March 2020

Probabilistic Inference with Generating Functions for Population Dynamics of Unmarked Individuals

Kevin Winner

PROBABILISTIC INFERENCE WITH GENERATING FUNCTIONS FOR POPULATION DYNAMICS OF UNMARKED INDIVIDUALS

A Dissertation Presented

by

KEVIN WINNER

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2020

College of Information and Computer Sciences

PROBABILISTIC INFERENCE WITH GENERATING FUNCTIONS FOR POPULATION DYNAMICS OF UNMARKED INDIVIDUALS

A Dissertation Presented

by

KEVIN WINNER

Approved as to style and content by:

Daniel Sheldon, Chair

Justin Domke, Member

Benjamin Marlin, Member

Christopher Sutherland, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

It is common wisdom that grad school (and PhDs in particular) is generally a grueling, emotionally challenging, and *difficult* time in the life of an academic. Of course, the (averaged, biased) perspective of those who've made it out the other end seems to me to be that in spite of all the challenging aspects of the process it is ultimately a worthwhile and intellectually rewarding endeavor. By comparison, the past few years I've spent at UMass have been not only incredibly rewarding and edifying but also an overwhelmingly positive period of my life. There are so many people here in Amherst and elsewhere whose emotional and academic support has made it possible for me to not only complete a piece of work of which I am immensely proud but to honestly enjoy the process of doing it, and I'd like to use this space to briefly thank just a few of them.

Firstly, I am infinitely grateful to my advisor Dan Sheldon, who introduced me to a new domain, taught me an immeasurable amount about research, machine learning, statistics, and writing and who was unreasonably supportive throughout the entire process to boot. I consider myself extremely lucky to have stumbled into his group years ago as I could not possibly have wished for a better advisor. Thank you for everything Dan.

I'd also like to thank the wonderful members of my thesis committee: Justin Domke, Ben Marlin, and Chris Sutherland, to whom I (and this research) owe a great deal. Thank you all for being a part of the process with me and for helping to shape this project and this document. Also at UMass I'd like to thank the administrative team, particularly Leeanne Leclerc, Michele Roberts, and Eileen Hamel who gifted me with a remarkable amount of patience. From outside UMass, I'd like to thank three

other mentors and advisors: Justin Calabrese from SCBI, who inspired me to pursue my current line of research, Marie desJardins, who taught me so much about research and teaching during my time at UMBC, and Don Miner, who taught my undergrad AI elective at UMBC and then got me started in research in the first place.

Besides the research opportunity, one of the key factors that originally drew me to UMass was the deeply connected community within the department. This community taught me so much about how to survive and thrive in grad school and in the academic universe beyond grad school. Within that community, a few folks deserve special mention: Garrett Bernstein, Amanda Gentzel, and Kaleigh Clary, who joined me for an uncountable number of adventures and board game nights; the complete crew of housemates at 608, particularly Joie Wu, Keen Sung, Larkin Flodin, Li Yang Ku, Myungha Jang, and Tony Ohmann; Kyle Hollins Wray for running several years of tabletop adventures and Dirk Ruiken, John Vilk, Matteo Brucato, Samer Nashed, Zack Serritella, and others for playing through them with me; and Justin Svegliato, Katie Keith, Lucas Chaufournier, Su Lin Blodgett, and Tiffany Liu for helping me try and maintain the positive community at UMass for future generations.

Finally, beyond academia, I'd like to thank my mom, Leslie Vansant, and my sister, Arden Winner, for a lifetime of continuous encouragement and support through every single stage of my education. And last but not least, my wonderful partner Erin McVicar, who stood by me and encouraged me through the big, scary steps of defending, graduating, and moving to a new city and whose only compensation was getting a dog. I love you Erin, and thank you.

ABSTRACT

PROBABILISTIC INFERENCE WITH GENERATING FUNCTIONS FOR POPULATION DYNAMICS OF UNMARKED INDIVIDUALS

FEBRUARY 2020

KEVIN WINNER

B.Sc., UNIVERSITY OF MARYLAND, BALTIMORE COUNTY

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Daniel Sheldon

Modeling the interactions of different population dynamics (e.g. reproduction, migration) within a population is a challenging problem that underlies numerous ecological research questions. Powerful, interpretable models for population dynamics are key to developing intervention tactics, allocating limited conservation resources, and predicting the impact of uncertain environmental forces on a population. Fortunately, probabilistic graphical models provide a robust mechanistic framework for these kinds of problems. However, in the relatively common case where individuals in the population are *unmarked* (i.e. indistinguishable from one another), models of the population dynamics naturally contain a deceptively challenging statistical feature: **discrete latent variables with unbounded/countably infinite support**. Unfortunately, existing inference algorithms for discrete distributions are applicable only for finite distributions and while approximate inference algorithms exist for countably infinite

discrete distributions, they are generally unreliable and inefficient. In this work, we develop the *first known* general-purpose polynomial-time exact inference algorithms for this class of models using a novel representation based on probability generating functions. These methods are flexibe, easy to use, and *significantly* faster than existing approximate solutions. We also introduce a novel approximation scheme based on this technique that allows it to gracefully scale to populations well beyond the computational limits of any previously known exact or approximate general-purpose inference algorithm for population dynamics. Finally, we conduct an ecological case study on historical data demonstrating the downstream impact of these advances to a large scale population monitoring setting.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Count distributions (discrete distributions with infinite support, commonly $\mathbb{N}^0$, the set of nonnegative integers) such as the Poisson distribution are widely used to describe real world phenomena in ecology (Zonneveld, 1991; Royle, 2004; Dail & Madsen, 2011), epidemiology (Farrington et al., 2003; Panaretos, 2007; Kvitkovicova & Panaretos, 2011), queueing theory (Eick et al., 1993; Blanghaps et al., 2013) and numerous other fields in the natural and social sciences. Despite their prevalence, count distributions pose a significant challenge for probabilistic inference when used in latent variable models.

To understand why, consider the simple model shown in Figure 1.1. This model is known in the ecology literature as the N-mixture model (Royle, 2004) and is widely used in the study of population dynamics (Kéry et al., 2005; Mazerolle et al., 2007; Wenger & Freeman, 2008). The N-mixture model nicely showcases the subtle difficulty of performing inference with count-valued latent variables. In the original version of the N-mixture model, n is assumed to have a Poisson prior with rate λ and each observed variable y_k is assumed to be iid with conditional distribution $y_k \sim \text{Binomial}(n, \rho)$.

Given this model and some values for $\mathbf{y} = (y_1, \ldots, y_K)$, a researcher may be interested in fitting the values of λ and/or ρ to their data. To do so, they write the equation for the posterior distribution $p(n, \mathbf{y})$, construct the likelihood function by marginalizing out the latent variable n: $p(\mathbf{y}) = \sum_{n \in \mathbb{N}^0} p(n, \mathbf{y})$, and finally optimize the likelihood (e.g., numerically). Mathematically, this process seems trivial, but

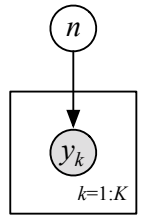

Figure 1.1. The N-mixture model of Royle (2004). In this model, K repeated independent surveys are conducted of a study population. In each survey k, a count-valued record y_k is sampled iid from the true latent abundance n. Traditionally $y_k \sim \text{Binomial}(n, \rho)$ and $n \sim \text{Poisson}(\lambda)$, though other authors have extended the model beyond this case.

unfortuantely for count distributions the infinite sum introduced by marginalization cannot be computed by direct summation. Recently, Haines (2016) presented an efficient closed form formulation of the likelihood function for the original formulation of the N-mixture presented here, but in the general case where a closed form does not exist, another strategy is needed.

In practice, researchers who wish to study models with latent count distributions have adopted one of the following broad classes of approaches:

1. Ignore the latent process and perform regression on the observed counts directly (Link & Sauer, 1994; Ralph et al., 1995; Link & Sauer, 1997; Schmidt & Pellet, 2009),

2. Truncate the support of the latent count distributions to a finite range (Royle, 2004; Dail & Madsen, 2011; Fiske & Chandler, 2011; Zipkin et al., 2014; Dennis et al., 2015; Hostetler & Chandler, 2015),

3. Employ a sampling strategy such as MCMC (Kéry et al., 2009; Zipkin et al., 2014; Hostetler & Chandler, 2015; Winner et al., 2015).

In the work that follows, we will focus on Options 2 and 3 above. Option 3 is discussed in Chapter 7, but Option 2 is the method implemented in `unmarked` (Fiske

& Chandler, 2011) (the R package used by ecologists for fitting these models to data) and comprises the baseline in the majority of our experiments, so we will devote some attention to it here.

In the truncated approach, the support of the count-valued latent variables (which is originally $\mathbb{N}^0$ by definition) is restricted to $[0, N_{\max}]$ where $N_{\max}$ is a new hyperparameter we call the "truncation parameter". Once the support has been truncated to a finite discrete range, traditional message passing algorithms for discrete distributions (Pearl, 1986; Lauritzen & Spiegelhalter, 1988; Jensen et al., 1990; Shenoy & Shafer, 1990) may then be applied.

In general, the runtime of message passing algorithms in the truncated approach scales directly with the chosen value of $N_{\max}$ and so much attention has been given to strategies for setting $N_{\max}$ (Dail & Madsen, 2011; Fiske & Chandler, 2011; Dennis et al., 2015). Setting $N_{\max}$ too low may have an unintended impact on inference and particularly on learning (Couturier et al., 2013; Dennis et al., 2015) while setting $N_{\max}$ too high is slow and discourages exhaustive model selection. In our experiments, however, even a conservative value for $N_{\max}$ is still prone to lead to bias in the case of parameter estimation when compared with an exact inference method.

In this work, we develop a new approach to inference for models with latent count variables. The key insight is to formulate a new representation for infinite factors based on probability generating functions (PGFs). In the chapters that follow, we will develop this representation as well as a family of novel exact and approximate inference algorithms that use this PGF representation. Collectively, we refer to this approach as "PGF inference".

PGF inference is a very flexible approach and our exact inference algorithms are the first known exact inference algorithms for many dynamics models including extensions of the N-mixture models and so-called "Dail-Madsen" models (Dail & Madsen, 2011; Hostetler & Chandler, 2015) (with notable exception to those specific models

addressed by Dennis et al. (2015) and Haines (2016)). We propose a novel family of dynamics models we call "latent count models" (LCMs) that generalizes N-mixture models and Dail-Madsen models and demonstrate how to apply PGF inference to this family. Finally, we show empirically and theoretically that PGF inference is significantly faster than existing general-purpose inference solutions.

1.1 Related Applications

In this work, we adopt the language of previous population monitoring research in the ecological community and use models of population dynamics from that literature to demonstrate our techniques. However, models with count-valued variables arise in a number of other domains and the PGF inference techniques we develop for the ecological community may also be applicable to many of these domains.

Fully observed time-series of counts have been extensively studied as "integer-valued autoregressive" (INAR) models (Al-Osh & Alzaid, 1987; McKenzie, 2003) with applications in many fields, including sociology, economics, medicine, and ecology. Similar models have arisen in queuing theory. The most closely related model is known as the $M_t/G/\infty$ queue (Eick et al., 1993). Notably, the process is typically assumed to be fully observed in these domains and as a result the types of questions and inference challenges are very different from our domain where the count-valued variables are latent.

Recently, the epidemiology community has adopted the models from (Royle, 2004) and (Dail & Madsen, 2011) directly to model the spread of diseases in unmarked populations (Brintz et al., 2018). In these settings, unmarked models may reflect disease spread through poorly monitored populations, diseases with latent infections, or even data that has been intentionally aggregated due to privacy concerns. This is very exciting, as there seems to exist a direct analogue between the domains, but as of yet this link seems relatively unexplored.

1.2 Contributions

To our knowledge, our PGF inference techniques represent the first known broadly-applicable, polynomial-time exact inference technology for a wide class of models with latent count distributions including, but not necessarily limited to, LCMs. Our exact inference algorithms scale to larger populations than existing approximate methods. The approximate PGF inference technique we develop in Chapter 6 is the first known inference algorithm for LCMs that does not scale with population size, meaning robust models of population dynamics can now be fit to data from massive geographic and temporal scales. Chapter 5 presents a novel generalization of forward-mode automatic differentiation (Griewank & Walther, 2008) for evaluating high-order nested derivatives more efficiently than existing naive approaches. Below we enumerate the major contributions presented in this thesis:

1. Formulation of Latent Count Models (Chapters 4 and 5)
    (a) Unification of existing open population dynamics models (Chapter 7)
2. PGF inference
    (a) PGF representation for count distributions (Chapter 4)
    (b) Efficient exact PGF inference algorithms (Chapters 4 and 5)
    (c) Moment matching PGF approximation scheme (Chapter 6)
    (d) Scalable approximate PGF inference algorithm (Chapter 6)
3. Generalized forward-mode automatic differentiation (Chapter 5)

1.3 Thesis organization

The rest of this document is organized as follows. Chapter 2 reviews the relevant background material for statistical inference and learning in graphical models. The hidden Markov model (HMM), the foundational model we use throughout the rest of

the document, is reviewed in Section 2.1 and then in Chapter 3 we review domain-specific extensions of the HMM in the population dynamics literature. Chapter 3 concludes with a novel model for population dynamics that will be used throughout the later chapters.

In Chapter 4, we introduce the core PGF representation that underpins all of PGF inference and is used throughout the rest of the thesis. Sections 2.3.1, 4.2, and 4.2.1 serve as a useful background on PGFs and collect many useful operations for manipulating PGFs. In Section 4.2.2 we demonstrate our first PGF inference algorithm: PGF-FORWARD, an analogue to the Forward algorithm for Hidden Markov Models (Rabiner, 1989; Ghahramani, 2001) using PGFs. In Section 4.2.2 we demonstrate how to implement PGF-FORWARD for a special case of the LCM. While this is the fastest of the exact inference algorithms we present, it does not generalize well to other models.

In Chapter 5, we extend PGF inference in a black-box fashion using an extension of standard forward-mode automatic differentiation (Griewank & Walther, 2008) (autodiff) to a more complete subset of the LCM family. The key insight in this chapter is that the majority of inference tasks in PGF inference don't require a complete representation of the PGFs of intermediate factors, rather we generally only need the value of the PGF (and some of its high-order derivatives) at a single point. This is very similar to what is done by forward-mode autodiff, but requires a significant generalization to handle functions with nested derivatives, which we present in Section 5.2.2. In Section 5.2.2.3, we introduce GDUAL-FORWARD, an implementation of PGF-FORWARD that uses our generalized forward-mode autodiff. Like our symbolic algorithm from Chapter 4, GDUAL-FORWARD is not an approximation, yet still scales better than the baseline approximate inference technique.

Our exact inference algorithms from Chapters 4 and 5, along with the approximate baseline TRUNC and sampling-based algorithms have the undesirable property that

the computation time scales with two quantities: the total number of observation events, and the magnitude of the observed counts. In Chapter 6, we introduce an approximate PGF inference algorithm APGF-FORWARD-S based on assumed density filtering (Minka, 2001) (ADF) that scales with the number of observation events only. In Section 6.2.1 we derive a moment-matching scheme called APGF for approximating a distribution represented with a PGF. In Sections 6.2.2 and 6.2.3 we show how to modify PGF-FORWARD to perform efficient approximate PGF inference using APGF in two approximate inference algorithms APGF-FORWARD and APGF-FORWARD-S.

Finally, in Chapter 7, we present an application of our approximate inference algorithm from Chapter 6 to a large-scale population study using historical data from the North American Breeding Bird Survey (Pardieck et al., 2019). This chapter also summarizes the incorporation of our PGF inference techniques into `unmarked` (Fiske & Chandler, 2011), the most widely used package in the ecological community for fitting LCMs to count data.

1.4 Publications

Chapters 4 and 5 of this thesis present material from the following two publications:

Ch 4: Winner, K. and Sheldon, D. Probabilistic Inference with Generating Functions for Poisson Latent Variable Models. In *Advances in Neural Information Processing Systems 29*, 2016.

Ch 5: Winner, K., Sujono, D., and Sheldon, D. Exact inference for integer latent-variable models. In *International Conference on Machine Learning (ICML)*, pp. 3761–3770, 2017.

Chapters 6 and 7 present unpublished work.

The following two publications are related but not featured here:

- Winner, K., Bernstein, G., and Sheldon, D. Inference in a Partially Observed Queueing Model with Applications in Ecology. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 2512–2520, 2015.

- Sheldon, D., Winner, K., and Sujono, D. Learning in integer latent variable models with nested automatic differentiation. In *Proceedings of the 35th International Conference on Machine Learning, (ICML)*, pp. 4622–4630, 2018.

Finally, the following unrelated publications feature research conducted during the course of my degree:

- Winner, K., Noonan, M. J., Fleming, C. H., Olson, K., Mueller, T., Sheldon, D., and Calabrese, J. M. Statistical inference for home range overlap. *Methods in Ecology and Evolution*, 2018.

- Lin, T.-Y., Winner, K., Bernstein, G., Mittal, A., Dokter, A. M., Horton, K. G., Nilsson, C., Van Doren, B. M., Farnsworth, A., La Sorte, F. A., Maji, S., and Sheldon, D. MistNet: Measuring historical bird migration in the US using archived weather radar data and convolutional neural networks. *Methods in Ecology and Evolution*, 2019.

CHAPTER 2

BACKGROUND

In this chapter we will review some foundational material for the techniques presented in the remainder of the thesis. We assume a basic familiarity with the fundamental concepts of graphical models and statistical inference and develop here the techniques and principles that form the baseline for our work. Chapter 3 has additional background on the models we build on and additional domain-specific related work. Where not otherwise specified, the primary reference for this review was Murphy (2012).

2.1 Graphical Models and Hidden Markov Models

The hidden Markov model (HMM) is a classic graphical model framework for sequence data and particularly for time-series data. In general, a graphical model is a concise way of specifying the dependencies between all the variables in our system. They come in two types: directed and undirected. In the work presented throughout this document, we focus exclusively on directed graphical models. An example of a directed graphical model (indeed an HMM) is shown in Figure 2.1. The HMM

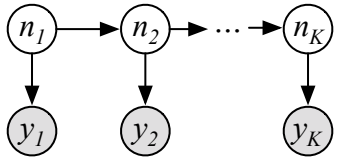

Figure 2.1. The hidden Markov model. The latent variables n_i form a Markov chain and the observed variables y_i depend only on a single corresponding latent variable.

is divided into a set of observed variables, denoted graphically with shaded nodes (the y_k variables in Figure 2.1) and a set of latent variables, denoted graphically with unshaded nodes (the n_k variables in Figure 2.1). Note that we use the $\mathbf{y}_{1:K}$ subscript notation to describe the tuple of $\{y_1, y_2, \ldots, y_K\}$ and so on. Also note that traditionally in an HMM x is used to denote the latent variables (instead of n), however in our applications the latent variables will always be count-valued (discrete with countably infinite support) and so we choose to use n in our presentation of HMMs to be consistent with later material.

In a directed graphical model like an HMM, the directed edges between nodes in the graph denote conditional dependencies between variables and imply a concrete method for factoring conditional distributions. More precisely, the graphical model in Figure 2.1 specifies for instance that the distribution of y_1 depends only on n_1, which is denoted by an edge between the two nodes.

In an HMM, the latent variables $\mathbf{n}_{1:K}$ form a "Markov chain": i.e. the distribution of one latent variable n_k depends only on the distribution of the preceding latent variable n_{k-1}. This property is known more generally as the "Markov property" and is key to performing efficient inference with HMMs.

The structure of the HMM graphical model allows us to factor the joint distribution of the latent variables as follows:

$$\begin{aligned} p(\mathbf{n}_{1:K}) &= p(n_1, n_2, \ldots, n_K) = p(n_1)p(n_2|n_1)\ldots p(n_K|n_{K-1}), \\ &= p(n_1)\prod_{k=2}^{K} p(n_k|n_{k-1}). \end{aligned} \tag{2.1}$$

Furthermore, the traditional HMM assumes that each observed variable depends only on a single corresponding latent variable, i.e. $p(y_k|\mathbf{n}_{1:K}) = p(y_k|n_k)$. As a result, the joint distribution of all the variables factors simply as follows:

$$
\begin{aligned}
p(\mathbf{n}_{1:K}, \mathbf{y}_{1:K}) &= p(\mathbf{n}_{1:K}) p(\mathbf{y}_{1:K} | \mathbf{n}_{1:K}), \\
&= p(n_1) \prod_{k=2}^{K} p(n_k | n_{k-1}) \prod_{k=1}^{K} p(y_k | n_k),
\end{aligned} \tag{2.2}
$$

where the three components of 2.2 are called the initial distribution ($p(n_1)$), the transition model ($p(n_k|n_{k-1})$), and the observation model ($p(y_k|n_k)$). When the transition model and observation model are the same for all k, the HMM is said to be "stationary" or "homogenous".

