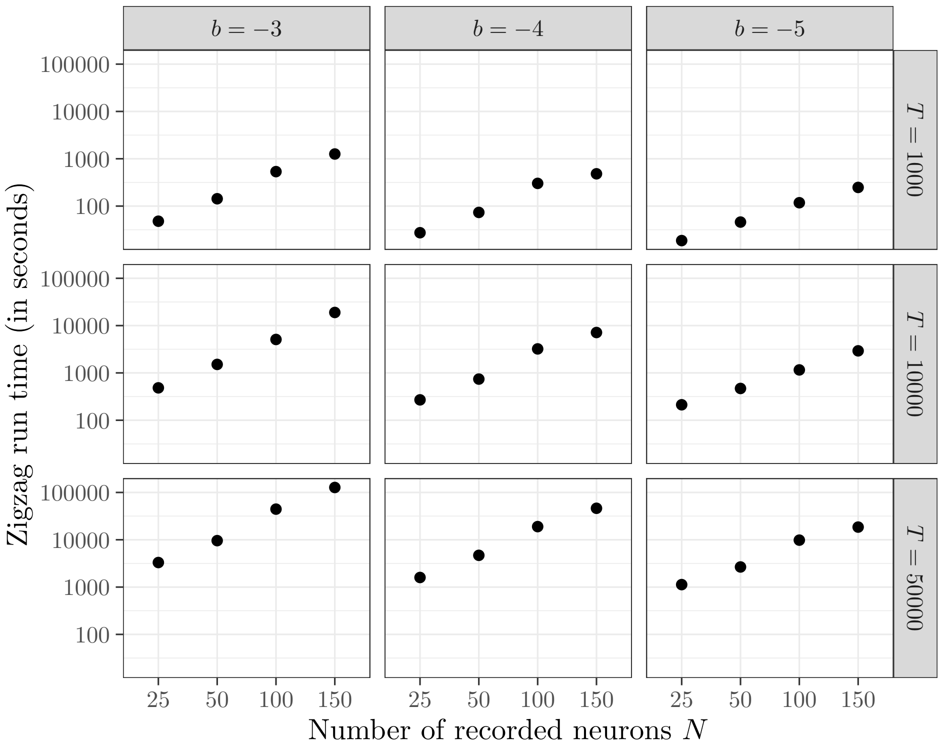


Figure 5.7: A companion plot for Figure 5.4 reporting the runtime (in seconds) it took to compute the zigzag trajectory exploring the posterior associated with each of the synthetic spike train. Note that for each posterior, the algorithm was run to generate a trajectory of length $\tau = 5000$ reported at regular intervals to obtain a chain of length 50000.

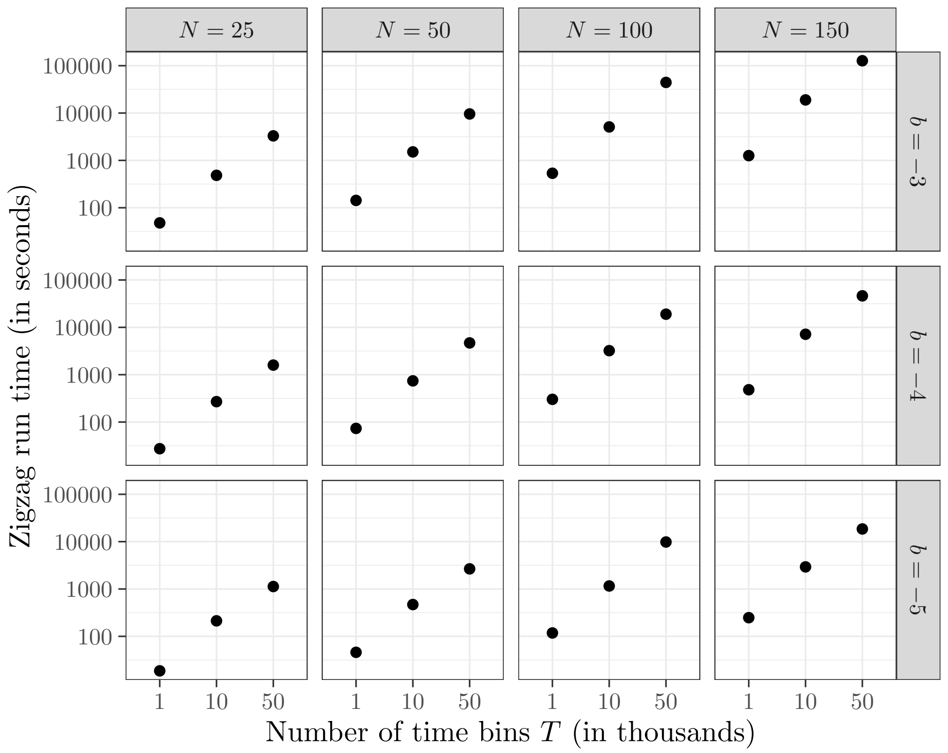


Figure 5.8: A companion plot for Figure 5.6 reporting the runtime (in seconds) it took to compute the zigzag trajectory exploring the posterior associated with each of the synthetic spike train. Note that for each posterior, the algorithm was run to generate a trajectory of length $\tau = 5000$ reported at regular intervals to obtain a chain of length 50000.

5.5 Conclusion and Future Work

Though the synthetic results in Section 5.4.2 are promising, we feel this is just the first step in investigating exactly how well our algorithm performs. The natural next step we intend to explore is using it to fit a model to real neural datasets. Doing so will require extending past the toy synthetic environment on which this chapter focused. For instance, we will need to augment the design matrix X to accommodate varying intercept terms and the presence of extrinsic covariates. Our algorithm should extend easily to this setting, barring very irregular structure in X that makes bounce time simulation costly.

Furthermore, one can easily imagine scenarios in which one would want to move past the unstructured spike-and-slab prior we have presented here. We intend to extend our algorithm to accommodate the latent variable structure in the sparsity, as proposed in Linderman and Adams (2014); Linderman et al. (2016). Incorporating this approach requires a method for updating the latent variables in the sampler. Since the latent variables may not be amenable to zigzag sampling directly, we intend to explore the zigzag Gibbs hybrid approach proposed in Sachs et al. (2020). This could facilitate the combining of the latent position model inference algorithm outlined in Chapter 4 with the work presented here. Another option would be a stochastic block model (Holland et al., 1983). We have also done some preliminary work using this approach to incorporate horseshoe (Carvalho et al., 2009) and horseshoe-like (Bhadra et al., 2017) priors within this Gibbs framework to provide different shrinkage options beyond spike-and-slab. These could also be combined with a latent variable structured sparsity as in Linderman et al. (2016).

In a different thread, one can easily imagine extending the approach present here to accommodate other link functions and spike distributions other than Bernoulli distribution with the logistic link. One caveat is that the gradient of the corresponding log likelihood should globally upper-bounded, otherwise our efficient thinning-based bounce simulation algorithms will not work. However, spike distributions such as the negative binomial with logistic link (Pillow and Scott, 2012), or the Poisson distribution with the softplus link (Zoltowski and Pillow, 2018) do meet this criteria.

Finally, we should note that the Polya-Gamma algorithm of (Linderman et al., 2016) may not be the only worthy competitor out there. It is worth exploring other competitor algorithms that can accommodate the discreteness of spike-and-slab, such as those involving reversible jump MCMC (Green, 1995). It may also be worth exploring how more generic algorithms such as random walk Metropolis or Hamiltonian Monte Carlo (Neal, 2011) would fair using our novel continuous and piece-wise differentiable parametrization of the spike-and-slab prior. One can also imagine extending this new parametrization of the spike-and-slab prior to other sparse modeling settings, such as sparse estimation of factor loadings in factor analysis (Carvalho et al., 2008), or non-negative matrix factorization with sparseness constraints (Hoyer, 2004). Finally, it is worth investigating if the approach we apply here for the zigzag also applies to other versions of piece-wise deterministic Markov processes such as the Bouncy Particle Sampler (Bouchard-Côté et al., 2018) or the Boomerang Sampler (Bierkens et al., 2020a).

Overall, we hope that through extending our work to apply to real data and real problems, as well as comparing to other existing algorithms, we can develop a technique that allows us to scale full Bayes inference up to massive modern spike train data.

Bibliography

Adair, T., Lemay, J., McDonald, A., Shaw, R., and Tewes, R. (2007). The Mount Bierstadt study: An experiment in unique damage formation in footwear. *Journal of Forensic Identification*, 57(2):199. 17

Adams, R. P., Murray, I., and MacKay, D. J. (2009). Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 9–16. ACM. 35

Aicher, C., Jacobs, A. Z., and Clauset, A. (2014). Learning latent block structure in weighted networks. *Journal of Complex Networks*, 3(2):221–248. 3

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of machine learning research*, 9(Sep):1981–2014. 3, 93

Aldous, D. J. (1981). Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598. 3

Aliverti, E. and Durante, D. (2019). Spatial modeling of brain connectivity data via latent distance models with nodes clustering. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 12(3):185–196. 99

Arias-Castro, E., Channarond, A., Pelletier, B., and Verzelen, N. (2018). On the estimation of latent distances using graph distances. *arXiv preprint arXiv:1804.10611*. 58

Athreya, A., Fishkind, D. E., Levin, K., Lyzinski, V., Park, Y., Qin, Y., Sussman, D. L., Tang, M., Vogelstein, J. T., and Priebe, C. E. (2017). Statistical inference on random dot product graphs: a survey. *arXiv preprint arXiv:1709.05454*. 3

Baddeley, A., Bárány, I., and Schneider, R. (2007). Spatial point processes and their applications. *Stochastic Geometry: Lectures given at the CIME Summer School held in Martina Franca, Italy, September 13–18, 2004*, pages 1–75. 4

Bales, B., Pourzanjani, A., Vehtari, A., and Petzold, L. (2019). Selecting the metric in hamiltonian monte carlo. *arXiv preprint arXiv:1905.11916*. 103

Barber, R. F., Drton, M., and Tan, K. M. (2016). Laplace approximation in high-dimensional bayesian regression. In *Statistical Analysis for High-Dimensional Data*, pages 15–36. Springer. 136

Bardenet, R., Doucet, A., and Holmes, C. (2017). On markov chain monte carlo methods for tall data. *The Journal of Machine Learning Research*, 18(1):1515–1557. 136

Beskos, A. and Stuart, A. (2009). Computational complexity of metropolis-hastings methods in high dimensions. In *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 61–71. Springer. 136

Betancourt, M. (2016). Identifying the optimal integration time in hamiltonian monte carlo. *arXiv preprint arXiv:1601.00225*. 95, 127

Betancourt, M. (2017). A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*. 95, 101, 103

Bhadra, A., Datta, J., Polson, N. G., and Willard, B. (2017). Horseshoe regularization for feature subset selection. *arXiv preprint arXiv:1702.07400*. 150, 166

Bhadra, A., Datta, J., Polson, N. G., Willard, B., et al. (2019). Lasso meets horseshoe: A survey. *Statistical Science*, 34(3):405–427. 150

Bierkens, J. and Duncan, A. (2017). Limit theorems for the zig-zag process. *Advances in Applied Probability*, 49(3):791–825. 142, 144