Note that in the work that follows, we will focus exclusively on HMMs where the latent variables and observed variables are all discrete (and particularly on models where their support is infinite), but the general HMM is equivalently defined whether the variables are continuous or discrete.

2.2 Inference and Learning in Hidden Markov Models

The inference and learning approaches based on probability generating functions that we develop later in this thesis are developed for hidden Markov models (HMMs). We believe the techniques apply more broadly, but to understand the material presented here, it will suffice to understand the variety of inference and learning techniques in use today for HMMs.

2.2.1 The forward-backward algorithm

The forward-backward algorithm is a powerful "message-passing" exact inference algorithm for HMMs. It is a special case of the more general belief propagation (BP) algorithm (Pearl, 1986) which is an exact inference algorithm for trees and other directed acyclic graphs. We describe HMMs in detail in Section 2.1, but here we focus on inference. In particular, the forward-backward algorithm computes the following quantities in an HMM: the "filtered" marginals $p(n_k|\mathbf{y}_{1:k})$, the "smoothed"

marginals $p(n_k|\mathbf{y}_{1:K})$, and the likelihood $p(\mathbf{y}_{1:K}|\theta)$ (where θ is the collection of all model parameters).

The forward-backward algorithm is comprised of two "sub-algorithms": the forward algorithm and the backward algorithm. The forward algorithm recursively computes each of the filtered marginals $p(n_k|\mathbf{y}_{1:k})$ for $k = 1$, $k = 2$, and so on in a "left to right" pass. Then the backward algorithm recursively computes each of the smoothed marginals $p(n_k|\mathbf{y}_{1:K})$ for $k = K - 1$, $k = K - 2$, and so on in a "right to left" pass.

2.2.1.1 Forward algorithm

The forward algorithm recursively computes "messages", which are unnormalized distributions of subsets of the variables. Specifically, define the "forward" messages: $\alpha_k(n_k) := p(n_k, \mathbf{y}_{1:k})$ as the joint distribution of the kth latent variable and the first k observed variables. These satisfy the recurrence:

$$\alpha_1(n_1) = p(y_1 \,|\, n_1)p(n_1), \tag{2.3}$$

$$\alpha_k(n_k) = p(y_k \,|\, n_k) \sum_{n_{k-1}} \alpha_{k-1}(n_{k-1})p(n_k \,|\, n_{k-1}), 1 < k \leq K. \tag{2.4}$$

This is the traditional definition of the forward messages, but later in this document we will find it useful to split this recurrence into two steps by defining a set of intermediate messages: $\gamma_k(n_k) := p(n_k, \mathbf{y}_{1:k-1})$. The recurrence then becomes:

$$\gamma_k(n_k) = \sum_{n_{k-1}} \alpha_{k-1}(n_{k-1})p(n_k \,|\, n_{k-1}), \tag{2.5}$$

$$\alpha_k(n_k) = \gamma_k(n_k)p(y_k \,|\, n_k). \tag{2.6}$$

We will refer to Equation (2.5) as the *prediction step* (the value of n_k is predicted based on the observations $y_{1:k-1}$), and Equation (2.6) as the *evidence step* (the new evidence y_k is incorporated). In finite models, the forward algorithm can compute the α_k messages for $k = 1, \ldots, K$ directly using Equations (2.5) and (2.6).

The α-messages can also be used to compute the data likelihood as follows:

$$p(\mathbf{y}_{1:K}) = \sum_{n_K} \alpha_K(n_K) = \sum_{n_K} p(n_K, \mathbf{y}_{1:K}) \tag{2.7}$$

2.2.1.2 Backward algorithm

Where the forward algorithm operates recursively from "left-to-right", from $k = 1$ to K, the backward algorithm operates recursively from "right-to-left", from $k = K$ to 1. Like the forward algorithm, the backward algorithm computes another set of messages, the β messages which are defined as:

$$\beta_k(n_k) := p(n_k, \mathbf{y}_{k+1:K}), \tag{2.8}$$

and which satisfy the following recurrence:

$$\beta_k(n_k) = \sum_{n_{k+1}} \beta_{k+1}(n_{k+1}) p(y_{k+1}|n_{k+1}) p(n_{k+1}|n_k). \tag{2.9}$$

After computing the α and β messages, the "smoothed" posterior marginals $\mu_k(n_k) := p(n_k, \mathbf{y}_{1:K})$ can be derived by $\mu_k(n_k) \propto \alpha_k(n_k)\beta_k(n_k)$.

2.2.2 Variable elimination

A related family of exact inference algorithms, which we explain here in the context of discrete HMMs, is variable elimination (VE). Variable elimination is defined for undirected graphical models, though it can be applied to directed graphical models (like an HMM) by converting the directed graph to an undirected one via a process called "moralization". For tree-structured graphs, moralization simply involves replacing the directed edges between pairs of nodes with undirected edges and then defining pairwise factors between each pair of connected nodes. For HMMs, the factors would be:

- $\psi(n_1)$,
- $\psi(n_k, n_{k-1}), 1 < k \leq K$,
- and $\psi(y_k, n_k), 1 \leq k \leq K$.

In moralization, the factors are defined such that the full joint distribution (continuing to use HMMs as an example) is factored as:

$$p(\mathbf{n}_{1:K}, \mathbf{y}_{1:K}) = \psi(n_1) \prod_{i=2}^{K} \psi(n_i, n_{i-1}) \prod_{j=1}^{K} \psi(n_j, y_j).$$

From this factorization, we can compute various marginal distributions by marginalizing out any subset of the latent variables (the observed variables $\mathbf{y}$ are fixed and thus do not need to be marginalized). For instance, we could compute the posterior marginal distribution of the kth latent variable as:

$$p(n_k, \mathbf{y}_{1:K}) = \sum_{n_1} \sum_{n_2} \cdots \sum_{n_{k-1}} \sum_{n_{k+1}} \cdots \sum_{n_K} \psi(n_1) \prod_{i=2}^{K} \psi(n_i, n_{i-1}) \prod_{j=1}^{K} \psi(n_j, y_j). \tag{2.10}$$

If each of the latent variables has M possible values, then marginalizing all the latent variables this way will take $\mathcal{O}(M^{K-1})$ time, which very quickly becomes intractable. Variable elimination addresses this by refactoring the equation above by iteratively "eliminating" one variable at a time. The process of "eliminating" a variable consists of the following steps, which we demonstrate for the process of eliminating n_1 from Equation (2.10):

1. reorder the summation operators so that the variable to be eliminated is last:

$$\sum_{n_2} \cdots \sum_{n_{k-1}} \sum_{n_{k+1}} \cdots \sum_{n_K} \sum_{n_1} \psi(n_1) \prod_{i=2}^{K} \psi(n_i, n_{i-1}) \prod_{j=1}^{K} \psi(n_j, y_j),$$

2. move all terms that do not include the variable to be eliminated outside of the innermost sum:

$$\sum_{n_2} \cdots \sum_{n_{k-1}} \sum_{n_{k+1}} \cdots \sum_{n_K} \prod_{i=3}^{K} \psi(n_i, n_{i-1}) \prod_{j=2}^{K} \psi(n_j, y_j) \sum_{n_1} \psi(n_1)\psi(n_2, n_1)\psi(y_1, n_1),$$

3. define a new "temporary" factor containing all the factors with the variable to be eliminated:

$$\tau_1'(n_1, n_2) := \psi(n_1)\psi(n_2, n_1)\psi(y_1, n_1),$$

   where the subscript of the new factor corresponds to the order in which we eliminate the variables,

4. marginalize the variable to be eliminated out of the new factor to define a new factor:

$$\tau_1(n_2) := \sum_{n_1} \psi(n_1)\psi(n_2, n_1)\psi(y_1, n_1),$$

5. substitute the new factor back into the original factored distribution:

$$\sum_{n_2} \cdots \sum_{n_{k-1}} \sum_{n_{k+1}} \cdots \sum_{n_K} \prod_{i=3}^{K} \psi(n_i, n_{i-1}) \prod_{j=2}^{K} \psi(n_j, y_j)\tau_1(n_2).$$

This process is then repeated for each of the variables to be eliminated, eventually yielding the desired marginal distribution. The efficiency of variable elimination depends on the order that the variables are eliminated.

For HMMs in particular, the forward algorithm is equivalent to performing variable elimination in the order $n_1, \ldots, n_K$ and the backward algorithm is equivalent to performing variable elimination in the order $n_K, \ldots, n_1$. To compute the smoothed marginals ($p(n_k \,|\, \mathbf{y}_{1:K})$), a naive approach would be to run variable elimination with the order $n_1, \ldots, n_{k-1}, n_K, n_{K-1}, \ldots, n_{k+1}$. However, to compute all K smoothed marginals would require K variable elimination passes with this naive approach. The

forward-backward algorithm avoids this by reusing components of the individual forward and backward algorithms. This is also the general principle of message-passing or belief propagation in tree-structured graphs: all marginals are computed using two variable elimination passes.

In a later chapter we will run variable elimination with a non-standard elimination order for HMMs as a way to compute smoothed marginals. This is because we do not curently know how to implement the full forward-backward algorithm with PGFs.

2.2.3 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods can also be applied to perform inference in HMMs, though they are arguably less commonly used for HMMs than message-passing algorithms like the forward-backward algorithm. The intuition behind MCMC methods is to perform a Markovian random walk over the space of latent variables, model parameters, or some other state space of interest in such a way that the proportion of time spent in each state $\mathbf{x}$ is proportional to some target density $p^*(\mathbf{x})$. For HMMs, the state space could for instance consist of the set of all latent variables, $\mathbf{n}_{1:K}$ and the target density could be the joint posterior distribution of the latent variables $p(\mathbf{n}_{1:K} \mid \mathbf{y}_{1:K})$. The collection of (correlated) samples generated over this random walk can then be used to estimate statistics of the true density.

The simplest MCMC method for HMMs and likely the most widely used MCMC method in the population dynamics community is known as Gibbs sampling. In a Gibbs sampler, each variable x_i is repeatedly sampled individually and conditioned on the most recently sampled value of all other variables x_{-i}, which are kept fixed. We use the notation x_i^j to denote the jth sample of the ith variable and $\mathbf{x}^j$ to denote the set of the jth sample of all variables. In the simplest Gibbs sampler where variables are resampled in a fixed sequence, this means we will resample each of the K variables once after every K iterations of the sampler. In this way, we'll begin

at some initial point $\mathbf{x}^0$ and repeatedly take a step along one axis i according to the conditional distribution $p(x_i|\mathbf{x}_{-i})$. So long as each axis/variable is selected with equal frequency/probability, the Gibbs sampler will eventually converge to the stationary distribution, i.e. $\lim_{n\to\infty} \mathbf{x}_n \sim p^*(\mathbf{x})$ (Geman & Geman, 1984; Gelfand & Smith, 1990). It is worth noting that typically the full conditional distribution $p(x_i|\mathbf{x}_{-i})$ can be simplified by conditioning only on the Markov blanket of x_i.

For HMMs, one simple form of a Gibbs sampler samples each of the latent variables in sequence, beginning with n_1 and continuing in an order such as: $\{n_1^1, n_2^1, n_3^1, \ldots, n_K^1, n_1^2, n_2^2, \ldots\}$. In an HMM, the Markov blanket of each latent variable has at most three other variables (the preceeding latent variable, the subsequent latent variable, and the corresponding observed variable) and the full conditional for the ith latent variable can be written:

$$p(n_i \,|\, \mathbf{n}_{-i}, \mathbf{y}) = p(n_i \,|\, n_{i-1}, n_{i+1}, y_i), \tag{2.11}$$

$$\propto p(n_i \,|\, n_{i-1}) p(n_i \,|\, n_{i+1}) p(n_i \,|\, y_i). \tag{2.12}$$

and so, when sampling sequentially, the jth sample of the ith latent variable is distributed as:

$$n_i^j \sim p(n_i^j \,|\, n_{i-1}^j, n_{i+1}^{j-1}, y_i).$$

Because each sample is conditioned only on the latest sample and not on the full history of samples, the sequence of samples produced by a Gibbs sampler or other MCMC sampler form a Markov chain. Note that this Markov chain of samples is unrelated to the Markov chain that forms the backbone of the HMM.

Given a set of M complete samples from a Gibbs sampler, we can estimate the target distribution by:

$$\tilde{p}(x_i = z) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{1}(x_i^j = z),$$

where $\mathbb{1}(e)$ is the indicator function, i.e. $\mathbb{1}(e) = 1$ if the expression e is true and $\mathbb{1}(e) = 0$ otherwise.

2.2.4 Learning in HMMs

One of the most common tasks in machine learning is parameter estimation: given a model specification and a set of training data, fit values for each of the model parameters that are consistent with the training data. There are a number of ways to do this, in this section we focus specifically on parameter estimation for HMMs and the case where ground-truth data for the latent variables is unavailable.

2.2.4.1 Maximum likelihood estimation

As discussed previously, the likelihood function $\mathcal{L}(\mathcal{D}|\theta) := p(\mathcal{D}|\theta)$ defines the probability of the observed data $\mathcal{D}$ given a particular configuration of the model parameters θ. Given some data, a set of values for the observed variables, the corresponding maximum likelihood estimate (MLE) parameters θ^{MLE} are then the configuration of θ which maximizes the likelihood of our data:

$$\theta^{\text{MLE}} := \arg\max_{\theta} \mathcal{L}(\mathcal{D}|\theta).$$

In the case of HMMs, the likelihood function is computed using the forward algorithm (see Equation (2.7)).

In practice, computing the MLE parameters by optimizing the likelihood is non-trivial. In general, it is not possible to optimize the likelihood explicitly, i.e. there is no closed form expression for θ^{MLE} and we must instead optimize the likelihood numerically. In the chapters that follow, we generally use standard numerical optimization methods from scientific computing packages, but a commonly used alternative for HMMs is the Baum-Welch algorithm, which we discuss below.

2.2.4.2 The Baum-Welch algorithm

For HMMs, one of the most common approaches to finding the MLE parameters is the Baum-Welch algorithm (Baum et al., 1970). For completeness, we review the algorithm here, but it will not be referenced elsewhere in this document.

The Baum-Welch algorithm is a special case of the Expectation-Maximization (EM) (Dempster et al., 1977) algorithm for HMMs. The EM algorithm in general is an iterative algorithm that alternates between two steps: first the expectation step (or E step), then the maximization step (or M step), then the E step again, then the M step, and so on. At any given time in the EM algorithm, we maintain an estimate θ^* of the model parameters. In the E step, we compute the expected value of the sufficient statistics of the model given the current estimate of the model parameters θ^* and then in the M step, we update our estimate of the parameters θ^* by maximizing the likelihood given all the observed variables and the expected value of the sufficient statistics. This is generally repeated until the parameter estimates converge.

The Baum-Welch algorithm is an efficient application of EM to HMMs with finite discrete-valued latent variables. In the Baum-Welch algorithm, the expected value of the sufficient statistics in the E step is computed from the smoothed posterior marginals by performing the forward-backward algorithm on the HMM. Given the expectation of the sufficient statistics, each of the terms of the initial state distribution, the transition matrix, and the emission distribution can then be easily estimated in the M step by setting them equal to their normalization realizations in the expected complete data.

Note that Baum-Welch works only when the latent variables have finite support and when the various distributions are all multinomial, though the more general EM framework is well defined (but as yet unimplementable) for HMMs with non-finite latent variables and arbitrary initial distributions, transition distributions, and emission distributions.

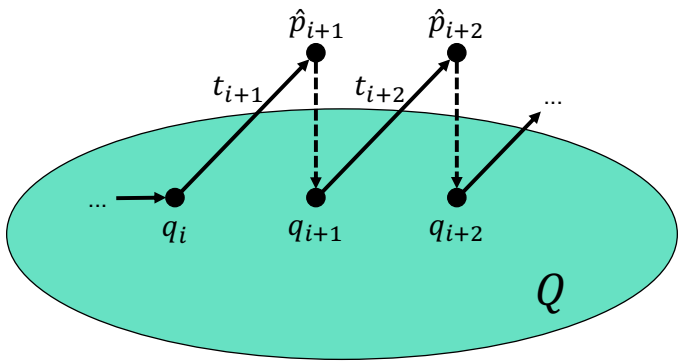

Figure 2.2. The assumed density filtering algorithm. A target distribution p is factored into a product of components $p = \prod_i t_i$. The intermediate distribution q_j is defined recursively as the product of the jth component t_j and the previous intermediate distribution q_{j-1}. After computing each intermediate distribution q_j, the distribution is projected back to a simple approximating family $\mathcal{Q}$.

2.2.5 Assumed density filtering

Assumed density filtering (ADF) is an approximate inference scheme for distributions where the true distribution $p(\theta)$ can be factored into a product of simple terms:

$$p(\theta) \propto \prod_{i=1}^{K} t_i(\theta). \tag{2.13}$$

The main idea in ADF is to define a simple approximating family $\mathcal{Q}$ (typically an exponential family) which is "simpler" than the true distribution $p(\theta)$. Beginning with an initial distribution $q_0(\theta) \in \mathcal{Q}$, we will proceed by incorporating the first t_1 factor into q_0 (the "update" step), projecting back to $\mathcal{Q}$ to get q_1 (the "projection" step), and so on for each t_i. This is demonstrated graphically in Figure 2.2.

The update step is defined as:

$$\hat{p}_i(\theta) = \frac{t_i(\theta) q_{i-1}(\theta)}{Z_i}, \tag{2.14}$$

where $Z_i = \int_\theta t_i(\theta) q_{i-1}(\theta) d\theta$ is the (optional) normalizer. The projection step from $\hat{p}_i(\theta)$ to $\mathcal{Q}$ is defined as:

$$q_i = \arg\min_{q \in \mathcal{Q}} \mathrm{D}_{\mathrm{KL}}(\hat{p}_i(\theta) || q(\theta)). \tag{2.15}$$

If $\mathcal{Q}$ is an exponential family, then the KL minimization for the projection step can be performed via moment matching. For instance, if $\mathcal{Q}$ is a family of spherical Gaussian distributions, then q_i is given by the following expectation constraints Minka (2001):

$$\begin{aligned} E_{q_i}[\theta] &= E_{\hat{p}_i}[\theta], \\ E_{q_i}[\theta^\intercal \theta] &= E_{\hat{p}_i}[\theta^\intercal \theta]. \end{aligned} \tag{2.16}$$

For tree-structured models such as HMMs, message-passing algorithms can be seen as a special case of ADF (Heskes & Zoeter, 2003). In particular, when the model is such that all messages remain in the exponential family, the forward algorithm for HMMs is a (non-approximate, trivial) special case of ADF. When messages do not remain in the exponential family, we can also define an approximate forward algorithm with ADF as follows. To begin, recall that the forward algorithm defines a factorization of $p(\mathbf{n}, \mathbf{y})$:

$$\begin{aligned} p(\mathbf{n}, \mathbf{y}) &= \prod_{1 < i \leq K} \psi_i(n_{i-1}, n_i, y_i), \\ \psi_1(n_1, n_0, y_1) &\propto p(n_1) p(y_1 | n_1), \\ \psi_i(n_i, n_{i-1}, y_i) &\propto p(y_i | n_i) \sum_{n_{i-1}} p(n_i | n_{i-1}), \quad i > 1. \end{aligned}$$

Each update step incorporates the next $\psi_i(n_i, n_{i-1}, y_i)$ term as before (which corresponds to one iteration of the forward algorithm):

$$\hat{p}_i(\mathbf{n}_{1:i}, \mathbf{y}_{1:i}) = \hat{p}_{i-1}(\mathbf{n}_{1:i-1}, \mathbf{y}_{1:i-1}) \cdot \psi_i(n_i, n_{i-1}, y_i). \tag{2.17}$$

By marginalizing out all variables except n_i from $\hat{p}_i(\mathbf{n}_{1:i}, \mathbf{y}_{1:i})$, we derive the following approximation of the corresponding filtered marginal over n_i:

$$m_i(n_i) \propto \sum_{\mathbf{n}_{1:i-1}} \hat{p}_i(\mathbf{n}_{1:i}, \mathbf{y}_{1:i}). \tag{2.18}$$

We then project the filtered marginal $m_i(n_i)$ back to $\mathcal{Q}$ by:

$$\hat{m}_i = \arg\min_{m \in \mathcal{Q}} \mathrm{D}_{\mathrm{KL}}(m_i \parallel m). \tag{2.19}$$

Subsequent update steps may then use the approximate filtered marginals $\hat{m}_i$ in place of $\hat{p}_i$:

$$\hat{p}_i(\mathbf{n}_{1:i}, \mathbf{y}_{1:i}) = \hat{m}_{i-1}(n_{i-1}) \cdot \psi_i(n_i, n_{i-1}, y_i). \tag{2.20}$$

In Chapter 6, we will develop an approximation algorithm that is similar to ADF in the PGF domain for HMMs. However we do not know whether our approximating family is an exponential family and so our projection step is subtly different from that used in ADF. More details on the relationship between the two algorithms are in Section 6.1.1.