Bierkens, J., Fearnhead, P., Roberts, G., et al. (2019a). The zig-zag process and super-efficient sampling for bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320. 137, 142, 145, 152

Bierkens, J., Grazzi, S., Kamatani, K., and Roberts, G. (2020a). The boomerang sampler. *arXiv preprint arXiv:2006.13777*. 166

Bierkens, J., Grazzi, S., van der Meulen, F., and Schauer, M. (2020b). A piecewise deterministic monte carlo method for diffusion bridges. *arXiv preprint arXiv:2001.05889*. 142

Bierkens, J., Roberts, G. O., Zitt, P.-A., et al. (2019b). Ergodicity of the zigzag process. *The Annals of Applied Probability*, 29(4):2266–2301. 142, 144

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877. 136

Bloem-Reddy, B. and Cunningham, J. (2016). Slice sampling on hamiltonian trajectories. In *International Conference on Machine Learning*, pages 3050–3058. 104

Bodziak, W. J. (2017). *Forensic Footwear Evidence.* CRC Press. 9, 12

Bodziak, W. J., Hammer, L., Johnson, G. M., and Schenck, R. (2012). Determining the significance of outsole wear characteristics during the forensic examination of footwear impression evidence. *Journal of Forensic Identification*, 62(3):254–278. 12

Bollobás, B., Janson, S., and Riordan, O. (2007). The phase transition in inhomogeneous random graphs. *Random Structures and Algorithms*, 31:3–122. 48, 63, 64

Borg, I. and Groenen, P. J. (2005). *Modern Multidimensional Scaling: Theory and Applications.* Springer. 82

Borgs, C., Chayes, J., Cohn, H., and Zhao, Y. (2014). An l^p theory of sparse graph convergence i: Limits, sparse random graph models, and power law distributions. *Transactions of the American Mathematical Society.* 3, 48, 63, 126

Borgs, C., Chayes, J. T., Cohn, H., and Holden, N. (2016). Sparse exchangeable graphs and their limits via graphon processes. Electronic pre-print, arxiv:1601.07134. 48, 65

Bouchard-Côté, A., Vollmer, S. J., and Doucet, A. (2018). The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *Journal of the American Statistical Association*, 113(522):855–867. 136, 145, 166

Brémaud, P. and Massoulié, L. (1996). Stability of nonlinear hawkes processes. *The Annals of Probability*, pages 1563–1588. 6

Brillinger, D. R. (1992). Nerve cell spike train data analysis: a progression of technique. *Journal of the American Statistical Association*, 87(418):260–271. 136

Brillinger, D. R., Guttorp, P. M., Schoenberg, F. P., El-Shaarawi, A. H., and Piegorsch, W. W. (2002). Point processes, temporal. *Encyclopedia of environmetrics*, 3:1577–1581. 4

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov chain Monte Carlo.* CRC press. 25

Brown, E. N., Kass, R. E., and Mitra, P. P. (2004). Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature neuroscience*, 7(5):456–461. 136

Campbell, T. and Broderick, T. (2018). Bayesian coreset construction via greedy iterative geodesic ascent. *arXiv preprint arXiv:1802.01737.* 136

Caron, F. (2012). Bayesian nonparametric models for bipartite graphs. In *Advances in Neural Information Processing Systems*, pages 2051–2059. 64

Caron, F. and Fox, E. B. (2014). Sparse graphs using exchangeable random measures. arxiv:1401.1137. 3, 48, 63, 64

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1). 95, 103, 113, 115, 116

Carrington, P. J., Scott, J., and Wasserman, S. (2005). *Models and methods in social network analysis*, volume 28. Cambridge university press. 93

Carvalho, C. M., Chang, J., Lucas, J. E., Nevins, J. R., Wang, Q., and West, M. (2008). High-dimensional sparse factor modeling: applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456. 166

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. In *Artificial Intelligence and Statistics*, pages 73–80. 150, 166

Champod, C., Evett, I. W., and Jackson, G. (2004). Establishing the most appropriate databases for addressing source level propositions. *Science & Justice*, 44(3):153–164. 12

Chao, W.-L., Solomon, J., Michels, D., and Sha, F. (2015). Exponential integration for hamiltonian monte carlo. In *International Conference on Machine Learning*, pages 1142–1151. 102, 104

Chatterjee, S. (2015). Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43(1):177–214. 58

Chen, C., Rao, V., Buntine, W., and Teh, Y. W. (2013). Dependent normalized random measures. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 969–977, Atlanta, Georgia, USA. PMLR. 34

Chen, L., Vogelstein, J. T., Lyzinski, V., and Priebe, C. E. (2016). A joint graph inference case study: the c. elegans chemical and electrical connectomes. In *Worm*, volume 5, page e1142041. Taylor & Francis. 93

Chen, Y., Xin, Q., Ventura, V., and Kass, R. E. (2018). Stability of point process spiking neuron models. *Journal of computational neuroscience*, pages 1–14. 6

Chiu, G. S. and Westveld, A. H. (2011). A unifying approach for food webs, phylogeny, social networks, and statistics. *Proceedings of the National Academy of Sciences*, 108(38):15881–15886. 93

Choi, D. S. and Wolfe, P. J. (2011). Learnability of latent position network models. In *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, pages 521–524. IEEE. 57

Clauset, A., Moore, C., and Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98. 3, 93

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory.* John Wiley, New York, second edition. 78, 91

Cox, D. R. and Isham, V. (1980). *Point processes*, volume 12. CRC Press. 5

Crane, H. (2018). *Probabilistic foundations of statistical network analysis.* Chapman and Hall/CRC. 93

Crane, H. and Dempsey, W. (2016). A framework for statistical network modeling. Electronic pre-print, arxiv:1509.08185. 50

CSAFE (2019). *Longitudinal Shoe Outsole Impression Study.* Center for Statistics and Applications in Forensic Evidence. `http://forensicstats.org/shoeoutsoleimpressionstudy/`. 32

Dabbs, B., Adhikari, S., and Sweet, T. (2020). Conditionally independent dyads (cid) network models: a latent variable approach to statistical social network analysis. *Social Networks*, Revision Under Review. 93

Daley, D. J. and Vere-Jones, D. (2007). *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure.* Springer Science & Business Media. 4, 17

Damary, N. K., Mandel, M., Wiesner, S., Yekutieli, Y., Shor, Y., and Spiegelman, C. (2018). Dependence among randomly acquired characteristics on shoeprints and their features. *Forensic Science International*, 283:173–179. 16, 17

D'Amour, A. and Airoldi, E. (2016). Misspecification, sparsity, and superpopulation inference for sparse social networks. 50

Davenport, M. A., Plan, Y., Van Den Berg, E., and Wootters, M. (2014). 1-bit matrix completion. *Information and Inference: A Journal of the IMA*, 3(3):189–223. 49, 58, 77

de Valpine, P. (2018). How we make mcmc comparisons. `https://nature.berkeley.edu/~pdevalpine/MCMC_comparisons/nimble_MCMC_comparisons.html`. Accessed: 2020-04-24. 113

Devroye, L. (1986). Non-uniform random variate generation. 145

Diaconis, P. and Janson, S. (2008). Graph limits and exchangeable random graphs. *Rendiconti di Matematica e delle sue Applicazioni*, 28:33–61. 2, 47

Diaz, J., McDiarmid, C., and Mitsche, D. (2018). Learning random points from geometric graphs or orderings. *arXiv preprint arXiv:1809.09879*. 58

Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3. 127

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222. 95

Dunson, D. B. and Park, J.-H. (2008). Kernel stick-breaking processes. *Biometrika*, 95(2):307–323. 34

Edwards, H. T. and Gotsonis, C. (2009). Strengthening forensic science in the United States: A path forward. *Statement before the United States Senate Committee on the Judiciary*. 10

Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60. 93

Escobar, M. D. and West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588. 19

Evett, I., Lambert, J., and Buckleton, J. (1998). A Bayesian approach to interpreting footwear marks in forensic casework. *Science & Justice*, 38(4):241–247. 10, 11, 12, 33

Facey, O., Hannah, I., and Rosen, D. (1992). Shoe wear patterns and pressure distribution under feet and shoes, determined by image analysis. *Journal of the Forensic Science Society*, 32(1):15–25. 12

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *Ann. Statist.*, 1(2):209–230. 34

Fosdick, B. K., McCormick, T. H., Murphy, T. B., Ng, T. L. J., and Westling, T. (2018). Multiresolution network models. *Journal of Computational and Graphical Statistics*, (just-accepted):1–33. 93, 99, 127

Foti, N. and Williamson, S. (2012). Slice sampling normalized kernel-weighted completely random measure mixture models. In *Advances in Neural Information Processing Systems*, pages 2240–2248. 34

Foti, N. J. and Williamson, S. A. (2015). A survey of non-exchangeable priors for Bayesian nonparametric models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):359–371. 18, 35

Fruchtenicht, T., Herzig, W., and Blackledge, R. D. (2002). The discrimination of two-dimensional military boot impressions based on wear patterns. *Science & Justice*, 42(2):97–104. 12

Gamerman, D. and Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press. 93, 113

Gelfand, A. E., Kottas, A., and MacEachern, S. N. (2005). Bayesian nonparametric spatial modeling with Dirichlet process mixing. *Journal of the American Statistical Association*, 100(471):1021–1035. 34

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press. 93

Geyer, C. J. (1992). Practical markov chain monte carlo. *Statistical science*, pages 473–483. 136

Gibbs, A. L. and Su, F. E. (2002). On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435. 81

Gilbert, E. N. (1961). Random plane networks. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):533–543. 52

Girolami, M. and Calderhead, B. (2011). Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214. 95, 103

Goldenberg, A., Zheng, A. X., Fienberg, S. E., Airoldi, E. M., et al. (2010). A survey of statistical network models. *Foundations and Trends® in Machine Learning*, 2(2):129–233. 93

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732. 150, 166

Griffin, J., Leisen, F., et al. (2018). Modelling and computation using ncorm mixtures for density regression. *Bayesian Analysis*, 13(3):897–916. 19

Griffin, J. E. and Leisen, F. (2017). Compound random measures and their use in bayesian non-parametrics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(2):525–545. 11, 19, 20

Hahn, P. R., He, J., and Lopes, H. F. (2019). Efficient sampling for gaussian linear regression with arbitrary priors. *Journal of Computational and Graphical Statistics*, 28(1):142–154. 115, 136

Hamilton, J. and Press, P. U. (1994). *Time Series Analysis*. Princeton University Press. 141

Handcock, M. S., Raftery, A. E., and Tantrum, J. M. (2007). Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354. 51, 94, 99

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media. 148

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58:83–90. 4

Hecker, M., Lambeck, S., Toepfer, S., Van Someren, E., and Guthke, R. (2009). Gene regulatory network inference: data integration in dynamic models—a review. *Biosystems*, 96(1):86–103. 93

Herlau, T., Schmidt, M. N., and Mørup, M. (2016). Completely random measures for modelling block-structured sparse networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29 [NIPS 2016]*, pages 4260–4268. Curran Associates. 65

Herman, M. (2016). *Measurements and Scoring Procedures for Footwear Impression Comparisons.* National Institute for Standards in Technology. `https://www.nist.gov/sites/default/files/documents/2016/12/19/mherman.pdf`. 33