2.3 Distribution Representation

Throughout this work, we will be addressing the challenge of *representation* for discrete distributions. Classically, there are a number of ways to represent a discrete distribution. Given a parametric form of the distribution, such as $X \sim \text{Poisson}(\lambda)$, we can specify the entire distribution by its parameters (in the Poisson case λ).

For arbitrary discrete distributions, we can also represent the terms of the probability mass function explicitly as a vector/matrix/tensor. Indeed, this is commonly the representation used when working with discrete distributions in algorithms such as the forward-backward algorithm (see Section 2.2.1). However, as we explore in much greater detail later, this approach is not applicable for arbitrary discrete distributions with infinite support. To represent such a distribution, one approach from the ecology literature Royle (2004); Dail & Madsen (2011) has been to truncate the

support of the distribution to some finite range. Throughout this work we will commonly use TRUNC to refer specifically to the forward algorithm using such a truncated representation.

In this work, we adopt probability generating functions (PGFs) for inference, which we discuss below:

2.3.1 Probability generating functions

Let $x = (x_1, \ldots, x_d)$ be a vector of nonnegative integer-valued random variables where $x_i \in \mathcal{X}_i \subseteq \mathbb{N}^0$. The set $\mathcal{X}_i$ may be finite (e.g., to model binary or finite discrete variables), but we assume without loss of generality that $\mathcal{X}_i = \mathbb{N}^0$ for all i by defining factors to take value zero for integers outside of $\mathcal{X}_i$. For any set $\alpha \subseteq \{1, \ldots, d\}$, define the subvector $x_\alpha := (x_i, i \in \alpha)$. We consider probability models of the form

$$p(x) = \frac{1}{Z} \prod_{\alpha \in \mathcal{A}} \psi_\alpha(x_\alpha),$$

where Z is a normalization constant and $\{\psi_\alpha\}$ is a set of factors $\psi_\alpha : [\mathbb{N}^0]^{|\alpha|} \to \mathbb{R}^+$ indexed by subsets $\alpha \subseteq \{1, \ldots, d\}$ in a collection $\mathcal{A}$.

A general factor ψ_α on integer-valued variables cannot be finitely represented. We instead use the formalization of probability generating functions. Let $s = (s_1, \ldots, s_d)$ be a vector of indeterminates corresponding to the random variables x. The joint PGF of a factor ψ_α is

$$F_\alpha(s_\alpha) = \sum_{x_\alpha} \psi_\alpha(x_\alpha) \cdot \prod_{i \in \alpha} s_i^{x_i} = \sum_{x_\alpha} \psi_\alpha(x_\alpha) \cdot s_\alpha^{x_\alpha}.$$

Here, for two vectors a and b with the same index set $\mathcal{I}$, we have defined $a^b = \prod_{i \in \mathcal{I}} a_i^{b_i}$. The sum is over all vectors x_α of non-negative integers.

Univariate PGFs of the form $F(s) = \sum_{x=0}^{\infty} \Pr(X = x) s^x = \mathbb{E}[s^X]$, where X is a nonnegative integer-valued random variable, are widely used in probability and

statistics (Resnick, 2013; Casella & Berger, 2002), and have a number of nice properties. A PGF uniquely encodes the distribution of X, and there are formulas to recover moments and entries of the the probability mass function from the PGF. Most common distributions have closed-form PGFs, e.g., $F(s) = \exp\{\lambda(s-1)\}$ when $X \sim \text{Poisson}(\lambda)$. Similarly, the joint PGF F_α uniquely encodes the factor ψ_α, and we will develop a set of useful operations on joint PGFs. Note that we abuse terminology slightly by referring to the generating function of the factor ψ_α as a *probability* generating function; however, it is consistent with the view of ψ_α as an unnormalized probability distribution.

2.4 Automatic Differentiation

In Chapter 6, we will demonstrate a relationship between PGF inference and function evaluation and show how techniques from forward-mode automatic differentiation (autodiff) (Griewank et al., 2000; Griewank & Walther, 2008) can be applied to perform a black-box version of PGF inference. Here we will review forward-mode autodiff, based principally on the presentation of (Griewank & Walther, 2008) and (Baydin et al., 2018). Reverse-mode autodiff, while widely used in the machine learning community, does not play a role in the work presented here and so we exclude it from discussion. For a broader review of autodiff, including reverse-mode, see (Baydin et al., 2018).

2.4.1 Forward-mode autodiff

Often in machine learning we have a complex function $f(x)$ and we are interested in knowing some derivative of f, $\frac{d}{dx}f(x)$. Historically, there are two commonly used classes of approach to this kind of differentiation: symbolic differentiation (either by manually working out the derivatives or by employing a symbolic differentiation

library/tool) and numerical differentiation (by finite difference approximation). Automatic differentiation is distinct from both.

Like numerical differentiation, autodiff does not yield a general symbolic expression for the requested derivative. Instead, it evaluates the derivative at a fixed point/set of points. However, like symbolic differentiation, it is an exact method, computing the correct value of the requested derivatives. Finally, as the name implies and similar to both symbolic libraries/tools and numerical differentiation, autodiff is a more-or-less "black-box" method: the user need only provide an implementation of their function $f(x)$ and autodiff takes it from there.

The core concept of forward-mode automatic differentiation is a process we call "function lifting". If $h(x)$ is an elementary function (e.g. binary addition and multiplication, unary power, exp, etc.) then the "lifted" version of $h(x)$ is the function that simultaneously implements the original elementary operation as well as the derivative of that operation. Other functions that are ultimately comprised of some combination of elementary operations can similarly be lifted by applying the chain rule through each of the component operations in a "forward accumulation" pass.

This "lifting" process is typically performed in practice using a data structure known as a "dual number". A dual number is a first-degree Taylor series with the following form:

$$x + x'\epsilon,$$

where x is a scalar representing the original value, x' is the derivative of x with respect to a fixed input variable, and ϵ is a new formal variable denoting a perturbation of the input variable. We will use the notation $\langle x, x' \rangle$ to denote dual numbers. Conveniently, dual number arithmetic efficiently lifts many elementary operations. As a result, dual numbers (as truncated Taylor polynomials) are often the base representation in modern autodiff libraries. Function lifting is typically achieved by overloading elementary operations with their lifted equivalents. In this way, even a complex

function can be lifted automatically so long as it is ultimately comprised only of elementary operations that are overloaded by the autodiff library.

For example, suppose we have a function in our system like $y = \sin(z)$ where z is some intermediate variable in the program and y and z are scalars. If we want to know the derivatives of y with respect to some input variable x: $\frac{dy}{dx}$, we can lift sin to operate on a dual number: $\mathcal{L}(\sin)(\langle z, z' \rangle)$ where $z' = \frac{dz}{dx}$ and $\mathcal{L}$ is the "lift operator" that lifts a function to operate on dual numbers. For primitive operations, the lifted equivalents are well known (see Griewank & Walther, 2008) and nested/combined primitives can be joined according to similar principles.

The general principles of forward-mode autodiff can also be naturally extended to compute high-order derivatives and jacobians, a topic which we explore in more detail in Chapter 6.

CHAPTER 3

MODELS FOR LATENT COUNTS

In this work, we focus on models containing latent variables whose support is discrete and countably infinite. Most commonly, these variables are "count-valued", meaning more specifically that their support is $\mathbb{N}^0$, the set of non-negative integers. We call these variables "count variables". Such variables could represent the size of a population, the frequency of an event, and many other discrete variable whose support is positive and unbounded.

Models with latent count-valued variables arise commonly as partially observed time series, wherein the latent count variables represent something like the size of a indirectly observed population or a time-varying event frequency. In our case, we are interested in modeling the size of a population that may be changing over time by expressing the population dynamics of the system. In this chapter, we review commonly used latent count models for population dynamics which can be seen as special cases of the hidden Markov model (HMM), the more general family of graphical models which we presented in Section 2.1. In 3.2, we present a formulation of a novel family of population dynamics models that generalizes and extends existing models and which will serve as the primary example in subsequent chapters.

3.1 Models for Population Dynamics of Unmarked Individuals

The work described in this thesis builds directly on prior work on the dynamics of "closed" and "open" populations in the ecological literature. The models used

previously in this line of research can be shown to be special cases of the more general model we introduce in Section 3.2.

Notably all of the models presented here contain latent count variables (discrete latent variables whose support is countably infinite), meaning that traditional inference algorithms cannot be applied directly. As a result, previous applications of these models have always been forced to include a small but significant alteration to the model: truncating the support of the latent variables to some finite value $N_{\max}$. This is not intended to be a mechanistic component of the models and indeed is left out of the model descriptions of the work where those models are developed but is instead a concession to the needs of implementability. The impact of this implicit modeling assumption is discussed in much greater detail throughout the main chapters of this thesis, where we refer to it as the "truncated method" or, when applied specifically to the forward algorithm, as TRUNC.

3.1.1 Closed populations

The N-mixture model, presented originally by Royle (2004), is a statistical model for "closed" populations, i.e. populations where individuals are assumed not to enter or leave the population over the survey period. In it, repeated observations are conducted of a population via transect or point count. The true abundance n is assumed to be a latent variable whose distribution (called the mixture distribution) is either Poisson or negative binomial. The observed variables $y_k \in \{y_1, \ldots, y_K\}$ are iid binomial samples of n. The two possible model configurations originally proposed in (Royle, 2004) are given below and described graphically in Figure 3.1:

$$n \sim \text{Poisson}(\lambda) \qquad\qquad n \sim \text{NegBin}(\alpha, r)$$

$$y_k \sim \text{Binomial}(n, \rho) \qquad\qquad y_k \sim \text{Binomial}(n, \rho)$$

However, these models are based fundamentally on the "closure" assumption: that the observations are conducted on time scale over which the abundance is unlikely to have changed. Therefore the utility of the N-mixture model for modeling population

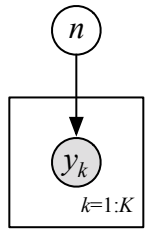

Figure 3.1. The N-mixture model for closed populations. The latent variable n represents the true abundance of a population and does not change between observation events. The observed variables y_k are iid and represent the observed count at each sampling occasion.

dynamics (wherein the population size is presumably changing over time) is necessarily limited. In order to measure the effects of population dynamics, we must move to models for "open" populations.

3.1.2 Open populations

In (Dail & Madsen, 2011), the authors introduced an extension of the N-mixture model for "open" populations in which the size of the population is allowed to change in between the surveys that are conducted. This model, hereafter referred to as the "Dail-Madsen" model, assumed that while the abundance may vary between observations, the sequence of latent abundances satisfies the Markov property. In other words, the size of the population at time $t + 1$ depends only on the size of the population at time t.

This key assumption leads to the graphical representation of the Dail-Madsen model shown in Figure 3.2, which may be recognized as a hidden Markov model (HMM). All that remains is to specify the initial distribution of abundance $p(n_1)$, the transition dynamics $p(n_k|n_{k-1})$, and the observation dynamics $p(y_k|n_k)$. In their original presentation of the model, Dail and Madsen factored the transition dynamics into two components: survival (which they assume to be distributed as a binomial) and recruitment (which they assume to be distributed as a Poisson). These transition

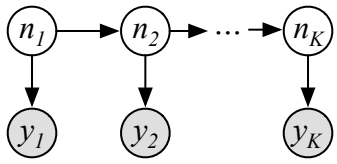

Figure 3.2. The Dail-Madsen model for open populations. The latent variables n_i are discrete valued and represent the true population size at each sampling occasion and the observed variables y_i represent the observed count at each sampling occasion.

dynamics are a special case of the Latent Branching Process model we describe in more detail below. Finally, the Dail-Madsen mdoel assumes that the observation process is binomial and that the initial distribution is Poisson or negative binomial.

In the following years, numerous authors have extended the Dail-Madsen model by introducing new transition dynamics (Hostetler & Chandler, 2015), introducing explicit spatial interactions (Chandler et al., 2011, 2013), and modeling demographic dynamics (Zipkin et al., 2014).

3.2 Latent Branching Process

We consider a hidden Markov model with integer latent variables $n_1, \ldots, n_K$ and integer observed variables $y_1, \ldots, y_K$. All variables are assumed to be non-negative. The model is most easily understood in the context of its application to population ecology or branching processes (which are similar): in these cases, the variable n_k represents the size of a hidden population at time t_k, and y_k represents the number of individuals that are observed at time t_k. However, the model is equally valid without this interpretation as a flexible class of autoregressive processes (McKenzie, 2003).

We introduce some notation to describe the model. For an integer random variable n, write $y = \rho \circ n$ to mean that $y \sim \text{Binomial}(n, \rho)$. This operation is known as "binomial thinning": the count y is the number of "survivors" from the original count n. We can equivalently write $y = \sum_{i=1}^{n} x_i$ for iid $x_i \sim \text{Bernoulli}(\rho)$ to highlight the fact that this is a compound distribution. Indeed, compound distributions will play

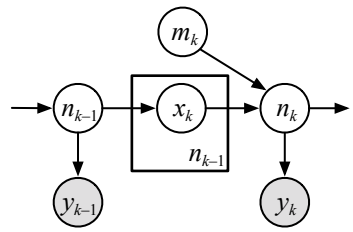

Figure 3.3. The latent branching process model. This model describes a hidden Markov model with a particular set of transition dynamics. The observed variables y_k represent a noisy survey of each n_k, the latent population size at each time. The transition dynamics are split into two parts: an offspring process represented by x_k and an independent immigration process represented by m_k.

a key role: for independent integer random variables n and x, let $z = n \odot x$ denote the compound random variable $z = \sum_{i=1}^{n} x_i$, where $\{x_i\}$ are independent copies of x.

Our model, the "latent branching process" (LBP) is shown graphically in Figure 3.3 and is described concretely by:

$$n_k = m_k + \sum_{i=1}^{n_{k-1}} x_{k-1,i}, \tag{3.1}$$

$$y_k \sim \text{Binomial}\, n_k, \rho_k. \tag{3.2}$$

The variable n_k represents the population size at time t_k. The random variable $x_{k-1} = \sum_{i=1}^{n_{k-1}} x_{k-1,i}$ is the number of *offspring* of individuals from the previous time step, where $x_{k-1,i}$ is the total number of individuals "caused by" the ith individual alive at time t_{k-1}. The collection of $x_{k-1,i}$ variables at each time step (i.e. $\{x_{k-1,1}, x_{k-1,2}, \dots\}$) being iid. This definition of offspring is flexible enough to model immediate offspring, surviving individuals, and descendants of more than one generation. The random variable m_k is the number of immigrants at time t_k, and y_k is the number of individuals observed at time t_k, with the assumption that each individual is observed independently with probability ρ_k. We have left unspecified the distributions of m_k and $x_{k,i}$, which we term the *immigration* and *offspring* distributions,

respectively. These may be arbitrary distributions over non-negative integers. We will assume the initial condition $n_0 = 0$, though the model can easily be extended to accommodate arbitrary initial distributions.

3.2.1 Connections to Other Models

This model specializes to capture many different models in the literature. The latent process of Eq. (3.1) is a Galton-Watson branching process with immigration (Watson & Galton, 1875; Heathcote, 1965). It also captures a number of different $AR(1)$ (first-order autoregressive) processes for integer variables (McKenzie, 2003); these typically assume that the offspring process is binomial thinning of the current individuals, i.e., $X_k \sim \text{Bernoulli}(\delta_k)$. For clarity when describing this as an offspring distribution, we will refer to it as *Bernoulli offspring*. With Bernoulli offspring and time-homogenous Poisson immigration, the model is an $M/M/\infty$ queue (McKenzie, 2003); with time-varying Poisson immigration it is an $M_t/M/\infty$ queue (Eick et al., 1993).

Many of the models in Section 3.1 are special cases of the LBP. When immigration is zero after the first time step and $x_k = 1$, the population size is a fixed random variable, and we recover the N-mixture model of (Royle, 2004). With Poisson immigration and Bernoulli offspring, we recover the basic model of (Dail & Madsen, 2011) for open metapopulations. In Chapter 7, we investigate how several other extensions of the "Dail-Madsen" model fit into our LBP framework. Finally, other related models for unmarked insect populations also fall within this framework (Zonneveld, 1991; Gross et al., 2007).

CHAPTER 4

PROBABILISTIC INFERENCE WITH GENERATING FUNCTIONS FOR POISSON LATENT VARIABLE MODELS

4.1 Introduction

A key reason for the success of graphical models is the existence of fast algorithms that exploit the graph structure to perform inference. For models with a simple enough graph structure, these algorithms can compute marginal probabilities exponentially faster than direct summation.

However, these fast exact inference methods apply only to a relatively small class of models—those for which the basic operations of marginalization, conditioning, and multiplication of constituent factors can be done efficiently. In most cases, this means that the user is limited to models where the variables are either discrete (and finite) or Gaussian, or they must resort to some approximate form of inference.

Why are Gaussian and discrete models tractable while others are not? The key issue is one of *representation*. If we start with factors that are all discrete or all Gaussian, then: (1) factors can be represented exactly and compactly, (2) conditioning, marginalization, and multiplication can be done efficiently in the compact representation, and (3) each operation produces new factors of the same type, so they can also be represented exactly and compactly.

Many models fail the restriction of being discrete or Gaussian even though they are qualitatively "easy". Section 3.1.1 provides a simple example, the *N-mixture* model (Royle, 2004) that is commonly used to interpret field surveys in ecology. The latent variable $n \sim \text{Poisson}(\lambda)$ represents the unknown number of individual

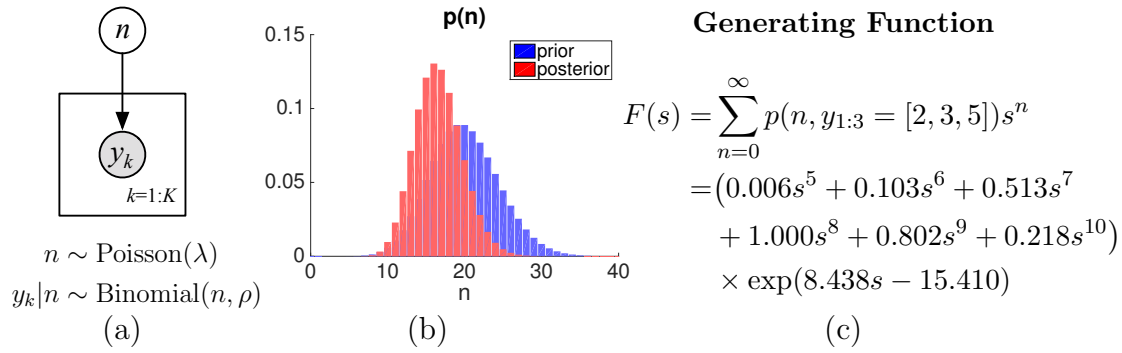

Figure 4.1. Generating functions for the N-mixture model. The N-mixture model (Royle, 2004) is a simple model with a Poisson latent variable for which no exact inference algorithm is known: (a) the model, (b) the prior and posterior for $\lambda = 20$, $\rho = 0.25, y_1 = 2, y_2 = 5, y_3 = 3$, (c) a closed form representation of the generating function of the unnormalized posterior, which is a compact and exact description of the posterior.

animals at a given site, which has support $\mathbb{N}^0$. Repeated surveys are conducted at the site during which the observer detects each individual with probability ρ, so each observation y_k is Binomial(n, ρ). From these observations (usually across many sites with shared λ), the scientist wishes to infer n and fit λ and ρ.