Hjort, N. L., Holmes, C., Müller, P., and Walker, S. G. (2010). *Bayesian Nonparametrics.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. 18

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30. 87

Hoff, P. D. (2005). Bilinear mixed-effects models for dyadic data. *Journal of the american Statistical association*, 100(469):286–295. 58

Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002). Latent space approaches to social network analysis. *J. Am. Stat. Association*, 97(460):1090–1098. 2, 47, 51, 59, 62, 94, 96, 97, 99

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623. 95, 103, 115, 127

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137. 2, 166

Horn, R. A. and Johnson, C. R. (1990). *Matrix analysis.* Cambridge university press. 83, 84, 87, 89

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469. 166

Huggins, J., Adams, R. P., and Broderick, T. (2017). Pass-glm: polynomial approximate sufficient statistics for scalable bayesian glm inference. In *Advances in Neural Information Processing Systems*, pages 3611–3621. 136

Huggins, J. H., Kasprzak, M., Campbell, T., and Broderick, T. (2019). Practical posterior error bounds from variational objectives. *arXiv preprint arXiv:1910.04102.* 137

Jewell, S., Spencer, N., and Bouchard-Côté, A. (2015). Atomic spatial processes. In *International Conference on Machine Learning*, pages 248–256. 34

Ji, P. and Jin, J. (2016). Coauthorship and citation networks for statisticians. *The Annals of Applied Statistics*, 10(4):1779–1812. 93

Johndrow, J. E., Orenstein, P., and Bhattacharya, A. (2017). Bayes shrinkage at gwas scale: Convergence and approximation theory of a scalable mcmc algorithm for the horseshoe prior. *arXiv preprint arXiv:1705.00841*. 150

Johndrow, J. E., Smith, A., Pillai, N., and Dunson, D. B. (2018). Mcmc for imbalanced categorical data. *Journal of the American Statistical Association*, pages 1–10. 159

Jordan, M. I. (2010). *Hierarchical models, nested models and completely random measures*, pages 207–218. New York: Springer. 18

Jun, J. J., Steinmetz, N. A., Siegle, J. H., Denman, D. J., Bauza, M., Barbarits, B., Lee, A. K., Anastassiou, C. A., Andrei, A., Aydın, Ç., et al. (2017). Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232. 136, 158

Kallenberg, O. (2002). *Foundations of Modern Probability.* Springer-Verlag, New York, second edition. 59

Kartun-Giles, A. P., Krioukov, D., Gleeson, J. P., Moreno, Y., and Bianconi, G. (2018). Sparse power-law network model for reliable statistical predictions based on sampled data. *Entropy*, 20(4):257. 50

Kass, R. E., Amari, S.-I., Arai, K., Brown, E. N., Diekman, C. O., Diesmann, M., Doiron, B., Eden, U. T., Fairhall, A. L., Fiddyment, G. M., et al. (2018). Computational neuroscience: Mathematical and statistical perspectives. *Annual review of statistics and its application*, 5:183–214. 136

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998). Markov chain monte carlo in practice: a roundtable discussion. *The American Statistician*, 52(2):93–100. 112, 136, 159

Kass, R. E., Kelly, R. C., and Loh, W.-L. (2011). Assessment of synchrony in multiple neural spike trains using loglinear point process models. *The annals of applied statistics*, 5(2B):1262. 6

Kass, R. E. and Ventura, V. (2001). A spike-train probability model. *Neural computation*, 13(8):1713–1720. 139

Kim, B., Lee, K. H., Xue, L., and Niu, X. (2018). A review of dynamic network models with latent variables. *Statistics surveys*, 12:105. 95, 122, 127

Kingman, J. (1967). Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78. 18

Kingman, J. (1992). *Poisson Processes*, volume 3. Clarendon Press. 142, 145, 152

Kingman, J. F. C. (1993). *Poisson Processes.* Oxford University Press, Oxford. 4, 52, 67, 75

Kong, B., Ramanan, D., and Fowlkes, C. (2017). Cross-domain forensic shoeprint matching. In *British Machine Vision Conference (BMVC)*. 9

Koskela, J. (2020). Zig-zag sampling for discrete structures and non-reversible phylogenetic mcmc. *arXiv preprint arXiv:2004.08807*. 142, 145, 152

Kottas, A. and Sansó, B. (2007). Bayesian mixture modeling for spatial Poisson process intensities, with applications to extreme value analysis. *Journal of Statistical Planning and Inference*, 137(10):3151–3163. 34

Krioukov, D. and Ostilli, M. (2013). Duality between equilibrium and growing networks. *Physical Review E*, 88(2):022808. 49, 52, 53

Krivitsky, P. N. and Handcock, M. S. (2008). Fitting position latent cluster models for social networks with latentnet. *Journal of Statistical Software*, 24. 99

Krivitsky, P. N., Handcock, M. S., Raftery, A. E., and Hoff, P. D. (2009). Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models. *Social networks*, 31(3):204–213. 99, 127

Laub, P. J., Taimre, T., and Pollett, P. K. (2015). Hawkes processes. *arXiv preprint arXiv:1507.02822*. 4, 6

Laurent, B. and Massart, P. (2000). Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338. 70

Ledoux, M. and Talagrand, M. (1991). *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag, Berlin. 71, 72

Leimkuhler, B. and Reich, S. (2004). *Simulating hamiltonian dynamics*, volume 14. Cambridge university press. 104

Lijoi, A., Nipoti, B., and Prünster, I. (2014). Bayesian inference with dependent normalized completely random measures. *Bernoulli*, 20(3):1260–1291. 34

Linderman, S. and Adams, R. (2014). Discovering latent network structure in point process data. In *International Conference on Machine Learning*, pages 1413–1421. 165