This model is very simple: all variables are marginally Poisson, and the unnormalized posterior has a simple form (e.g., see Figure 4.1b). However, until recently, there was no known algorithm to exactly compute the likelihood $p(y_{1:K})$. The naive way is to sum the unnormalized posterior $p(n, y_1, \ldots, y_K)$ over all possible values of n. However, n has a countably infinite support, so this is not possible. In practice, users of this and related models truncate the infinite sum at a finite value (Royle, 2004). A recent paper developed an exact algorithm for the N-mixture model, but one with running time that is exponential in K (Dennis et al., 2015). For a much broader class of models with **latent count variables** (Kéry et al., 2009; Fiske & Chandler, 2011; Chandler et al., 2011; Dail & Madsen, 2011; Zipkin et al., 2014), there are *no known exact inference algorithms*. Current methods either truncate the support (Fiske & Chandler, 2011; Chandler et al., 2011; Dail & Madsen, 2011), which

is slow and interacts poorly with parameter estimation (Couturier et al., 2013; Dennis et al., 2015), or use MCMC (Kéry et al., 2009; Zipkin et al., 2014), which is slow and for which convergence is hard to assess.

The key difficulty with these models is that we lack *finite and computationally tractable representations* of factors over variables with a countably infinite support, such as the posterior distribution in the N-mixture model, or intermediate factors in exact inference algorithms.

The main contribution of this chapter is to develop *compact* and *exact* representations of countably infinite factors using probability generating functions (PGFs) and to show how to perform variable elimination in the domain of generating functions. We provide the first exact pseudo-polynomial time inference algorithms (i.e., polynomial in the magnitude of the observed variables) for a subset of the latent branching process (LBP) model described in Section 3.2, which includes LBPs whose transition dynamics are Poisson. In this chapter, we refer to this subset of the LBP family as the "Poisson hidden Markov model" or Poisson HMM.

For example, the generating function of the unnormalized N-mixture posterior is shown in Figure 4.1c, from which we can *efficiently* recover the likelihood $p(y_1 = 2, y_2 = 5, y_3 = 3) = F(1) = 0.0025$. For Poisson HMMs, we first develop a PGF-based forward algorithm to compute the likelihood, which enables efficient parameter estimatation. We then develop a "tail elimination" approach to compute posterior marginals. Experiments show that our exact algorithms are much faster than existing approximate approaches, and lead to better parameter estimation.

4.1.1 Related work

Several previous works have used factor transformations for inference. Bickson & Guestrin (2010) show how to perform inference in the space of characteristic functions (see also (Mao & Kschischang, 2005)) for a certain class of factor graphs. Xue et al.

(2016) perform variable elimination in discrete models using Walsh-Hadamard transforms. Jha et al. (2010) use generating functions (over finite domains) to compute the partition function of Markov logic networks. McKenzie (2003) describes the use of PGFs in discrete time series models, which are related to our models except they are fully observed, and thus require no inference.

4.2 Variable Elimination with Generating Functions

Our approach to inference in models with latent count variables will be to implement the same abstract set of operations as variable elimination, but using a representation based on probability generating functions. In this chapter, we will demonstrate these techniques on "Poisson HMMs", a special case of the latent branching process (LBP) model from Section 3.2. The Poisson HMM is identical to the LBP except that it assumes that the arrival distribtution is always Poisson and that the offspring distribution is always Bernoulli, i.e.:

$$m_k \sim \text{Bernoulli}(\delta_k),$$
$$x_{k,i} \sim \text{Poisson}(\lambda_k).$$

However, because variable elimination will produce intermediate factors on larger sets of variables, and to prepare for later chapters where we will generalize these methods to a larger class of models, we will abstract for now from the Poisson HMM and employ notation general for graphical models with multivariate factors and their corresponding multivariate generating functions. Background on probability generating functions (PGFs) and their notation is in Section 2.3.1. A concrete application of these techniques to the Poisson HMM is presented beginning in Section 4.2.2.

4.2.1 Operations on Generating Functions

Our goal is to perform variable elimination using factors represented as PGFs. To do this, the basic operations we need to support are are multiplication, marginalization, and "entering evidence" into factors (reducing the factor by fixing the value of one variable). In this section we state a number of results about PGFs that show how to perform such operations. For the most part, these are either well known or variations on well known facts about PGFs (see Feller, 1968, Chapters 11, 12), which we present here for ease of reference. The proofs of all operations are collected at the end of the section.

First, we see that marginalization of factors is very easy in the PGF domain:

Proposition 4.1 (Marginalization)**.** *Let* $\psi_{\alpha\setminus i}(x_{\alpha\setminus i}) := \sum_{x_i \in \mathcal{X}_i} \psi_\alpha(x_{\alpha\setminus i}, x_i)$ *be the factor obtained from marginalizing* i *out of* ψ_α*. The joint PGF of* $\psi_{\alpha\setminus i}$ *is* $F_{\alpha\setminus i}(s_{\alpha\setminus i}) = F_\alpha(s_{\alpha\setminus i}, 1)$*. The normalization constant* $\sum_{x_\alpha} \psi_\alpha(x_\alpha)$ *is equal to* $F_\alpha(1, \ldots, 1)$*.*

Entering evidence is also straightforward:

Proposition 4.2 (Evidence)**.** *Let* $\psi_{\alpha\setminus i}(x_{\alpha\setminus i}) := \psi_\alpha(x_{\alpha\setminus i}, a)$ *be the factor resulting from observing the value* $x_i = a$ *in* ψ_α*. The joint PGF of* $\psi_{\alpha\setminus i}$ *is* $F_{\alpha\setminus i}(s_{\alpha\setminus i}) = \frac{1}{a!}\frac{\partial^a}{\partial s_i^a} F_\alpha(s_\alpha)\big|_{s_i=0}$*.*

Multiplication in the PGF domain—i.e., computing the PGF of the product $\psi_\alpha(x_\alpha)\psi_\beta(x_\beta)$ of two factors ψ_α and ψ_β—is not straightforward in general. However, for certain types of factors, multiplication is possible. We give two cases.

Proposition 4.3 (Multiplication: Binomial thinning)**.** *Let* $\psi_{\alpha\cup j}(x_\alpha, x_j) = \psi_\alpha(x_\alpha) \cdot \text{Binomial}(x_j | x_i, \rho)$ *be the factor resulting from expanding* ψ_α *to introduce a thinned variable* $x_j := \rho \circ x_i$*, where* $i \in \alpha$ *and* $j \notin \alpha$*. The joint PGF of* $\psi_{\alpha\cup j}$ *is* $F_{\alpha\cup j}(s_\alpha, s_j) = F_\alpha(s_{\alpha\setminus i}, s_i(\rho s_j + 1 - \rho))$*.*

Proposition 4.4 (Multiplication: Addition of two variables)**.** *Let* $\psi_\gamma(x_\alpha, x_\beta, x_k) := \psi_\alpha(x_\alpha)\psi_\beta(x_\beta)\mathbb{I}\{x_k = x_i + x_j\}$ *be the joint factor resulting from the introduction of a*

new variable $x_k = x_i + x_j$*, where* $i \in \alpha$*,* $j \in \beta$*,* $k \notin \alpha \cup \beta$*,* $\gamma := \alpha \cup \beta \cup \{k\}$*. The joint PGF of* ψ_γ *is* $F_\gamma(s_\alpha, s_\beta, s_k) = F_\alpha(s_{\alpha\setminus i}, s_k s_i) F_\beta(s_{\beta\setminus j}, s_k s_j)$*.*

The four basic operations above are enough to perform variable elimination on a large set of models. In practice, it is useful to introduce additional operations that combine two of the above operations.

Proposition 4.5 (Thin then observe)**.** *Let* $\psi'_\alpha(x_\alpha) := \psi_\alpha(x_\alpha) \cdot \text{Binomial}(a|x_i, \rho)$ *be the factor resulting from observing the thinned variable* $\rho \circ x_i = a$ *for* $i \in \alpha$*. The joint PGF of* ψ'_α *is* $F'_\alpha(s_\alpha) = \frac{1}{a!}(s_i\rho)^a \frac{\partial^a}{\partial t_i^a} F_\alpha(s_{\alpha\setminus i}, t_i)\Big|_{t_i = s_i(1-\rho)}$*.*

Proposition 4.6 (Thin then marginalize)**.** *Let* $\psi_{(\alpha\setminus i)\cup j}(x_{\alpha\setminus i}, x_j) := \sum_{x_i} \psi_\alpha(x_\alpha) \cdot \text{Binomial}(x_j|x_i, \rho)$ *be the factor resulting from introducing* $x_j := \rho \circ x_i$ *and then marginalizing* x_i *for* $i \in \alpha, j \notin \alpha$*. The joint PGF of* $\psi_{(\alpha\setminus i)\cup j}$ *is* $F_{(\alpha\setminus i)\cup j}(s_{\alpha\setminus i}, s_j) = F_\alpha(s_{\alpha\setminus i}, \rho s_j + 1 - \rho)$*.*

Proposition 4.7 (Add then marginalize)**.** *Let* $\psi_\gamma(x_{\alpha\setminus i}, x_{\beta\setminus j}, x_k) := \sum_{x_i, x_j} \psi_\alpha(x_\alpha) \cdot \psi_\beta(x_\beta) \mathbb{I}\{x_k = x_i + x_j\}$ *be the factor resulting from the deterministic addition* $x_i + x_j = x_k$ *followed by marginalization of* x_i *and* x_j*, where* $i \in \alpha, j \in \beta, k \notin \alpha \cup \beta, \gamma := (\alpha\setminus i) \cup (\beta\setminus j) \cup \{k\}$*. The joint PGF of* ψ_γ *is* $F_\gamma(s_{\alpha\setminus i}, s_{\beta\setminus j}, s_k) = F_\alpha(s_{\alpha\setminus i}, s_k) F_\beta(s_{\beta\setminus j}, s_k)$*.*

Proofs — Operations on Generating Functions

Proof of Proposition 4.1. This is a standard fact about multivariate PGFs:

$$F_\alpha(s_{\alpha\setminus i}, 1) = \sum_{x_\alpha} \psi_\alpha(x_\alpha) s_{\alpha\setminus i}^{x_{\alpha\setminus i}} 1^{x_i} = \sum_{x_{\alpha\setminus i}} \Big(\sum_{x_i} \psi_\alpha(x_{\alpha\setminus i}, x_i) \Big) s_{\alpha\setminus i}^{x_{\alpha\setminus i}}$$

The fact $\sum_{x_\alpha} \psi_\alpha(x_\alpha) = F_\alpha(1, \ldots, 1)$ follows by marginalizing each variable one at a time. □

Proof of Proposition 4.2.

$$\frac{\partial^a}{\partial s_i^a} F_\alpha(s_\alpha)\Big|_{s_i=0} = \sum_{x_{\alpha\setminus i}} \sum_{x_i} \psi_\alpha(x_{\alpha\setminus i}, x_i) s_{\alpha\setminus i}^{x_{\alpha\setminus i}} \frac{\partial^a}{\partial s_i^a} s_i^{x_i}\Big|_{s_i=0} = a! \sum_{x_{\alpha\setminus i}} \psi_\alpha(x_{\alpha\setminus i}, a) s_{\alpha\setminus i}^{x_{\alpha\setminus i}}$$

The final equality holds because $\frac{\partial^a}{\partial s_i^a} s_i^{x_i}\big|_{s_i=0} = a!$ if $x_i = a$ and zero otherwise. □

Proof of Proposition 4.3. The PGF is

$$\begin{aligned}
F_{\alpha\cup j}(s_\alpha, s_j) &= \sum_{x_\alpha} \sum_{x_j} \psi_\alpha(x_\alpha) \operatorname{Binomial}(x_j \mid x_i, \rho) s_\alpha^{x_\alpha} s_j^{x_j} \\
&= \sum_{x_\alpha} \psi_\alpha(x_\alpha) s_\alpha^{x_\alpha} \sum_{x_j} \operatorname{Binomial}(x_j \mid x_i, \rho) s_j^{x_j} \\
&= \sum_{x_\alpha} \psi_\alpha(x_\alpha) s_\alpha^{x_\alpha} (\rho s_j + 1 - \rho)^{x_i} \\
&= \sum_{x_\alpha} \psi_\alpha(x_\alpha) s_{\alpha\setminus i}^{x_{\alpha\setminus i}} \big(s_i(\rho s_j + 1 - \rho)\big)^{x_i} \\
&= F_\alpha(s_{\alpha\setminus i}, s_i(\rho s_j + 1 - \rho))
\end{aligned}$$

In the third line, we used the fact that the PGF of the Binomial distribution is $\sum_x \operatorname{Binomial}(x|n, \rho) s^x = (\rho s + 1 - \rho)^n$. □

Proof of Proposition 4.4.

$$\begin{aligned}
F_\gamma(s_\alpha, s_\beta, s_k) &= \sum_{x_\alpha, x_\beta, x_k} \psi_\alpha(x_\alpha)\psi_\beta(x_\beta) \mathbb{I}\{x_k = x_i + x_j\} s_\alpha^{x_\alpha} s_\beta^{x_\beta} s_k^{x_k} \\
&= \sum_{x_\alpha, x_\beta} \psi_\alpha(x_\alpha)\psi_\beta(x_\beta) s_\alpha^{x_\alpha} s_\beta^{x_\beta} s_k^{x_i + x_j} \\
&= \sum_{x_\alpha, x_\beta} \psi_\alpha(x_\alpha)\psi_\beta(x_\beta) \cdot s_{\alpha\setminus i}^{x_{\alpha\setminus i}} \cdot (s_k s_i)^{x_i} \cdot s_{\beta\setminus j}^{x_{\beta\setminus j}} \cdot (s_k s_j)^{x_j} \\
&= \Big(\sum_{x_\alpha} \psi_\alpha(x_\alpha) \cdot s_{\alpha\setminus i}^{x_{\alpha\setminus i}} \cdot (s_k s_i)^{x_i}\Big) \cdot \Big(\sum_{x_\beta} \psi_\beta(x_\beta) \cdot s_{\beta\setminus j}^{x_{\beta\setminus j}} \cdot (s_k s_j)^{x_j}\Big) \\
&= F_\alpha(s_{\alpha\setminus i}, s_k s_i) \cdot F_\beta(s_{\beta\setminus j}, s_k s_j)
\end{aligned}$$

□

Proof of Proposition 4.5. We can combine Propositions 4.3 and 4.2 to first expand the factor with a thinned variable $x_j = \rho \circ x_i$ and then observe $x_j = a$. We get

$$\begin{aligned} F'_\alpha(s_\alpha) &= \frac{1}{a!}\frac{\partial^a}{\partial s_j^a} F_\alpha(s_{\alpha\setminus i}, s_i(\rho s_j + 1 - \rho))\Big|_{s_j=0} \\ &= \frac{1}{a!}\left(\frac{\partial^a}{\partial t_i^a} F_\alpha(s_{\alpha\setminus i}, t_i)(s_i\rho)^a\Big|_{t_i = s_i(\rho s_j + 1 - \rho)}\right)\Bigg|_{s_j=0} \\ &= \frac{1}{a!}(s_i\rho)^a \frac{\partial^a}{\partial t_i^a} F_\alpha(s_{\alpha\setminus i}, t_i)\Big|_{t_i = s_i(1-\rho)}. \end{aligned}$$

□

Proof of Proposition 4.6. This is an immediate consequence of Proposition 4.3 and Proposition 4.1 by setting $s_i = 1$ in Proposition 4.3. □

Proof of Proposition 4.7. This is an immediate consequence of Proposition 4.4 and Proposition 4.1 by setting $s_i = 1$ and $s_j = 1$ in Proposition 4.4. □

4.2.2 The PGF-Forward Algorithm for Poisson HMMs

We now use the operations from the previous section to implement the forward algorithm for Poisson HMMs in the domain of PGFs. The forward algorithm is an instance of variable elimination, but in HMMs is more easily described using the following recurrence for the joint probability $p(n_k, y_{1:k})$:

$$\underbrace{p(n_k, y_{1:k})}_{\alpha_k(n_k)} = \sum_{n_{k-1}} \underbrace{p(n_{k-1}, y_{1:k-1})}_{\alpha_{k-1}(n_{k-1})} p(n_k|n_{k-1})p(y_k|n_k)$$

We can compute the "forward messages" $\alpha_k(n_k) := p(n_k, y_{1:k})$ in a sequential forward pass, assuming it is possible to enumerate all possible values of n_k to store the messages and compute the recurrence. In our case, n_k can take on an infinite number of values, so this is not possible.

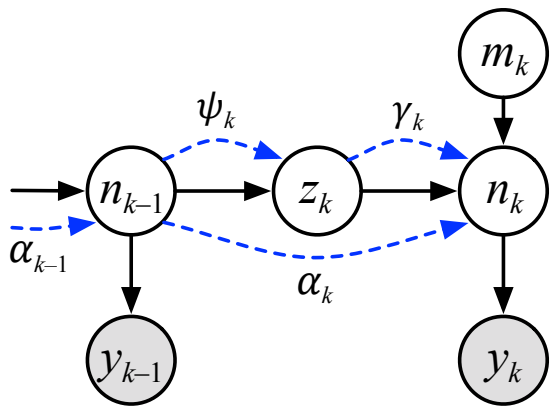

Figure 4.2. Factor graph of the Poisson HMM.

We proceed instead using generating functions. To apply the operations from the previous section, it is useful to instantiate explicit random variables m_k and z_k for the number of new arrivals in step k and survivors from step $k-1$, respectively, to get the model (see Figure 4.2):

$$m_k \sim \text{Poisson}(\lambda_k), \qquad z_k = \delta_{k-1} \circ n_{k-1},$$
$$n_k = m_k + z_k, \qquad y_k = \rho_k \circ n_k.$$

Such that the transition distribution $p(n_k|n_{k-1})$ can now be written:

$$p(n_k|n_{k-1}) = \sum_{m_k=0}^{\infty} \sum_{z_k=0}^{\infty} p(m_k)p(z_k|n_{k-1})p(n_k|z_k, m_k),$$

which can be equivalently written as a discrete convolution over all m_k and z_k st $m_k + z_k = n_k$. Substituting this into the recurrence for $\alpha_k(n_k)$ and reorganizing terms gives the following updated recurrence:

$$\alpha_k(n_k) = p(y_k|n_k) \underbrace{\sum_{m_k=0}^{\infty} \sum_{z_k=0}^{\infty} p(m_k)p(n_k|z_k, m_k) \overbrace{\sum_{n_{k-1}=0}^{\infty} \alpha_{k-1}(n_{k-1})p(z_k|n_{k-1})}^{\psi_k(z_k)}}_{\gamma_k(n_k)} \tag{4.1}$$

Algorithm 1 FORWARD

1: $\psi_1(z_1) := \mathbb{I}\{z_1 = 0\}$
2: **for** $k = 1$ to K **do**
3: $\quad \gamma_k(n_k) := \sum_{z_k, m_k} \psi_k(z_k) p(m_k) \mathbb{I}\{n_k = z_k + m_k\}$
4: $\quad \alpha_k(n_k) := \gamma_k(n_k) p(y_k \mid n_k)$
5: $\quad$ **if** $k < K$ **then**
6: $\quad\quad \psi_{k+1}(z_{k+1}) := \sum_{n_k} \alpha_k(n_k) p(z_{k+1} \mid n_k)$

Algorithm 2 PGF-FORWARD

1: $\Psi_1(s) := 1$
2: **for** $k = 1$ to K **do**
3: $\quad \Gamma_k(s) := \Psi_k(s) \cdot \exp\{\lambda_k(s-1)\}$
4: $\quad A_k(s) := \textsc{Evidence}(\Gamma_k(s), y_k, \rho_k)$
5: $\quad$ **if** $k < K$ **then**
6: $\quad\quad \Psi_{k+1}(s) := A_k\big(\delta_k s + 1 - \delta_k\big)$
7: **function** $\textsc{Evidence}(F(s), y, \rho)$
8: $\quad$ **return** $\frac{1}{y!}(s\rho)^y \cdot F^{(y)}\big(s(1-\rho)\big)$

We have introduced the intermediate factors $\psi_k(z_k)$ and $\gamma_k(n_k)$ to clarify the implementation.