Linderman, S., Adams, R. P., and Pillow, J. W. (2016). Bayesian latent structure discovery from multi-neuron recordings. In *Advances in neural information processing systems*, pages 2002–2010. xix, 94, 100, 136, 137, 139, 141, 148, 149, 157, 158, 159, 161, 162, 163, 165, 166

Livingstone, S., Betancourt, M., Byrne, S., Girolami, M., et al. (2019). On the geometric ergodicity of hamiltonian monte carlo. *Bernoulli*, 25(4A):3109–3138. 112

Lloyd, J. R., Orbanz, P., Ghahramani, Z., and Roy, D. M. (2012). Random function priors for exchangeable arrays with applications to graphs and relational data. In Bartlett, P., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25 [NIPS 2012]*, pages 1007–1015. Curran Associates. 3

Ma, Z. and Ma, Z. (2017). Exploration of large networks via fast and universal latent space model fitting. *arXiv preprint arXiv:1705.02372*. 58

MacEachern, S. N. (2000). Dependent Dirichlet processes. *Unpublished manuscript, Department of Statistics, The Ohio State University*, pages 1–40. 34

Maclaurin, D. and Adams, R. P. (2015). Firefly Monte Carlo: Exact MCMC with subsets of data. In *International Joint Conference on Artificial Intelligence*. 95, 104, 107

Mangoubi, O. and Smith, A. (2017). Rapid mixing of hamiltonian monte carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*. 112

Mangoubi, O. and Smith, A. (2019). Mixing of hamiltonian monte carlo on strongly log-concave distributions 2: Numerical integrators. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 586–595. 112

Mannseth, J., Kleppe, T. S., and Skaug, H. J. (2016). On the application of higher order symplectic integrators in hamiltonian monte carlo. *arXiv preprint arXiv:1608.07048*. 104

McFarland, D. D. and Brown, D. J. (1973). Social distance as a metric: A systematic introduction to smallest space analysis. In Laumann, E. O., editor, *Bonds of Pluralism: The Form and Substance of Urban Social Networks*, pages 213–253. John Wiley, New York. 51

Meester, R. and Roy, R. (1996). *Continuum Percolation*. Cambridge University Press, Cambridge. 48, 52, 64

Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the american statistical association*, 83(404):1023–1032. 137, 148

Møller, J., Syversveen, A. R., and Waagepetersen, R. P. (1998). Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482. 35

Murray, I., Adams, R., and MacKay, D. (2010). Elliptical slice sampling. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 541–548. 26, 39, 115

Neal, R. M. (2003). Slice sampling. *Annals of statistics*, pages 705–741. 26, 38, 39

Neal, R. M. (2011). *MCMC using Hamiltonian dynamics*, chapter 5. 95, 101, 102, 136, 150, 166

Newman, M. E. (2002). Spread of epidemic disease on networks. *Physical review E*, 66(1):016128. 93

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45:167–256. 47

Newman, M. E. J. (2010). *Networks: An Introduction.* Oxford University Press, Oxford, England. 3, 48

Nishimura, A. and Suchard, M. A. (2018). Prior-preconditioned conjugate gradient for accelerated gibbs sampling in" large n & large p" sparse bayesian logistic regression models. *arXiv preprint arXiv:1810.12437*. 136

Nystrom, N. A., Levine, M. J., Roskies, R. Z., and Scott, J. R. (2015). Bridges: a uniquely flexible hpc resource for new communities and data analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, pages 1–8. 128, 164

Orbanz, P. and Roy, D. M. (2015). Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:437–461. 3, 48

Pakman, A. and Paninski, L. (2014). Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542. 104, 106

Palla, K., Caron, F., and Teh, Y. W. (2016). Bayesian nonparametrics for sparse dynamic networks. Electronic pre-print, arXiv:1607.01624. 65

Papaspiliopoulos, O., Roberts, G. O., and Sköld, M. (2007). A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73. 109

Park, M. and Pillow, J. W. (2011). Receptive field inference with localized priors. *PLoS Comput Biol*, 7(10):e1002219. 148

Park, T. and Casella, G. (2008). The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686. 148

PCAST (2016). *Report to the President, Forensic Science in Criminal Courts: Ensuring Scientific Validity of Feature-comparison Methods.* Executive Office of the President of the United States, President's Council of Advisors on Science and Technolgy. 11, 13, 17, 31

Penrose, M. (2003). *Random Geometric Graphs.* Oxford University Press, Oxford. 52

Penrose, M. D. (1991). On a continuum percolation model. *Advances in Applied Probability*, 23:536–556. 52

Petraco, N. D., Gambino, C., Kubic, T. A., Olivio, D., and Petraco, N. (2010). Statistical discrimination of footwear: a method for the comparison of accidentals on shoe outsoles inspired by facial recognition techniques. *Journal of Forensic Sciences*, 55(1):34–41. 12, 17

Piironen, J., Vehtari, A., et al. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, 11(2):5018–5051. 150

Pillow, J. W. and Scott, J. (2012). Fully bayesian inference for neural models with negative-binomial spiking. In *Advances in neural information processing systems*, pages 1898–1906. 158, 166

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E., and Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995. 139, 160

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). Coda: convergence diagnosis and output analysis for mcmc. *R news*, 6(1):7–11. 112

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349. 136, 158

Raftery, A. E., Niu, X., Hoff, P. D., and Yeung, K. Y. (2012). Fast inference for the latent space network model using a case-control approximate likelihood. *Journal of Computational and Graphical Statistics*, 21(4):901–919. 94, 99, 100

Ramirez, A. D. and Paninski, L. (2014). Fast inference in generalized linear models via expected log-likelihoods. *Journal of computational neuroscience*, 36(2):215–234. 136

Rasmussen, C. E. (2000). The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems*, pages 554–560. 19