FORWARD (Algorithm 1) is a dynamic programming algorithm based on this recurrence to compute the α_k messages for all k. However, it cannot be implemented due to the infinite sums. PGF-FORWARD (Algorithm 2) instead performs the same operations in the domain of generating functions—Ψ_k, Γ_k, and A_k are the PGFs of ψ_k, γ_k, and α_k, respectively. Each line in PGF-FORWARD implements the operation in the corresponding line of FORWARD using the operations given in Section 4.2.1. In Line 1, $\Psi_1(s) = \sum_{z_1} \psi_1(z_1) s^{z_1} = 1$ is the PGF of ψ_1. Line 3 uses "Add then marginalize" (Proposition 4.7) combined with the fact that the Poisson PGF for m_k is $\exp\{\lambda_k(s-1)\}$. Line 4 uses "Thin then observe" (Proposition 4.5), and Line 6 uses "Thin then marginalize" (Proposition 4.6).

Implementation and Complexity. The PGF-FORWARD algorithm as stated is symbolic. It remains to see how it can be implemented efficiently. For this, we need to respresent and manipulate the PGFs in the algorithm efficiently. We do so based on the following result:

Theorem 4.1. *All PGFs in the* PGF-FORWARD *algorithm have the form* $f(s)\exp\{as + b\}$ *where* f *is a polynomial with degree at most* $Y = \sum_k y_k$.

Proof. We verify the invariant inductively. It is clearly satisfied in Line 1 of PGF-FORWARD ($f(s) = 1, a = b = 0$). We check that it is preserved for each op-

eration within the loop. In Line 3, suppose $\Psi_k(s) = f(s)\exp\{as + b\}$. Then $\Gamma_k(s) = f(s)\exp\{(a + \lambda_k)s + (b - \lambda_k)\}$ has the desired form.

In Line 4, assume that $\Gamma_k(s) = f(s)\exp\{as + b\}$. Then one can verify by taking the y_kth derivative of $\Gamma_k(s)$ that $A_k(s)$ is given by:

$$A_k(s) = (a\rho_k)^{y_k} \cdot \left(s^{y_k} \sum_{\ell=0}^{y_k} \frac{f^{(\ell)}(s(1-\rho_k))}{a^\ell \ell!(y_k - \ell)!} \right) \cdot \exp\{a(1-\rho_k)s + b\}$$

The scalar $(a\rho)^{y_k}$ can be combined with the polynomial coefficients or the scalar $\exp(b)$ in the exponential. The second term is a polynomial of degree $y_k + \deg(f)$. The third term has the form $\exp\{a's+b'\}$. Therefore, in Line 4, $A_k(s)$ has the desired form, and the degree of the polynomial part of the representation increases by y_k.

In Line 6, suppose $A_k(s) = f(s)\exp\{as+b\}$. Then $\Psi_{k+1}(s) = g(s)\exp\big\{a\delta_k s + \big(b + a(1-\delta_k)\big)\big\}$, where $g(s)$ is the composition of f with the affine function $\delta_k s + 1 - \delta_k$, so g is a polynomial of the same degree as f. Therefore, $\Psi_{k+1}(s)$ has the desired form.

We have shown that each PGF retains the desired form, and the degree of the polynomial is initially zero and increases by y_k each time through the loop, so it is always bounded by $Y = \sum_k y_k$. □

The important consequence of Theorem 4.1 is that we can represent and manipulate PGFs in PGF-FORWARD by storing at most Y coefficients for the polynomial f plus the scalars a and b. The detailed algorithm, based on this proof of Theorem 4.3, is given in Algorithm 3.

The complexity of the resulting algorithm is given below:

Theorem 4.2. *The running time of* PGF-FORWARD *for Poisson HMMs is* $\mathcal{O}(KY^2)$.

Proof. We assume a polynomial f is represented as a vector of coefficients $\{f_i\}$ of length $\deg(f) + 1$. ARRIVALS takes constant time. The running time of EVIDENCE is

Algorithm 3 PGF-FORWARD implementation

Require: Vectors λ, δ, ρ, y
Ensure: Likelihood $p(y_{1:K})$

```
1:
2: a ← 0, b ← 0, f(s) ← 1
3: for k = 1 to K do
4:     [a, b] ← arrivals(a, b, λ_k)
5:     [a, f] ← evidence(a, f, y_k, ρ_k)
6:     if k < K then
7:         [a, b, f] ← survivors(a, b, f, δ_k)
8: return f(1) exp{a + b}
9:
10: function ARRIVALS(a, b, λ)
11:     a′ ← a + λ
12:     b′ ← b − λ
13:     return a′, b′
14: function EVIDENCE(a, f, y, ρ)
15:     a′ ← a(1 − ρ)
16:     g ← 0, df ← f
17:     for ℓ = 0 to y do
18:         g ← g + df/(a^ℓ ℓ!(y − ℓ)!)
19:         df ← DERIV(df)
20:     g ← COMPOSE(g, s(1 − ρ))
21:     g ← (aρ)^y s^y g
22:     return a′, g
23:
24: function SURVIVORS(a, b, f, δ)
25:     a′ ← aδ
26:     b′ ← b + a(1 − δ)
27:     f′ ← COMPOSE(f, δs + 1 − δ)
28:     return a′, b′, f′
```

Algorithm 4 TAIL-ELIMINATE

Ensure: Unnormalized marginal $p(n_i, y_{1:K})$

$\phi_{i,i+1}(n_i, z_{i+1}) := \alpha_i(n_i)p(z_{i+1}|n_i)$
for $j = i+1$ to K **do**
  $\eta_{ij}(n_i, n_j) := \sum_{m_j, z_j} \phi(n_i, z_j)p(m_j)p(n_j|z_j, m_j)$
  $\theta_{ij}(n_i, n_j) := \eta_{ij}(n_i, n_j)p(y_j|n_j)$
  if $j < K$ **then**
    $\phi_{i,j+1}(n_i, z_{j+1}) := \theta_{ij}(n_i, n_j)p(z_j|n_{j-1})$
return $p(n_i, y_{1:K}) = \sum_{n_K} \theta_{iK}(n_i, n_K)$

Algorithm 5 PGF-TAIL-ELIMINATE

Ensure: PGF of unnormalized marginal $p(n_i, y_{1:K})$

$\Phi_{i,i+1}(s,t) := A_i(s(\delta_i t + 1 - \delta_i))$
for $j = i+1$ to K **do**
  $H_{ij}(s,t) := \Phi_{ij}(s,t)\exp\{\lambda_k(t-1)\}$
  $\Theta_{ij}(s,t) := \frac{1}{y_j!}(t\rho_j)^{y_j} \frac{\partial^{y_j} H_{ij}(s,u)}{\partial u^{y_j}}\Big|_{u=t(1-\rho_j)}$
  if $j < K$ **then**
    $\Phi_{i,j+1}(s,t) := \Theta_{ij}(s, \delta_j t + 1 - \delta_j)$
return $\Theta_{iK}(s,1)$

$\mathcal{O}(y\deg(f)) = \mathcal{O}(Y^2)$: Lines 19 and 20 are executed y times and take time proportional to $\deg(g)$ and $\deg(df)$, respectively, each of which is no more than $\deg(f)$. The operations outside the loop are bounded by $\mathcal{O}(y + \deg(f))$. (Note that the COMPOSE operation in Line 22 is linear in $\deg(g)$—simply multiply the ith coefficient of g by $(1-\rho)^i$ for all i.) The SURVIVORS function takes $\mathcal{O}(Y^2)$ time. The COMPOSE operation in Line 29 is more costly than the one on Line 22: we must expand $\sum_i g_i(\delta s + 1 - \delta)^i$ to compute the coeffients of s^i for all i—this can be done in $O(\deg(g)^2)$ time by a number of methods, e.g., applying the Binomial Theorem to expand each term. The ARRIVALS, EVIDENCE, and SURVIVORS functions are each called K or $K-1$ times. Therefore, the overall running time is $\mathcal{O}(KY^2)$. □

4.2.3 Computing Marginals by Tail Elimination

PGF-FORWARD allows us to efficiently compute the likelihood in a Poisson HMM. We would also like to compute posterior marginals, the standard approach for which is the forward-backward algorithm Rabiner (1989). A natural question is whether there is an efficient PGF implementation of the backward algorithm for Poisson HMMs. While we were able to derive this algorithm symbolically, the functional form of the PGFs is more complex and we do not know of a polynomial-time implementation.

Instead, we adopt a variable elimination approach that is less efficient in terms of the number of operations performed on factors ($\mathcal{O}(K^2)$ instead of $\mathcal{O}(K)$ to compute all posterior marginals) but with the significant advantage that those operations are efficient. The key principle is to always eliminate predecessors before successors in the Poisson HMM. This allows us to apply operations similar to those in PGF-FORWARD.

Define $\theta_{ij}(n_i, n_j) := p(n_i, n_j, y_{1:j})$ for $j > i$. We can write a recurrence for θ_{ij} similar to Equation (4.1). For $j > i + 1$:

$$\theta_{ij}(n_i, n_j) = p(y_j|n_j) \underbrace{\sum_{m_j, z_j} p(m_j) p(n_j|z_j, m_j) \overbrace{\sum_{n_{j-1}} \theta_{i,j-1}(n_i, n_{j-1}) p(z_j|n_{j-1})}^{\phi_{ij}(n_i, z_j)}}_{\eta_{ij}(n_i, n_j)}.$$

We have again introduced intermediate factors, with probabilistic meanings $\phi_{ij}(n_i, z_j) = p(n_i, z_j, y_{1:j-1})$ and $\eta_{ij}(n_i, n_j) = p(n_i, n_j, y_{1:j-1})$.

PGF-TAIL-ELIMINATE (Algorithm 5) is a PGF-domain dynamic programming algorithm based on this recurrence to compute the PGFs of the θ_{ij} factors for all $j \in \{i + 1, \ldots, K\}$. The non-PGF version of the algorithm is given in Algorithm 4 for comparison. We use Θ_{ij}, Φ_{ij}, and H_{ij} to represent the joint PGFs of θ_{ij}, ϕ_{ij}, and η_{ij}, respectively. The algorithm can also be interpreted as variable elimination using the order $z_{i+1}, n_{i+1}, \ldots, z_K, n_K$, after having already eliminated variables $n_{1:i-1}$ and $z_{1:i-1}$ in the forward algorithm, and therefore starting with the PGF of $\alpha_i(n_i)$. PGF-

TAIL-ELIMINATE concludes by marginalizing n_K from Θ_{iK} to obtain the PGF of the unnormalized posterior marginal $p(n_i, y_{1:K})$. Each line of PGF-TAIL-ELIMINATE uses the same operations given in Section 4.2.1. Line 1 uses "Binomial thinning" (Proposition 4.3), Line 3 uses "Add then marginalize" (Proposition 4.7), Line 4 uses "Thin then observe" (Proposition 4.5) and Line 6 uses "Thin then marginalize" (Proposition 4.6).

Implementation and Complexity. The considerations for implementating PGF-TAIL-ELIMINATE are similar to those of PGF-FORWARD, with the details being slightly more complex due to the larger factors. The detailed implementation is found in Algorithm 6.

Algorithm 6 PGF-TAIL-ELIMINATE implementation

Require: Vectors λ, δ, ρ, y, index i, parameters f, a, b of initial PGF $A_i(s) = f(s)\exp\{as + b\}$ (from PGF-FORWARD)
Ensure: Final PGF for unnormalized marginal $p(n_i, y_{1:K})$ in form $f(s)\exp\{as + b\}$

1: Initialize: $f(s,t)\exp\{ast + bs + ct + d\}$
2: $[a, b, c, d, f] \leftarrow$ INIT-SURVIVORS(a, b, f, δ_i)
3: **for** $j = i + 1$ to K **do**
4: $\quad [c, d] \leftarrow$ ARRIVALS(c, d, λ_k)
5: $\quad [a, c, f] \leftarrow$ EVIDENCE(a, c, f, y_k, ρ_k)
6: $\quad$ **if** $k < K$ **then**
7: $\quad\quad [a, b, c, d, f] \leftarrow$ SURVIVORS$(a, b, c, d, f, \delta_k)$
8: **return** $f(s, 1)\exp\{(a + b)s + (c + d)\}$

9: **function** INIT-SURVIVORS(a, b, f, δ)
10: $\quad a' \leftarrow a\delta$
11: $\quad b' \leftarrow b(1 - \delta)$
12: $\quad c' \leftarrow 0$
13: $\quad d' \leftarrow b$
14: $\quad f'(s,t) \leftarrow \sum_i f_i s^i (\delta t + 1 - \delta)^i$
15: $\quad$ **return** a', b', c', d', f'

16: **function** ARRIVALS(c, d, λ)
17: $\quad c' \leftarrow c + \lambda$
18: $\quad d' \leftarrow d - \lambda$
19: $\quad$ **return** c', d'

20: **function** EVIDENCE(a, c, f, y, ρ)
21: $\quad a' \leftarrow a(1 - \rho)$
22: $\quad c' \leftarrow c(1 - \rho)$
23: $\quad g \leftarrow 0$, $df \leftarrow f$
24: $\quad$ **for** $\ell = 0$ to y **do**
25: $\quad\quad g \leftarrow g + \dfrac{\text{MULT}(df, (as + c)^{y-\ell})}{\ell!(y - \ell)!}$
26: $\quad\quad df \leftarrow$ PARTIAL(df, t)
27: $\quad g \leftarrow$ COMPOSE$(g, t(1 - \rho))$
28: $\quad g \leftarrow \rho^y s^y g$
29: $\quad$ **return** a', g

30: **function** SURVIVORS(a, b, f, δ)
31: $\quad a' \leftarrow a\delta$
32: $\quad b' \leftarrow b + a(1 - \delta)$
33: $\quad c' \leftarrow c\delta$
34: $\quad d' \leftarrow d + c(1 - \delta)$
35: $\quad f' \leftarrow$ COMPOSE$(f, \delta t + 1 - \delta)$
36: $\quad$ **return** a', b', f'

Similar to PGF-FORWARD, the following theorems describe the closed form of the factors in PGF-TAIL-ELIMINATE and bound its runtime.

Theorem 4.3. *All PGFs in the* PGF-TAIL-ELIMINATE *algorithm have the form* $f(s,t) \cdot \exp\{ast + bs + ct + d\}$ *where* f *is a bivariate polynomial with maximum exponent most* $Y = \sum_k y_k$.

Proof. We again proceed inductively. From the proof of Theorem 4.1, we initially have that $A_i(s) = f(s)\exp\{as+b\}$ where $\deg(f) = \sum_{k=1}^{i} y_k$. Then, in Line 1, we have

$$\Psi_{i,i+1}(s,t) = f\big(s(\delta_i t + 1 - \delta_i)\big)\exp\{a\delta_i st + a(1-\delta_i)s + b\}$$

The first term is a bivariate polynomial $f'(s,t) := \sum_{i=0}^{\deg(f)} f_i s^i(\delta_i t + 1 - \delta_i)^i$ with max-degree equal to $\deg(f)$, and the second term has the desired exponential form.

In Line 3, suppose $\Phi_{ij}(s,t) = f(s,t)\exp\{ast + bs + ct + d\}$. Then $H_{ij}(s,t) = f(s,t)\exp\{ast + cs + (c+\lambda_k)t + (d-\lambda_k)\}$, which has the desired form.

In Line 4, the suppose $H_{ij}(s,u) = f(s,u)\exp\{ast + bs + cu + d\}$. One can verify by calculating the yth partial derivative of H_{ij} with respect to u that:

$$\Theta_{ij}(s,t) = \rho_j^{y_j} \cdot \left(t^{y_j} \sum_{\ell=0}^{y_j} \frac{(as+c)^{y_j-\ell}}{\ell!(y_j-\ell)!} \cdot \frac{\partial^\ell}{\partial u^\ell} f(s,u)\Big|_{u=t(1-\rho_j)} \right) \cdot \exp\left\{a(1-\rho_j)st + bs + c(1-\rho)t + d\right\}$$

The term in parentheses is again a bivariate polynomial—the largest exponent of s and t have both increased by y_j, so the max-degree increases by y_j. The exponential term is in the desired form and can absorb the scalar ρ^{y_j}. Therefore, in Line 4, $\Theta_{ij}(s,t)$ has the desired form, and the degree of the polynomial part of the representation increases by y_j.

In Line 6, suppose $\Theta_{ij}(s,t) = f(s,t)\exp\{ast + bs + ct + d\}$. Then $\Phi_{i,j+1}(s,t) = g(s,t)\exp\left\{a\delta_k st + \big(b + a(1-\delta_k)\big)s + c\delta_k t + \big(d + c(1-\delta_k)\big)\right\}$, where $g(s,t) = f(s,h(t))$ is the composition of f with the affine function $h(t) = \delta_k t + 1 - \delta_k$, so g is a bivariate polynomial of the same degree as f. Therefore, $\Phi_{i,j+1}(s,t)$ has the desired form.

We have shown that each PGF retains the desired form. Furthermore, the max-degree of the polynomial is initially equal to $\sum_{k=1}^{i} y_k$ and increases by y_j for all $j = i+1$ to K, so it is always bounded by $Y = \sum_{k=1}^{K} y_k$. □

Theorem 4.4. PGF-TAIL-ELIMINATE *can be implemented to run in time* $\mathcal{O}(Y^3(\log Y + K))$*, and the PGFs for all marginals can be computed in time* $\mathcal{O}(KY^3(\log Y + K))$*.*

Proof. We assume for simplicity that all polynomials have max-degree equal to the upper bound Y. A bivariate polynomial is represented as a matrix of Y^2 coefficients for the monomials $s^i t^j$.

The running time of INIT-SURVIVORS function is dominated by Line 16, which takes $\mathcal{O}(Y^2)$ time. For each term in the sum, the coefficients of the polynomial $(\delta t + 1 - \delta)^i$ can be computed in $\mathcal{O}(i) = \mathcal{O}(Y)$ time (e.g., by the Binomial Theorem) and then multiplied by f_i to determine the coefficients of $s^i t^j$ for all j. This repeats $\mathcal{O}(Y)$ times, once for each term in the sum.

The running time of ARRIVALS is $\mathcal{O}(1)$.

The running time of SURVIVORS is $\mathcal{O}(Y^3)$. The COMPOSE operation in Line 41 can be structured as

$$\sum_{i,j} f_{ij} s^i (\delta t + 1 - \delta)^j = \sum_i s^i \sum_j f_{ij} (\delta t + 1 - \delta)^j$$

For each value of i, we compose the univariate polynomial $\sum_j f_{ij} t^j$ with the affine function $\delta t + 1 - \delta$. This can be done in $O(Y^2)$ time, as in the proof of Theorem 4.2, for a total running time of $O(Y^3)$.

The total running time of PGF-TAIL-ELIMINATE *excluding* the EVIDENCE function is therefore $O(KY^3)$.

The running time of one call to EVIDENCE is $\mathcal{O}(yY^2 \log Y)$. It is dominated by Line 29. The multiplication in this line can be structured as

$$\Big(\sum_{i,j} (df)_{ij} s^i t^j\Big)(as + c)^{y-\ell} = \sum_j t^j \Big(\sum_i (df)_{ij} s^i\Big)(as + c)^{y-\ell}$$

For each value of j, we multiply two univariate polynomials in s whose total degree is at most Y. This can be done in time $\mathcal{O}(Y \log Y)$ using a fast Fourier transform.

We repeat this at most $Y \cdot y$ times—once for each possible value of j and ℓ. The total running time of a single call to EVIDENCE is therefore $\mathcal{O}(yY^2 \log Y)$. The total running time of all calls to the evidence function is $\mathcal{O}(\sum_{j=i+1}^{K} y_k Y^2 \log Y) = \mathcal{O}(Y^3 \log Y)$.