Rastelli, R. (2018). The sparse latent position model for nonnegative weighted networks. *arXiv preprint arXiv:1808.09262*. 50

Rastelli, R., Friel, N., and Raftery, A. E. (2016). Properties of latent variable network models. *Network Science*, 4(4):407–432. 62, 95, 97, 127

Rastelli, R., Maire, F., and Friel, N. (2018). Computationally efficient inference for latent position network models. *arXiv preprint arXiv:1804.02274*. 94, 100

Ravikumar, P., Wainwright, M. J., Raskutti, G., Yu, B., et al. (2011). High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980. 89

Richetelli, N., Lee, M. C., Lasky, C. A., Gump, M. E., and Speir, J. A. (2017a). Classification of footwear outsole patterns using Fourier transform and local interest points. *Forensic Science International*, 275:102–109. 9

Richetelli, N., Nobel, M., Bodziak, W. J., and Speir, J. A. (2017b). Quantitative assessment of similarity between randomly acquired characteristics on high quality exemplars and crime scene impressions via analysis of feature size and shape. *Forensic Science International*, 270:211–222. 13

Rieke, F., Warland, D., Van Steveninck, R. D. R., Bialek, W. S., et al. (1999). *Spikes: exploring the neural code*, volume 7. MIT press Cambridge. 136

Ripley, B. D. (2009). *Stochastic simulation*, volume 316. John Wiley & Sons. 112, 159

Roberts, G., Rosenthal, J., et al. (1997). Geometric ergodicity and hybrid markov chains. *Electronic Communications in Probability*, 2:13–25. 112

Roberts, G. O., Rosenthal, J. S., et al. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367. 100

Rocha, I., Janssen, J., and Kalyaniwalla, N. (2017). Recovering the structure of random linear graphs. *arXiv preprint arXiv:1704.03189*. 58

Ročková, V. and George, E. I. (2018). The spike-and-slab lasso. *Journal of the American Statistical Association*, 113(521):431–444. 148

Rodriguez, A., Dunson, D. B., and Gelfand, A. E. (2008). The nested Dirichlet process. *Journal of the American Statistical Association*, 103(483):1131–1154. 34

Rosenthal, J. S. (2017). Simple confidence intervals for mcmc without clts. *Electron. J. Statist.*, 11(1):211–214. 112

Sachs, M., Sen, D., Lu, J., and Dunson, D. (2020). Posterior computation with the gibbs zig-zag sampler. *arXiv preprint arXiv:2004.04254*. 142, 165

Salter-Townshend, M. and McCormick, T. H. (2017). Latent space models for multiview network data. *The annals of applied statistics*, 11(3):1217. 100

Salter-Townshend, M. and Murphy, T. B. (2013). Variational bayesian inference for the latent position cluster model for network data. *Computational Statistics & Data Analysis*, 57(1):661–671. 94

Schweinberger, M., Krivitsky, P. N., and Butts, C. T. (2017). Foundations of finite-, super-, and infinite-population random graph inference. Electronic pre-print, arxiv:1707.04800. 50

Seginer, Y. (2000). The expected norm of random matrices. *Combinatorics, Probability and Computing*, 9:149–166. 80

Sen, D., Sachs, M., Lu, J., and Dunson, D. (2019). Efficient posterior sampling for high-dimensional imbalanced logistic regression. *arXiv preprint arXiv:1905.11232*. 136, 137, 142, 146, 147

Sewell, D. K. and Chen, Y. (2016). Latent space models for dynamic networks with weighted edges. *Social Networks*, 44:105–116. 3

Shahbaba, B., Lan, S., Johnson, W. O., and Neal, R. M. (2014). Split Hamiltonian Monte Carlo. *Statistics and Computing*, 24(3):339–349. 95, 104

Shalizi, C. R. and Asta, D. (2017). Consistency of maximum likelihood embedding for continuous latent-space network models. Submitted. 57, 58

Shalizi, C. R. and McFowland III, E. (2016). Estimating causal peer influence in homophilous social networks by inferring latent locations. *arXiv preprint arXiv:1607.06565*. 94, 98

Shalizi, C. R. and Rinaldo, A. (2013). Consistency under sampling of exponential random graph models. *Annals of Statistics*, 41:508–535. 50, 66

Sheets, H. D., Gross, S., Langenburg, G., Bush, P. J., and Bush, M. A. (2013). Shape measurement tools in footwear analysis: a statistical investigation of accidental characteristics over time. *Forensic Science International*, 232(1-3):84–91. 13

Shor, Y., Wiesner, S., Tsach, T., Gurel, R., and Yekutieli, Y. (2017). Inherent variation in multiple shoe-sole test impressions. *Forensic Science International*. 10, 13

Shortreed, S., Handcock, M. S., and Hoff, P. (2006). Positional estimation within a latent space model for networks. *Methodology*, 2(1):24–33. 106, 113

Skerrett, J., Neumann, C., and Mateos-Garcia, I. (2011). A Bayesian approach for interpreting shoemark evidence in forensic casework: Accounting for wear features. *Forensic Science International*, 210(1-3):26–30. 10, 12

Snijders, T. A. B. (2010). Conditional marginalization for exponential random graph models. *Journal of Mathematical Sociology*, 34:239–252. 50

Song, D., Wang, H., Tu, C. Y., Marmarelis, V. Z., Hampson, R. E., Deadwyler, S. A., and Berger, T. W. (2013). Identification of sparse neural functional connectivity using penalized likelihood estimation and basis functions. *Journal of computational neuroscience*, 35(3):335–357. 139