The overall running time is therefore $\mathcal{O}(Y^3(K + \log Y))$. □

4.2.4 Extracting Posterior Marginals and Moments

After computing the PGF of the posterior marginals, we wish to compute the actual probabilities and other quantities, such as the moments, of the posterior distribution. This can be done efficiently:

Theorem 4.5. *The PGF of the unnormalized posterior marginal $p(n_i, y_{1:K})$ has the form $F(s) = f(s)\exp\{as + b\}$ where $f(s) = \sum_{j=0}^{m} c_j s^j$ is a polynomial of degree $m \leq Y$. Given the parameters of the PGF, the posterior mean, the posterior variance, and an arbitrary entry of the posterior probability mass function can each be computed in $\mathcal{O}(m) = \mathcal{O}(Y)$ time as follows, where $Z = f(1)\exp\{a + b\}$:*

1. $\mu := \mathbb{E}[n_i | y_{1:k}] = e^{a+b-\log Z} \sum_{j=0}^{m} (a + m) c_j$
2. $\sigma^2 := \mathrm{Var}(n_i | y_{1:k}) = \mu - \mu^2 + e^{a+b-\log Z} \sum_{j=0}^{m} ((a + m)^2 - m) c_j$
3. $\Pr(n_i = \ell | y_{1:k}) = e^{b-\log Z} \sum_{j=0}^{\min\{m,\ell\}} c_j \frac{a^{\ell-i}}{(\ell-i)!}$

Proof. We assume for the proof that the PGF is already normalized, which can be done by setting $b \leftarrow b - \log Z$. For (i) and (ii), we use the following standard facts about PGFs: $\mu = F^{(1)}(1)$ and $\sigma^2 = F^{(2)}(1) - \mu^2 + \mu$ Casella & Berger (2002). Then we have:

$$
\begin{aligned}
\mu = F^{(1)}(1) &= \frac{d}{ds} f(s) e^{as+b}\Big|_{s=1} \\
&= f'(1)e^{a+b} + af(1)e^{a+b} \\
&= e^{a+b} \sum_{i=0}^{m} (mf_i + af_i) \\
&= e^{a+b} \sum_{i=0}^{m} (a+m) f_i
\end{aligned}
$$

And

$$
\begin{aligned}
F^{(2)}(1) &= \frac{d^2}{ds^2} f(s) e^{as+b}\Big|_{s=1} \\
&= f^{(2)}(1)e^{a+b} + 2f^{(1)}(1)ae^{a+b} + a^2 f(1) \\
&= e^{a+b} \left(f^{(2)}(1) + 2af^{(1)}(1) + a^2 f(1)) \right) \\
&= e^{a+b} \sum_{i=0}^{m} \left(m(m-1)f_i + 2amf_i + a^2 f_i \right) \\
&= e^{a+b} \sum_{i=0}^{m} ((a+m)^2 - m) f_i
\end{aligned}
$$

For part (iii), we use the following standard fact about the Taylor expansion of the exponential:

$$
e^{as} = \sum_{j=0}^{\infty} \frac{a^j}{j!} s^j
$$

Then we have:

$$
\begin{aligned}
F(s) &= \Big(\sum_{i=0}^{m} f_i s^i \Big) e^{as+b} \\
&= e^b \Big(\sum_{i=0}^{m} f_i s^i \Big) \Big(\sum_{j=0}^{\infty} \frac{a^j}{j!} s^j \Big) \\
&= e^b \sum_{i=0}^{m} \sum_{j=0}^{\infty} f_i \frac{a^j}{j!} s^{i+j} \\
&= e^b \sum_{\ell=0}^{\infty} s^\ell \sum_{i=0}^{\min\{m,\ell\}} f_i \frac{a^{\ell-i}}{(\ell-i)!}
\end{aligned}
$$

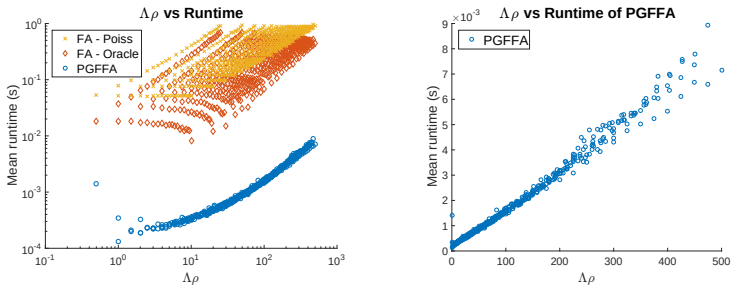

Figure 4.3. Runtime of PGF-FORWARD and truncated algorithm vs. $\Lambda\rho$. Left: log-log scale. Right: PGF-FORWARD only, linear scale.

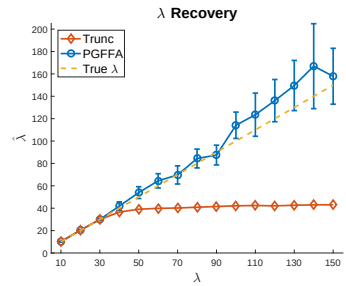

Figure 4.4. Parameter estimation w/ PGF-FORWARD

The final expression reveals the unique explicit representation of the PGF as a formal power series in s. The coefficient of s^ℓ, which is equal to the value of the PMF at ℓ, is $e^b \sum_{i=0}^{\min\{m,\ell\}} f_i \frac{a^{\ell-i}}{(\ell-i)!}$. □

4.3 Experiments

We conducted experiments to demonstrate that our method is faster than standard approximate approaches for computing the likelihood in Poisson HMMs, that it leads to better parameter estimates, and to demonstrate the computation of posterior marginals on an ecological data set.

4.3.1 Running Time

We compared the runtimes of PGF-FORWARD and the truncated forward algorithm, a standard method for Poisson HMMs in the ecology domain (Dail & Madsen, 2011). The runtime of our algorithm depends on the magnitude of the observed counts. The runtime of the truncated forward algorithm is very sensitive to the setting of the trunctation parameter $N_{\max}$: smaller values are faster, but may underestimate the likelihood. Selecting $N_{\max}$ large enough to yield correct likelihoods but small enough to be fast is difficult (Chandler; Couturier et al., 2013; Dennis et al., 2015). We evaluated two strategies to select $N_{\max}$. The first is an oracle strategy, where we first

searched for the smallest value of $N_{\max}$ for which the error in the likelihood is at most 0.001, and then compared vs. the runtime for that value (excluding the search time). The second strategy, adapted from (Dennis et al., 2015), is to set $N_{\max}$ such that the maximum discarded tail probability of the Poisson prior over any n_k is less than 10^{-5}.

To explore these issues we generated data from models with arrival rates $\lambda = \Lambda \times [0.0257, 0.1163, 0.2104, 0.1504, 0.0428]$ and survival rates $\delta = 0.2636 \times [1, 1, 1, 1]$ based on a model for insect populations (Zonneveld, 1991). We varied the overall population size parameter $\Lambda \in \{10, 20, \ldots, 100, 125, 150, \ldots, 500\}$, and detection probability $\rho \in \{0.05, 0.10, \ldots, 1.00\}$. For each parameter setting, we generated 25 data sets and recorded the runtime of both methods.

Figure 4.3 shows that PGF-FORWARD is 2–3 orders of magnitude faster than even the oracle truncated algorithm. The runtime is plotted against $\Lambda\rho \propto \mathbb{E}[Y]$, the primary parameter controlling the runtime of PGF-FORWARD. Empirically, the runtime depends linearly on the magnitude of observed counts instead of what we predicted theoretically (that it scaled quadratically) —this is likely due to the implementation, which is dominated by loops that execute $\mathcal{O}(Y)$ times, with much faster vectorized $\mathcal{O}(Y)$ operations within the loops.

4.3.2 Parameter Estimation

We now examine the impact of exact vs. truncated likelihood computations on parameter estimation in the N-mixture model (Royle, 2004). A well-known feature of this and related models is that it is usually easy to estimate the product of the population size parameter λ and detection probability ρ, which determines the mean of the observed counts, but, without enough data, it is difficult to estimate both parameters accurately, especially as $\rho \to 0$ (e.g., see (Dennis et al., 2015)). It was previously shown that truncating the likelihood can artificially suppress instances where the true maximum-likelihood estimates are infinite (Dennis et al., 2015), a

phenomenon that we also observed. We designed a different, simple, experiment to reveal another failure case of the truncated likelihood, which is avoided by our exact methods. In this case, the modeler is given observed counts over 50 time steps ($K = 50$) at 20 iid locations. She selects a heuristic fixed value of $N_{\max}$ approximately 5 times the average observed count based on her belief that the detection probability is not too small and this will capture most of the probability mass.

To evaluate the accuracy of parameter estimates obtained by numerically maximizing the truncated and exact likelihoods using this heuristic for $N_{\max}$ we generated true data from different values of λ and ρ with $\lambda\rho = \mathbb{E}[y]$ fixed to be equal to 10—the modeler does not know the true parameters, and in each cases chooses $N_{\max} = 5\mathbb{E}[y] = 50$. Figure 4.4 shows the results. As the true λ increases close to and beyond $N_{\max}$, the truncated method cuts off significant portions of the probability mass and severely underestimates λ. Estimation with the exact likelihood is noisier as λ increases and $\rho \to 0$, but not biased by truncation. While this result is not surprising, it reflects a realistic situation faced by the practitioner who must select this trunctation parameter.

4.3.3 Marginals

We demonstrate the computation of posterior marginals and parameter estimation on an end-to-end case study to model the abundance of Northern Dusky Salamanders at 15 sites in the mid-Atlantic US using data from (Zipkin et al., 2014). The data consists of 14 counts at each site, conducted in June and July over 7 years. We first fit a Poisson HMM by numerically maximizing the likelihood as computed by PGF-FORWARD. The model has three parameters total, which are shared across sites and time: arrival rate, survival rate, and detection probability. Arrivals are modeled as a homogenous Poisson process, and survival is modeled by assuming indvidual lifetimes are exponentially distributed. The fitted parameters indicated an arrival rate of 0.32

individuals per month, a mean lifetime of 14.25 months, and detection probability of 0.58.

Figure 4.5 shows the posterior marginals as computed by PGF-TAIL-ELIMINATE with the fitted parameters, which are useful both for model diagnostics and for population status assessments. The crosses show the posterior mean, and color intensity indicates the actual PMF. Overall, computing maximum likelihood estimates required 189 likelihood evaluations and thus $189 \times 15 = 2835$ calls to PGF-FORWARD, which took 24s total. Extracting posterior marginals at each site required 14 executions of the full PGF-TAIL-ELIMINATE routine (at all 14 latent variables), and took 1.6s per site. Extracting the marginal probabilities and posterior mean took 0.0012s per latent variable.

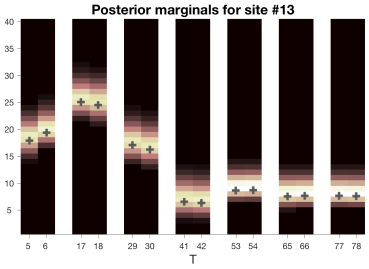

Figure 4.5. Posterior marginals for abundance of Northern Dusky Salamanders at 1 site. See text.

CHAPTER 5

EXACT INFERENCE FOR INTEGER LATENT-VARIABLE MODELS

5.1 Introduction

In Chapter 4, we introduced a new technique for exact inference in models with latent count variables. The approach executes the same operations as variable elimination, but with factors represented in a compact way using probability generating functions (PGFs). In that chapter, we demonstrated a *symbolic* implementation of an inference algorithm using the PGF representation. This symbolic approach is efficient, but depends crucially on properties of the Poisson distribution and the Poisson HMM.

In this chapter, we extend the PGF-based techniques from the preceding chapter to the broader class of latent branching processes (see Section 3.2). To do so, we introduce a new algorithmic technique based on higher-order automatic differentiation (Griewank & Walther, 2008) for inference with PGFs. A key insight is that most inference tasks do not require a full symbolic representation of the PGF. For example, the likelihood is computed by evaluating a PGF $F(s)$ at $s = 1$. Other probability queries can be posed in terms of derivatives $F^{(k)}(s)$ evaluated at either $s = 0$ or $s = 1$. In all cases, it suffices to *evaluate* F and its higher-order derivatives at *particular* values of s, as opposed to computing a compact symbolic representation of F. It may seem that this problem is then solved by standard techniques, such as higher-order forward-mode automatic differentiation (Griewank & Walther, 2008). However, the requisite PGF F is complex—it is defined recursively in terms of higher-order derivatives of other PGFs—and off-the-shelf automatic differentiation methods

do not apply. We therefore develop a novel recursive procedure using building blocks of forward-mode automatic differentiation (*generalized dual numbers* and *univariate Taylor polynomials*; Griewank & Walther, 2008) to evaluate F and its derivatives.

This algorithmic contribution leads to the first efficient exact algorithms for the family of latent branching processes (LBPs). Additional discussion of the LBP family and its relationship with other models is available in Chapter 3. Our algorithms permit exact calculation of the likelihood for all of these models even when they are partially observed.

We demonstrate experimentally that our new exact inference algorithms are more scalable than competing approximate approaches, and support learning via exact likelihood calculations in a broad class of models for which this was not previously possible.

5.1.1 Problem Statement

In this chapter, we seek to solve the following problems for the broad class of LBPs:

- Compute the likelihood $\mathcal{L}(\theta) = p(y_{1:K}; \theta)$ for any θ,
- Compute moments and values of the pmf of the *filtered marginals* $p(n_k \mid y_{1:k}; \theta)$, for any k, θ,
- Estimate parameters θ by maximizing the likelihood.

We focus technically on the first two problems, which will enable numerical optimization to maximize the likelihood. Another standard problem is to compute *smoothed marginals* $p(n_k \mid y_{1:K}; \theta)$ given both past and future observations relative to time step k. Although this is interesting, it is technically more difficult, and we defer it for future work.

5.2 Methods

The standard approach for inference in HMMs is the forward-backward algorithm (Rabiner, 1989), which is a special case of more general propagation or message-passing algorithms (Pearl, 1986; Lauritzen & Spiegelhalter, 1988; Jensen et al., 1990; Shenoy & Shafer, 1990). In Chapter 4, we showed how to implement the forward algorithm using PGFs for models with Bernoulli offspring and Poisson immigration. We will now extend that result to more general latent count models.

5.2.1 Forward Algorithm with PGFs

In Chapter 4 we observed that, for some conditional distributions $p(n_k \mid n_{k-1})$ and $p(y_k \mid n_k)$, the operations of the forward algorithm can be carried out using PGFs. In this section, we develop a more general presentation of the recurrences for the PGF-FORWARD algorithm that apply to the broader family of LBP models and reiterate the result of Theorem 4.5, which is critical for the autodiff approach we develop in Section 5.2.2.

In the PGF-FORWARD algorithm, the PGF-based equivalent to the traditional forward algorithm, we define the PGFs $\Gamma_k(u_k)$ and $A_k(s_k)$ of $\gamma_k(n_k)$ and $\alpha_k(n_k)$, respectively, as:

$$\Gamma_k(u_k) := \sum_{n_k=0}^{\infty} \gamma_k(n_k) u_k^{n_k}, \tag{5.1}$$

$$A_k(s_k) := \sum_{n_k=0}^{\infty} \alpha_k(n_k) s_k^{n_k}. \tag{5.2}$$

The PGFs Γ_k and A_k are power series in the variables u_k and s_k with coefficients equal to the message entries. Technically, Γ_k and A_k are *unnormalized* PGFs because the coefficients do not sum to one. However, the normalization constants are easily recovered by evaluating the PGF on input value 1: for example, $A_k(1) = \sum_{n_k} \alpha_k(n_k) = p(y_{1:k})$. This also shows that we can recover the likelihood as $A_K(1) = p(y_{1:K})$. After nor-

malizing, the PGFs can be interpreted as expectations, for example $A_k(s_k)/A_k(1) = \mathbb{E}[s_k^{N_k} \mid y_{1:k}]$.

In general, it is well known that the PGF $F(s)$ of a non-negative integer-valued random variable X uniquely defines the entries of the probability mass function and the moments of X, which are recovered from (higher-order) derivatives of F evaluated at zero and one, respectively:

$$\Pr(X = r) = F^{(r)}(0)/r!, \tag{5.3}$$

$$\mathbb{E}[X] = F^{(1)}(1), \tag{5.4}$$

$$\mathrm{Var}(X) = F^{(2)}(1) - \left[F^{(1)}(1)\right]^2 + F^{(1)}(1). \tag{5.5}$$

More generally, the first q moments are determined by the derivatives $F^{(r)}(1)$ for $r \leq q$. Therefore, if we can evaluate the PGF A_k and its derivatives for $s_k \in \{0, 1\}$, we can answer arbitrary queries about the filtering distributions $p(n_k, y_{1:k})$, and, in particular, solve our three stated inference problems.

But how can we compute values of A_k, Γ_k, and their derivatives? What form do these PGFs have? One key result from Chapter 4, which we generalize here, is the fact that there is also a recurrence relation among the PGFs.

Proposition 5.1. *Consider the probability model defined in Equations* (3.1) *and* (3.2)*. Let F_k be the PGF of the offspring random variable X_k, and let G_k be the PGF of the immigration random variable M_k. Then Γ_k and A_k satisfy the following recurrence:*

$$\Gamma_k(u_k) = A_{k-1}\big(F_k(u_k)\big) \cdot G_k(u_k) \tag{5.6}$$

$$A_k(s_k) = \frac{(s_k \rho_k)^{y_k}}{y_k!} \cdot \Gamma_k^{(y_k)}\big(s_k(1 - \rho_k)\big) \tag{5.7}$$

Proof. A slightly less general version of Equation (5.6) appeared in Chapter 4; the general version appears in the literature on branching processes with immigration (Heath-

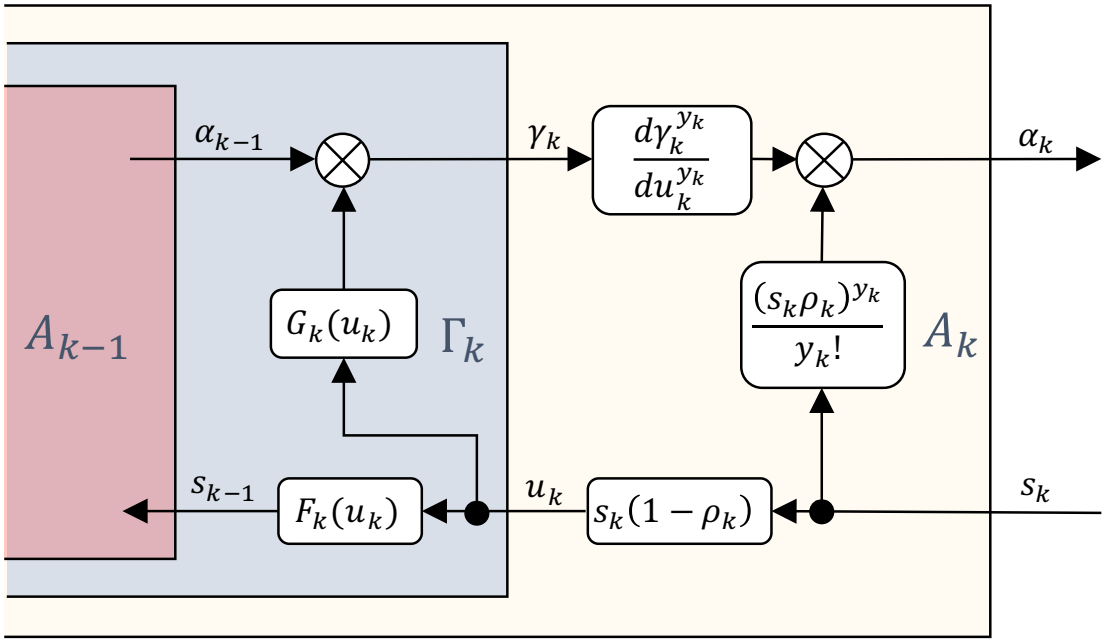

Figure 5.1. Circuit diagram of $A_k(s_k)$

cote, 1965). Equation (5.7) follows directly from general PGF operations outlined in Chapter 4. □

The PGF recurrence has the same two elements as the pmf recurrence in equations (2.5) and (2.6). Equation (5.6) is the prediction step: it describes the PGF of $\gamma_k(n_k) = p(n_k, y_{1:k-1})$ in terms of previous PGFs. Equation (5.7) is the evidence step: it describes the PGF for $\alpha_k(n_k) = p(n_k, y_{1:k})$ in terms of the previous PGF and the new observation y_k. Note that the evidence step involves the y_kth derivative of the PGF Γ_k from the prediction step, where y_k is the observed count. These high-order derivatives complicate the calculation of the PGFs.

5.2.2 Evaluating A_k via Automatic Differentiation

The recurrence reveals structure about A_k and Γ_k but does not immediately imply an algorithm. In Chapter 4, we showed how to use the recurrence to compute *symbolic*

representations of all PGFs in the special case of Bernoulli offspring and Poisson immigration: in this case, they proved that all PGFs have the form $F(s) = f(s)\exp(as+b)$, where f is a polynomial of bounded degree. Hence, they can be represented compactly and computed efficiently using the recurrence. The result is a *symbolic* representation, so, for example, one obtains a closed form representation of the final PGF A_K, from which the likelihood, entries of the pmf, and moments can be calculated. However, the compact functional form $f(s)\exp(as + b)$ seems to rely crucially on properties of the Poisson distribution. When other distributions are used, the size of the symbolic PGF representation grows quickly with K. It is an open question whether the symbolic methods can be extended to other classes of PGFs.

This motivates an alternate approach. Instead of computing A_k symbolically, we will *evaluate* A_k and its derivatives at particular values of s_k corresponding to the queries we wish to make (cf. Equations (5.3)–(5.5)). To develop the approach, it is helpful to consider the *feed-forward* computation for evaluating A_k at a particular value s_k. The circuit diagram in Figure 5.1 is a directed acyclic graph that describes this calculation; the nodes are intermediate quantities in the calculation, and the shaded rectangles illustrate the recursively nested PGFs.

Now, we can consider techniques from automatic differentiation (autodiff) to compute A_k and its derivatives. However, these will not apply directly. Note that A_k is defined in terms of higher-order derivatives of the function Γ_k, which depends on higher-order derivatives of Γ_{k-1}, and so forth. Standard autodiff techniques cannot handle these recursively nested derivatives. Therefore, we will develop a novel algorithm.

5.2.2.1 Computation Model and Dual Numbers

We now develop basic notation and building blocks that we will assemble to construct our algorithm. It is helpful to abstract from our particular setting and

describe a general model for derivatives within a feed-forward computation, following (Griewank & Walther, 2008). We consider a procedure that assigns values to a sequence of variables $v_0, v_1, \ldots, v_n$, where v_0 is the input variable, v_n is the output variable, and each intermediate variable v_j is computed via a function $\varphi_j(v_i)_{i \prec j}$ of some subset $(v_i)_{i \prec j}$ of the variables $v_{0:j-1}$. Here the *dependence* relation $i \prec j$ simply means that φ_j depends directly on v_i, and $(v_i)_{i \prec j}$ is the vector of variables for which that is true. Note that the dependence relation defines a directed acyclic graph G (e.g., the circuit in Figure 5.1), and $v_0, \ldots, v_n$ is a topological ordering of G.

We will be concerned with the values of a variable v_ℓ and its derivatives with respect to some earlier variable v_i. To represent this cleanly, we first introduce a notation to capture the partial computation between the assignment of v_i and v_ℓ. For $i \leq \ell$, define $f_{i\ell}(v_{0:i})$ to be the value that is assigned to v_ℓ if the values of the first i variables are given by $v_{0:i}$ (now treated as fixed input values). This can be defined formally in an inductive fashion:

$$f_{i\ell}(v_{0:i}) = \varphi_\ell(u_{ij})_{j \prec \ell}, \quad u_{ij} = \begin{cases} v_j & \text{if } j \leq i \\ f_{ij}(v_{0:i}) & \text{if } j > i \end{cases}$$

This can be interpreted as recursion with memoization for $v_{0:i}$. When φ_ℓ "requests" the value of u_{ij} of v_j: if $j \leq i$, this value was given as an input argument of $f_{i\ell}$, so we just "look it up"; but if $j > i$, we recursively compute the correct value via the partial computation from i to j. Now, we define a notation to capture derivatives of a variable v_ℓ with respect to an earlier variable v_i.

Definition 5.1 (Dual numbers)**.** *The generalized dual number $\langle v_\ell, dv_i \rangle_q$ for $0 \leq i \leq \ell$ and $q > 0$ is the sequence consisting of v_ℓ and its first q derivatives with respect to v_i:*

$$\langle v_\ell, dv_i \rangle_q = \left(\frac{\partial^p}{\partial v_i^p} f_{i\ell}(v_{0:i}) \right)_{p=0}^{q}$$

We say that $\langle v_\ell, dv_i \rangle_q$ *is a* dual number of order q with respect to v_i. *Let* $\mathbb{DR}_q$ *be the set of dual numbers of order* q. *We will commonly write dual numbers as:*

$$\langle s, du \rangle_q = \left(s, \frac{ds}{du}, \ldots, \frac{d^q s}{du^q} \right)$$

in which case it is understood that $s = v_\ell$ *and* $u = v_i$ *for some* $0 \leq i \leq \ell$, *and the function* $f_{i\ell}(\cdot)$ *will be clear from context.*

Our treatment of dual numbers and partial computations is more explicit than what is standard. In particular, we are explicit both about the variable v_ℓ we are differentiating and the variable v_i with respect to which we are differentiating. This is important for our algorithm, and also helps distinguish our approach from traditional automatic differentiation approaches. Forward-mode autodiff computes derivatives of all variables with respect to v_0, i.e., it computes $\langle v_j, dv_0 \rangle_q$ for $j = 1, \ldots, n$. Reverse-mode autodiff computes derivatives of v_n with respect to all variables, i.e., it computes $\langle v_n, dv_i \rangle_q$ for $i = n-1, \ldots, 0$. In each case, one of the two variables is fixed, so the notation can be simplified.

5.2.2.2 Operations on Dual Numbers

The general idea of our algorithm will resemble forward-mode autodiff. Instead of sequentially calculating the values $v_1, \ldots, v_n$ in our feed-forward computation, we will calculate dual numbers $\langle v_1, dv_{i_1} \rangle_{q_1}, \ldots, \langle v_n, dv_{i_n} \rangle_{q_n}$, where we leave unspecified (for now) the variables with respect to which we differentiate, and the order of the dual numbers. We will require three high-level operations on dual numbers. The first one is "lifting" a scalar function.

Definition 5.2 (Lifted Function)**.** *Let* $f : \mathbb{R}^m \to \mathbb{R}$ *be a function of variables* $x_1, \ldots, x_m$. *The* q*th-order* lifted function $\mathcal{L}^q f : (\mathbb{DR}_q)^m \to \mathbb{DR}_q$ *is the function that accepts as input dual numbers* $\langle x_1, du \rangle_q, \ldots, \langle x_m, du \rangle_q$ *of order* q *with respect to the same variable* u, *and returns the value* $\left\langle f(x_1, \ldots, x_m), du \right\rangle_q$.

Lifting is the basic operation of higher-order forward mode autodiff. For functions f consisting only of "primitive operations", the lifted function $\mathcal{L}^q f$ can be computed at a modest overhead relative to computing f.

Proposition 5.2 ((Griewank & Walther, 2008))**.** *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a function that consists only of the following primitive operations, where x and y are arbitrary input variables and all other numbers are constants: $x + cy$, $x * y$, x/y, x^r, $\ln(x)$, $\exp(x)$, $\sin(x)$, $\cos(x)$. Then $\mathcal{L}^q f$ can be computed in time $\mathcal{O}(q^2)$ times the running time of f.*

Based on this proposition, we will write algebraic operations on dual numbers, e.g., $\langle x, du\rangle_q \times \langle y, du\rangle_q$, and understand these to be lifted versions of the corresponding scalar operations. The standard lifting approach is to represent dual numbers as *univariate Taylor polynomials* (UTPs), in which case many operations (e.g., multiplication, addition) translate directly to the corresponding operations on polynomials. We will use UTPs in the proof of Theorem 5.1.

The second operation we will require is composition. Say that variable v_j *separates* v_i from v_ℓ if all paths from v_i to v_ℓ in G go through v_j.

Theorem 5.1 (Composition)**.** *Suppose v_j separates v_i from v_ℓ. In this case, the dual number $\langle v_\ell, dv_i\rangle_q$ depends only on the dual numbers $\langle v_\ell, dv_j\rangle_q$ and $\langle v_j, dv_i\rangle_q$, and we define the* composition operation*:*

$$\langle v_\ell, dv_j\rangle_q \circ \langle v_j, dv_i\rangle_q := \langle v_\ell, dv_i\rangle_q$$

If v_j does not separate v_i from v_ℓ, the written composition operation is undefined. The composition operation can be performed in $\mathcal{O}(q^2 \log q)$ time by composing two UTPs.

Proof. If all paths from v_i to v_ℓ go through v_j, then v_j is a "bottleneck" in the partial computation f_{il}. Specifically, there exist functions F and H such that $v_j = F(v_i)$

and $v_\ell = H(v_j)$. Here, the notation suppresses dependence on variables that either are not reachable from v_i, or do not have a path to v_ℓ, and hence may be treated as constants because they they do not impact the dual number $\langle v_\ell, v_i \rangle_q$. Now, our goal is to compute the higher-order derivatives of $v_\ell = H(F(v_i))$. Let $\hat{F}$ and $\hat{H}$ be infinite Taylor expansions about v_i and v_j, respectively, *omitting the constant terms* $F(v_i)$ and $H(v_j)$:

$$\hat{F}(\varepsilon) := \sum_{p=1}^{\infty} \frac{F^{(p)}(v_i)}{p!}\varepsilon^p, \quad \hat{H}(\varepsilon) := \sum_{p=1}^{\infty} \frac{H^{(p)}(v_j)}{p!}\varepsilon^p.$$

These are polynomials in ε, and the first q coefficients are given in the input dual numbers. The coefficient of ε^p in $\hat{U}(\varepsilon) := \hat{H}(\hat{F}(\varepsilon))$ for $p \geq 1$ is exactly $d^p v_\ell / dv_i^p$ (see Wheeler, 1987, where the composition of Taylor polynomials is related directly to the higher-order chain rule known as Faà dí Bruno's Formula). So it suffices to compute the first q coefficients of $\hat{H}(\hat{F}(\epsilon))$. This can be done by executing Horner's method (Horner, 1819) in *truncated Taylor polynomial* arithmetic (Griewank & Walther, 2008), which keeps only the first q coefficients of all polynomials (i.e., it assumes $\varepsilon^p = 0$ for $p > q$). After truncation, Horner's method involves q additions and q multiplications of polynomials of degree at most q. Polynomial multiplication takes time $\mathcal{O}(q \log q)$ using the FFT, so the overall running time is $\mathcal{O}(q^2 \log q)$. $\square$

We have assumed that all paths from v_i to v_ℓ go through v_j, and we wish to show that there exist functions $\tilde{F}$ and $\tilde{H}$ such that

$$v_j = F(v_i) := \tilde{F}(v_i, v_A)$$
$$v_\ell = H(v_j) := \tilde{H}(v_j, v_B)$$

and all nodes in v_A and v_B are either not reachable from v_i or have no path to v_ℓ. Note that if a variable v_k is not reachable from v_i, then the scalar value v_ℓ may still depend on v_k, but the derivatives $\frac{d^q v_\ell}{dv_i^q}$ do not depend on v_k, so it is safe to treat v_k as

a fixed constant relative to the dual number $\langle v_\ell, dv_i \rangle_q$. If v_k has no path to v_ℓ, neither v_ℓ nor the derivatives $\frac{d^q v_\ell}{dv_i^q}$ depend on v_k.

The construction of $\tilde{F}$ is easy:

$$\tilde{F}(v_i, v_A) := f_{ij}(v_i, v_{0:i-1})$$

The nodes $v_{0:i-1}$ precede v_i in the topological ordering, and hence have no path from i.

To construct $\tilde{H}$, we reason about the partial computation $f_{j\ell}(v_{0:\ell})$ from v_j to v_ℓ. Recall that this is defined recursively starting with $\varphi_\ell(v_k)_{k \prec \ell}$, and terminating whenever a variable v_p is reached for $p \leq j$. Consider any such variable v_p that is reached by the calculation. Then v_p must satisfy $p \prec q$ for $q > j$ (otherwise the recursion would not reach v_p), and, furthermore, there must be a path from v_q to v_ℓ (otherwise the recursion does not reach v_q). In other words, there is a path $v_p, v_q, \ldots, v_\ell$ for $q > j$. However, this implies that v_p is *not* reachable from v_i, otherwise we would contradict the assumption that all paths from v_p to v_ℓ go through v_j. Therefore, if we consider the subset of variables $v_B \subseteq v_{0:j-1}$ on which $f_{j\ell}$ depends, none of these variables is reachable from v_i. Therefore we can write:

$$\tilde{H}(v_j, v_B) = f_{j\ell}(v_j, v_B)$$

where we omit from $f_{j\ell}$ the arguments on which it does not depend.

The final operation we will require is differentiation. This will support local functions φ_ℓ that differentiate a previous value, e.g., $v_\ell = \varphi_\ell(v_j) = d^p v_j / dv_i^p$.

Definition 5.3 (Differential Operator)**.** *Let $\langle s, du \rangle_q$ be a dual number. For $p \leq q$, the differential operator D^p applied to $\langle s, du \rangle_q$ returns the dual number of order $q - p$ given by:*

$$D^p \langle s, du \rangle_q := \left(\frac{d^p s}{du^p}, \ldots, \frac{d^q s}{du^q} \right)$$

Algorithm 7 $A_k(s_k)$

if $k = 0$ **then**
  [1:] **return** $\alpha_k = 1$
[2:] $u_k = s_k(1 - \rho_k)$
[3:] $s_{k-1} = F_k(u_k)$
[4:] $\gamma_k = A_{k-1}(s_{k-1}) \cdot G_k(u_k)$
[5:] $\alpha_k = \frac{d^{y_k}}{du_k^{y_k}} \gamma_k \cdot (s_k \rho_k)^{y_k} / y_k!$
[6:] **return** α_k

Algorithm 8 $\mathcal{L}A_k(\langle s_k, ds_k \rangle_q)$ — GDUAL-FORWARD

if $k = 0$ **then**
  [1:] **return** $\langle \alpha_k, ds_k \rangle_q = (1, 0, \ldots, 0)$
[2:] $\langle u_k, ds_k \rangle_q = \langle s_k, ds_k \rangle_q \cdot (1 - \rho_k)$
[3:] $\langle s_{k-1}, du_k \rangle_{q+y_k} = \mathcal{L}F_k\big(\langle u_k, du_k \rangle_{q+y_k}\big)$
[4:] $\langle \gamma_k, du_k \rangle_{q+y_k} = \big[\mathcal{L}A_{k-1}\big(\langle s_{k-1}, ds_{k-1} \rangle_{q+y_k}\big) \circ \langle s_{k-1}, du_k \rangle_{q+y_k}\big] \times \mathcal{L}G_k\big(\langle u_k, du_k \rangle_{q+y_k}\big)$
[5:] $\langle \alpha_k, ds_k \rangle_q = \big[D^{y_k} \langle \gamma_k, du_k \rangle_{q+y_k} \circ \langle u_k, ds_k \rangle_q\big] \times \big(\rho_k \langle s_k, ds_k \rangle_q\big)^{y_k} / y_k!$
[6:] **return** $\langle \alpha_k, ds_k \rangle_q$

The differential operator can be applied in $\mathcal{O}(q)$ time.

This operation was defined in (Kalaba & Tesfatsion, 1986).

5.2.2.3 The GDUAL-FORWARD Algorithm

We will now use these operations to lift the function A_k to compute $\langle \alpha_k, s_k \rangle_q = \mathcal{L}A\big(\langle s_k, ds_k \rangle_q\big)$, i.e., the output of A_k and its derivatives with respect to its input. Algorithm 7 gives a sequence of mathematical operations to compute $A_k(s_k)$. Algorithm 8 shows the corresponding operations on dual numbers; we call this algorithm the *generalized dual-number forward algorithm* or GDUAL-FORWARD. Note that a dual number of a variable with respect to itself is simply $\langle x, dx \rangle_q = (x, 1, 0, \ldots, 0)$; such expressions are used without explicit initialization in Algorithm 8. Also, if the dual number $\langle x, dy \rangle_q$ has been assigned, we will assume the scalar value x is also available, for example, to initialize a new dual variable $\langle x, dx \rangle_q$ (cf. the dual number on the RHS of Line 3). Note that Algorithm 7 contains a non-primitive operation on Line 5: the derivative $d^{y_k}\gamma_k / du_k^{y_k}$. To evaluate this in Algorithm 8, we must manipulate the dual number of γ_k to be taken with respect to u_k, and not the original input value s_k, as in forward-mode autodiff. Our approach can be viewed as following a different recursive principle from either forward or reverse-mode autodiff: in the circuit dia-

gram of Figure 5.1, we calculate derivatives of each nested circuit with respect to its own input, starting with the innermost circuit and working out.

Theorem 5.2. *$\mathcal{L}A_K$ computes $\langle \alpha_k, ds_k \rangle_q$ in time $\mathcal{O}\big(K(q+Y)^2 \log(q+Y)\big)$ where $Y = \sum_{k=1}^{K} y_k$ is the sum of the observed counts and q is the requested number of derivatives. Therefore, the likelihood can be computed in $\mathcal{O}(KY^2 \log Y)$ time, and the first q moments or the first q entries of the filtered marginals can be computed in time $\mathcal{O}\big(K(q+Y)^2 \log(q+Y)\big)$.*

Proof. To see that GDUAL-FORWARD is correct, note that it corresponds to Algorithm 7, but applies the three operations from the previous section to operate on dual numbers instead of scalars. We will verify that the conditions for applying each operation are met. Lines 2–5 each use lifting of algebraic operations or the functions F_k and G_k, which are assumed to consist only of primitive operations. Lines 4 and 5 apply the composition operation; here, we can verify from Figure 5.1 that s_{k-1} separates u_k and α_{k-1} (Line 4) and that u_k separates s_k and γ_k (Line 5). The conditions for applying the differential operator on Line 5 are also met.

For the running time, note that the total number of operations on dual numbers in $\mathcal{L}A_K$, including recursive calls, is $\mathcal{O}(K)$. The order of the dual numbers is initially q, but increases by y_k in each recursive call (Line 4). Therefore, the maximum value is $q + Y$. Each of the operations on dual numbers is $\mathcal{O}(p^2 \log p)$ for dual numbers of order p, so the total is $\mathcal{O}(K(q+Y)^2 \log(q+Y))$. □

5.3 Experiments

In this section we describe simulation experiments to evaluate the running time of GDUAL-FORWARD against other algorithms, and to assess the ability to learn a wide variety of models for which exact likelihood calculations were not previously possible, by using GDUAL-FORWARD within a parameter estimation routine.

5.3.1 Running Time vs Y

We compared the running time of GDUAL-FORWARD with the symbolic implementation of the PGF-FORWARD algorithm from the previous chapter as well as TRUNC, the standard truncated forward algorithm (Dail & Madsen, 2011). PGF-FORWARD is only applicable to the Poisson HMM from Chapter 4, which, in the terminology of LBPs, is a model with a Poisson immigration distribution and a Bernoulli offspring distribution. TRUNC applies to any choice of distributions, but is approximate. For these experiments, we restrict to Poisson HMMs for the sake of comparison with the less general symbolic PGF-FORWARD algorithm.

A primary factor affecting running time is the magnitude of the counts. We measured the running time for all algorithms to compute the likelihood $p(y;\theta)$ for vectors $y := y_{1:K} = c \times (1,1,1,1,1)$ with increasing c. In this case, $Y = \sum_k y_k = 5c$. PGF-FORWARD and GDUAL-FORWARD have running times $\mathcal{O}(KY^2)$ and $\mathcal{O}(KY^2 \log Y)$, respectively, which depend only on Y and not θ. The running time of an FFT-based implementation of TRUNC is $O(KN_{\max}^2 \log N_{\max})$, where $N_{\max}$ is the value used to truncate the support of each latent variable. A heuristic is required to choose $N_{\max}$ so that it captures most of the probability mass of $p(y;\theta)$ but is not too big. The appropriate value depends strongly on θ, which in practice may be unknown. In preliminary experiments with realistic immigration and offspring models (see below) and *known* parameters, we found that an excellent heuristic is $N_{\max} = 0.4Y/\rho$, which we use here. With this heuristic, TRUNC's running time is $\mathcal{O}(\frac{K}{\rho^2} Y^2 \log Y)$.

Figure 5.3 shows the results for $\rho \in \{0.15, 0.85\}$, averaged over 20 trials with error bars showing 95% confidence intervals of the mean. GDUAL-FORWARD and TRUNC have the same asymptotic dependence on Y but GDUAL-FORWARD scales better empirically, and is exact. It is about 8x faster than TRUNC for the largest Y when $\rho = 0.15$, and 2x faster for $\rho = 0.85$. PGF-FORWARD is faster by a factor of $\log Y$ in theory and scales better in practice, but applies to fewer models.