Sorokin, P. A. (1927). *Social Mobility.* Harper and Brothers, New York. 51

Speir, J. A., Richetelli, N., Fagert, M., Hite, M., and Bodziak, W. J. (2016). Quantifying randomly acquired characteristics on outsoles in terms of shape and position. *Forensic science international*, 266:399–411. 11, 17

Spencer, N. A. (2020). Latentpositionnetworks. `https://github.com/neilspencer/LatentPositionsMCMC`. 128

Spencer, N. A. and Shalizi, C. R. (2019). Projective, sparse, and learnable latent position network models. *arXiv preprint arXiv:1709.09702*. 97, 126, 127

Spillane, J. P. and Hopkins, M. (2013). Organizing for instruction in education systems and school organizations: How the subject matters. *Journal of Curriculum Studies*, 45(6):721–747. 120

Spillane, J. P., Hopkins, M., and Sweet, T. M. (2018). School district educational infrastructure and change at scale: Teacher peer interactions and their beliefs about mathematics instruction. *American educational research journal*, 55(3):532–571. 97, 120

Srihari, S. N. and Su, C. (2011). Generative models and probability evaluation for forensic evidence. In *Pattern Recognition, Machine Intelligence and Biometrics*, pages 533–559. Springer. 12

Srihari, S. N. and Tang, Y. (2014). Computational methods for the analysis of footwear impression evidence. In *Computational Intelligence in Digital Forensics: Forensic Investigation and Applications*, pages 333–383. Springer. 9

Stone, R. S. (2006). Footwear examinations: mathematical probabilities of theoretical individual characteristics. *Journal of Forensic Identification*, 56(4):577. 17, 27, 30

Sussman, D. L., Tang, M., and Priebe, C. E. (2014). Consistent latent position estimation and vertex classification for random dot product graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:48–57. 58, 86

Sweet, T. and Adhikari, S. (2020). A latent space network model for social influence. *Psychometrika*, pages 1–24. 94, 97, 99

Sweet, T. M., Thomas, A. C., and Junker, B. W. (2013). Hierarchical network models for education research: Hierarchical latent space models. *Journal of Educational and Behavioral Statistics*, 38(3):295–318. 94

Taddy, M. A. (2010). Autoregressive mixture models for dynamic spatial Poisson processes: Application to tracking intensity of violent crime. *Journal of the American Statistical Association*, 105(492):1403–1417. 34

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2005). Sharing clusters among related groups: Hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems*, pages 1385–1392. 34

Thompson, W. C. and Newman, E. J. (2015). Lay understanding of forensic statistics: Evaluation of random match probabilities, likelihood ratios, and verbal equivalents. *Law and human behavior*, 39(4):332. 10

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288. 148

Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728. 112

Todeschini, A., Miscouridou, X., and Caron, F. (2016). Exchangeable random measures for sparse and modular graphs with overlapping communities. Electronic pre-print, arXiv:1602.02114. 65

Tokdar, S. T. and Kass, R. E. (2010). Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60. 25

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., et al. (2014). Xsede: Accelerating scientific discovery computing in science & engineering, 16 (5): 62–74, sep 2014. *URL https://doi. org/10.1109/mcse.* 128

Trippe, B., Huggins, J., Agrawal, R., and Broderick, T. (2019). LR-GLM: High-dimensional Bayesian inference using low-rank data approximations. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6315–6324, Long Beach, California, USA. PMLR. 136

Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., and Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of neurophysiology*, 93(2):1074–1089. 6, 136, 137, 139, 140

Tsybakov, A. B. (2008). *Introduction to Nonparametric Estimation.* Springer Verlag, New York. Translated by Vladimir Zaiats. 90

Turnbull, K. (2020). *Advancements in latent space network modelling.* PhD thesis, Lancaster University. 127

Veitch, V. and Roy, D. M. (2015). The class of random graphs arising from exchangeable random measures. Electronic pre-print, arXiv:1512.03099. 48, 63, 65

Vinci, G., Ventura, V., Smith, M. A., and Kass, R. E. (2018). Adjusted regularization in latent graphical models: Application to multiple-neuron spike count data. *The annals of applied statistics*, 12(2):1068. 136, 141, 148

Wasserman, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications.* Cambridge University Press, Cambridge, England. 51

Wilson, H. D. (2012). Comparison of the individual characteristics in the outsoles of thirty-nine pairs of Adidas supernova classic shoes. *Journal of Forensic Identification*, 62(3):194. 17

Wu, A., Koyejo, O., and Pillow, J. (2019). Dependent relevance determination for smooth and structured sparse regression. *Journal of Machine Learning Research*, 20(89):1–43. 136, 141, 148

Wyatt, J. M., Duncan, K., and Trimpe, M. A. (2005). Aging of shoes and its effect on shoeprint impressions. *Journal of Forensic Identification*, 55(2):181. 13

Xie, F. and Levinson, D. (2009). Modeling the growth of transportation networks: A comprehensive review. *Networks and Spatial Economics*, 9(3):291–307. 93

Xu, J. (2017). Rates of convergence of spectral methods for graphon estimation. *arXiv preprint arXiv:1709.03183*. 58

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018). Yes, but did it work?: Evaluating variational inference. *arXiv preprint arXiv:1802.02538*. 137

Yekutieli, Y., Shor, Y., Wiesner, S., and Tsach, T. (2012). Expert assisting computerized system for evaluating the degree of certainty in 2d shoeprints. Technical report, Technical report, Technical Report, TP-3211, National Institute of Justice. 10, 11, 14, 17, 27, 30

Yu, Y., Wang, T., and Samworth, R. J. (2014). A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323. 49, 74

Zoltowski, D. and Pillow, J. W. (2018). Scaling the poisson glm to massive neural datasets through polynomial approximations. In *Advances in Neural Information Processing Systems*, pages 3517–3527. 136, 148, 166