5.3.2 Running Time for Different θ

We also conducted experiments where we varied parameters and used an *oracle* method to select $N_{\max}$ for TRUNC. This was done by running the algorithm for increasing values of $N_{\max}$ and selecting the smallest one such that the likelihood was within 10^{-6} of the true value (see Winner & Sheldon, 2016).

We simulated data from Poisson HMMs and measured the time to compute the likelihood $p(y;\theta)$ for the *true* parameters $\theta = (\lambda, \delta, \rho)$, where λ is a vector whose kth entry is the mean of the Poisson immigration distribution at time k, and δ and ρ are scalars representing the Bernoulli survival probability and detection probability, respectively, which are shared across time steps. We set λ and δ to mimic three different biological models; for each, we varied ρ from 0.05 to 0.95. The biological models were as follows: 'PHMM' follows a temporal model for insect populations (Zonneveld, 1991) with $\lambda = (5.13, 23.26, 42.08, 30.09, 8.56)$ and $\delta = 0.26$; 'PHMM-peaked' is similar, but sets $\lambda = (0.04, 10.26, 74.93, 25.13, 4.14)$ so the immigration is temporally "peaked" at the middle time step; 'NMix' sets $\lambda = (80, 0, 0, 0, 0)$ and $\delta = 0.4$, which is similar to the N-mixture model (Royle, 2004), with no immigration following the first time step.

Figure 5.2 shows the running time of all three methods versus ρ. In these models, $\mathbb{E}[Y]$ is proportional to ρ, and the running times of GDUAL-FORWARD and PGF-FORWARD increase with ρ due to the corresponding increase in Y. PGF-FORWARD is faster by a factor of $\log Y$, but is applicable to fewer models. GDUAL-FORWARD perfoms best relative to PGF-FORWARD for the NMix model, because it is fastest when counts occur in early time steps.

Recall that the running time of TRUNC is $O(N_{\max}^2 \log N_{\max})$. For these models, the distribution of the *hidden* population depends only on λ and δ, and these are the primary factors determining $N_{\max}$. Running time decreases slightly as ρ increases,

because the observation model $p(y \mid n; \rho)$ exerts more influence restricting implausible settings of n when the detection probability is higher.

5.3.3 Parameter Estimation

To demonstrate the flexibility of the method, we used GDUAL-FORWARD within an optimization routine to compute maximum likelihood estimates (MLEs) for models with different immigration and growth distributions. In each experiment, we generated 10 independent observation vectors for $K = 7$ time steps from the same model $p(y; \theta)$, and then used the `L-BFGS-B` algorithm to numerically find θ to maximize the log-likelihood of the 10 replicates. We varied the distributional forms of the immigration and offspring distributions as well as the mean $R := \mathbb{E}[X_k]$ of the offspring distribution. We fixed the mean immigration $\lambda := \mathbb{E}[M_k] = 6$ and the detection probability to $\rho = 0.6$ across all time steps. The quantity R is the "basic reproduction number", or the average number of offspring produced by a single individual, and is of paramount importance for disease and population models. [1] We varied R, which was also shared across time steps, between 0.2 and 1.2. The parameters λ and R were learned, and ρ was fixed to resolve ambiguity between population size and detection probability. Each experiment was repeated 50 times; a very small number of optimizer runs failed to converge after 10 random restarts and were excluded.

Figure 5.4 shows the distribution of 50 MLE estimates for R vs. the true values for each model. In all cases the distribution of the estimate is centered around the true parameter. It is evident that GDUAL-FORWARD can be used effectively to produce parameter estimates across a variety of models for which exact likelihood computations were not previously possible.

[1] It is well known in branching processes that if $R > 1$, the population will tend to "explode" (grow to ∞), and if $R < 1$, the population will die out with probability 1.

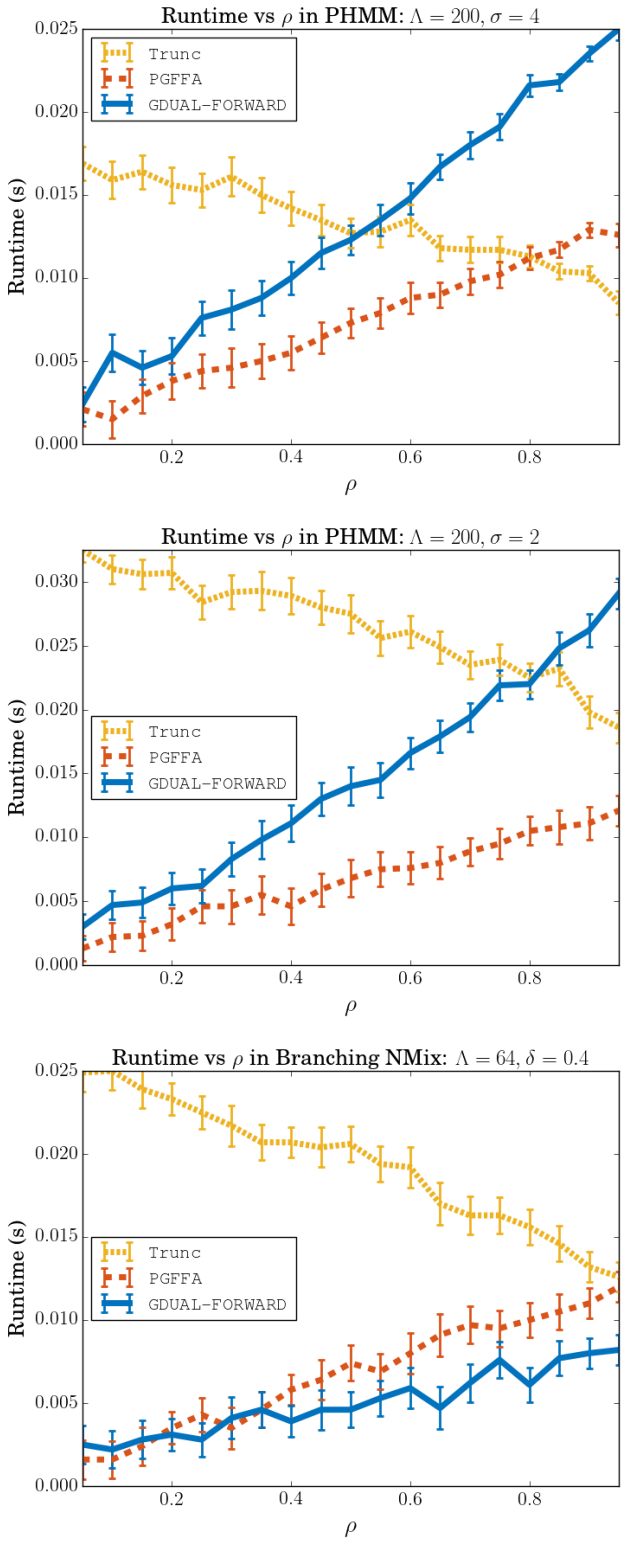

Figure 5.2. Runtime of GDUAL-FORWARD vs baselines. Top: PHMM. Center: PHMM-peaked. Bottom: NMix. See text for descriptions.

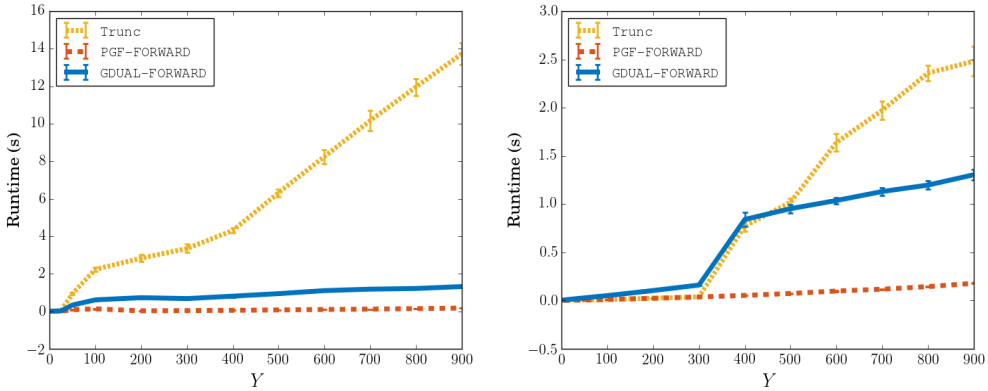

Figure 5.3. Running time vs. Y. Top: $\rho = 0.15$, bottom: $\rho = 0.85$.

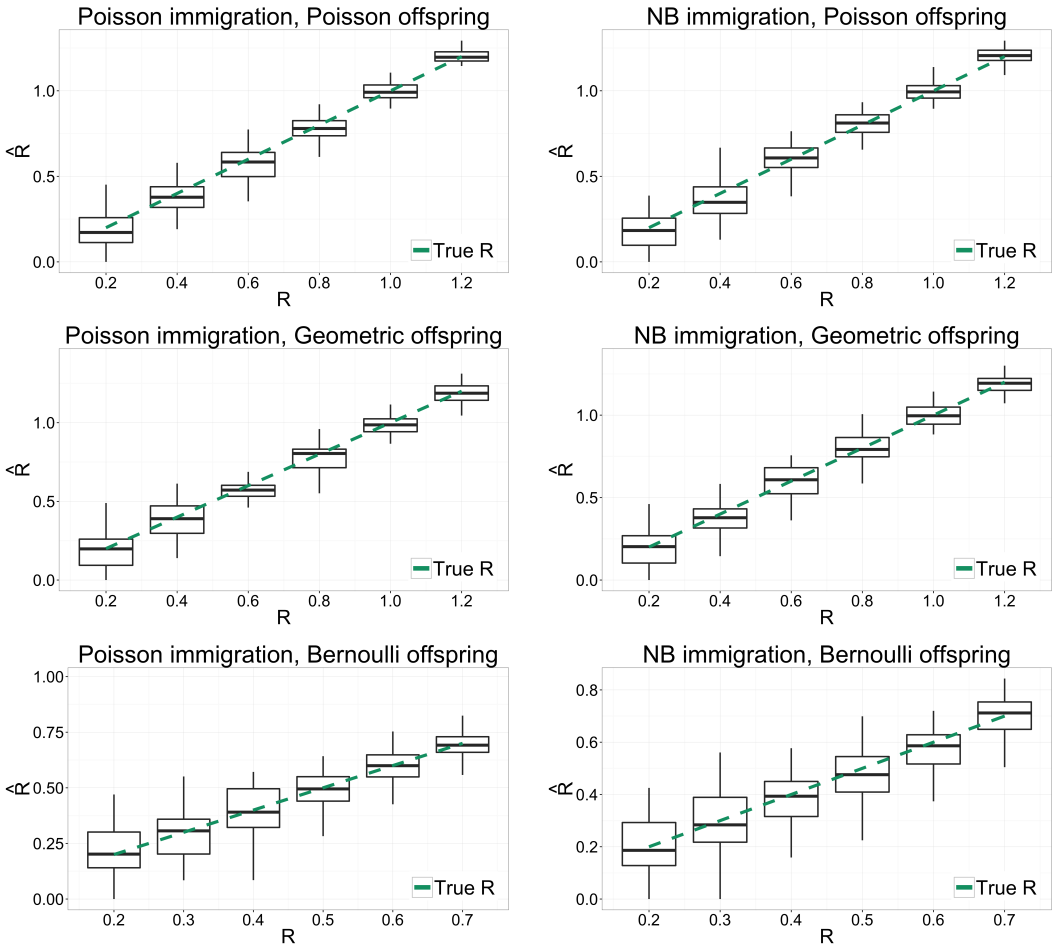

Figure 5.4. Estimates of R in different models. Titles indicate immigration and offspring distribution. 50 trials summarized as box plot for each model, parameter combination.

CHAPTER 6

APPROXIMATE INFERENCE FOR INTEGER LATENT-VARIABLE MODELS

6.1 Introduction

As we've seen in the preceding chapters, performing inference in models with count-valued latent variables is a difficult task. When faced with count data, a researcher generally has four options:

1. ignore the latent dynamics and perform regression on the counts (Link & Sauer, 1994; Ralph et al., 1995; Link & Sauer, 1997; Schmidt & Pellet, 2009),

2. truncate the support of the count distributions to a finite range (Royle, 2004; Dail & Madsen, 2011),

3. adopt a sampling strategy (Kéry et al., 2009; Hostetler & Chandler, 2015; Winner et al., 2015),

4. or, as we developed in preceding chapters, employ PGF inference (Winner & Sheldon, 2016; Winner et al., 2017; Sheldon et al., 2018).

As seen in previous chapters, our PGF-based techniques for exact inference scale significantly better than existing approximate inference approaches. However, the runtime of our exact inference methods as well as the approximate inference methods scales with the magnitude of the observed counts and the size of the population, either through a dependence on the magnitude of the latent abundance (the truncated method (Royle, 2004; Dail & Madsen, 2011; Dennis et al., 2015; Winner &

Sheldon, 2016; Winner et al., 2017)), through the magnitude of the observed counts (PGF-based methods (Winner et al., 2017; Sheldon et al., 2018)), or through the time to convergence (sampling (Winner et al., 2015)). As we increase the temporal and geographical scale/resolution at which we choose to study a population, the surveyed population size will also increase, eventually leading to a point where our data becomes too large for us to be apply any dynamics model.

In this chapter, we present a novel approximate algorithm based on the PGF inference techniques of the previous chapters that *completely removes the runtime scaling of PGF inference with respect to the magnitude of the data/the size of the population*. This approximate algorithm closely matches the PGF-FORWARD algorithm of Chapter 4 and still applies the *correct* models of population dynamics.

The approximation is introduced by assuming that some of the intermediate factors in the PGF-FORWARD algorithm belong to a simple parametric family of distributions and repeatedly projecting them back to this family as they deviate according to the general framework of the assumed density filtering (ADF) technique.

In Section 6.2.1, we present our novel parametric distribution of choice and a moment-matching routine called APGF to project a PGF into this family. Then in Sections 6.2.2 and 6.2.3 we show how to utilize APGF to perform ADF in the PGF-FORWARD algorithm. Finally, in Section 6.3 we evaluate the approximation quality and runtime of our algorithms against the most widely used baseline, TRUNC.

6.1.1 Relation to assumed density filtering

The approximation algorithm we develop in Section 6.2 is based on assumed density filtering (ADF) (Maybeck, 1982; Lauritzen, 1992; Boyen & Koller, 1998), an approximate inference scheme wherein a distribution of interest is factored into a sequence of intermediate components that are iteratively approximated by projecting back to a simple approximate family (see Section 2.2.5 for more details on ADF).

There are, however, several subtle differences between our algorithm and traditional ADF. Primarily, ADF typically uses an approximating family in the exponential family so that the minimization of KLD can be done via moment matching, i.e. selecting the member of the exponential family whose expected sufficient statistics match those of the distribution to be approximated.

The approximating family we propose in Section 6.3 is likely not in the exponential family, but we use a moment-matching scheme regardless. Our scheme matches the first two moments of n_k, which would correspond to an exponential family with n_k and n_k^2 as sufficient statistics. This is primarily due to features of working in the PGF domain: it is not known how to efficiently minimize (or even compute) the KLD between two arbitrary count distributions given only their PGFs, but we can compute a finite number of moments from a PGF quite easily (see Equation (6.1)). This allows us to construct a black-box projection method in the PGF domain based on traditional moment-matching, even though we can't guarantee that the projection is truly minimizing the KLD.

Assumed density filtering also has a well-known extension to the "Expectation Propagation" (EP) algorithm which we do not address in this chapter. As in previous chapters, the main bottleneck in extending this technique to EP has to do with not knowing how to efficiently multiply two arbitrary factors together in the PGF domain, which can be seen when trying to adapt the backward algorithm to the PGF domain and was the primary motivator behind the development of PGF-TAIL-ELIMINATE in Chapter 4. It is possible that the approximations we develop here may be able to mitigate this, but we leave this very interesting problem for future work.

6.2 Methods

Performing approximate inference with ADF requires three key components: a "simple" family of distributions, a procedure to project a distribution into the simple

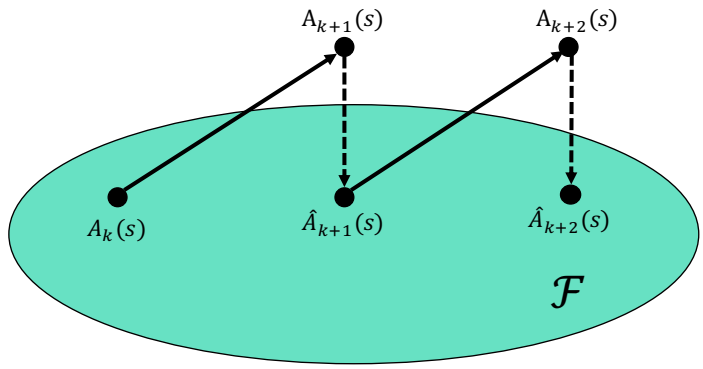

Figure 6.1. Assumed density filtering for PGF-FORWARD. $\mathcal{F}$ is a family of approximate distributions. After each iteration of PGF-FORWARD, the intermediate PGF A_k is projected to a corresponding approximate PGF $\hat{A}_k$ via APGF.

family (which in our case needs to be done in PGF space) and an inference algorithm that repeatedly applies this approximation (Minka, 2001). See Figure 6.2 for an overview of the complete process.

In Section 6.2.1 we define a novel count distribution parameterized by its mean and variance which unifies the binomial, Poisson, and negative binomial distributions. This distribution is the only discrete distribution we are aware of that can be under-, equi-, or over-dispersed and which is amenable to differentiation via autodiff (which is required for performing PGF inference). In Section 6.2.1 we also show how to project the PGF of a distribution into this family by using the corresponding moment generating function (MGF) in an algorithm called APGF.

In Section 6.2.2 we apply APGF to PGF-FORWARD, the base PGF inference algorithm for LBPs from Section 4.2.2. We call the resulting approximate inference algorithm APGF-FORWARD. One convenient side effect of using APGF to approximate the PGFs in PGF-FORWARD is that the challenging high-order derivatives required can now be expressed in closed form without the need to resort to autodiff. Although computing the derivatives this way is a small change and does not mathematically alter the computation, it does result in a fundamental improvement to the scalability and so in Section 6.2.3 we define this modified algorithm as APGF-FORWARD-S.

6.2.1 PGF Approximation – APGF

It is well known that for ADF, if our approximating family of distributions are in the exponential family, then we can project back to the approximating family (by minimizing KLD) via moment matching. In our case, the approximating family we define in Equation (6.3) is not obviously in the exponential family, and in general, it is not known how to evaluate the KL divergence between the PGFs of two arbitrary count distributions. We can, however, use a PGF to easily compute the moments of the corresponding distribution (as we show below). Given the first k moments of this distribution, we can then construct a distribution from our approximating family that matches those moments of the original distribution. While this approach is very similar to traditional moment matching, where we would match the expected sufficient statistics, this is indeed an important distinction between our method and traditional ADF, and so future work may seek to establish that our new moment matching scheme is guaranteed to also be minimizing the KLD. In the rest of this section, we will use "moment-matching" to refer to our new technique of moment matching in the PGF domain and not to the traditional sense of moment matching via sufficient statistics.

Let $F_X(s)$ be a (normalized or unnormalized) PGF for a random variable X that we wish to approximate by some computationally simpler PGF $\hat{F}_X(s)$. In Theorem 4.5 and Equations (5.4) and (5.5), we showed how to use a PGF to compute the mean and variance of a distribution. We can generalize this result to compute the kth moment of a RV X from its PGF $F_X(s)$ by constructing the MGF of X as follows:

$$\begin{aligned} M_X(t) &= F_X(e^t), \\ m_k &= E[X^k] = M^{(k)}(0), \\ &= \frac{d^k}{dt^k} F_X(e^t)\bigg|_{t=0}, \end{aligned}$$

so long as $F_X(s)$ is k-times differentiable at 1. As in Chapter 5, we can use dual numbers to efficiently compute the first k moments of X:

$$[m_0, \ldots, m_k] = F_X\left(\exp\left\{\langle t = 0, dt\rangle_k\right\}\right), \tag{6.1}$$

where $\langle t = 0, dt\rangle_k$ is a dual number for t and the k derivatives of t wrt itself at 0. In Taylor polynomial form, $\langle t = 0, dt\rangle_k$ is simply $[0, 1, 0, 0, \ldots]$.

Given the first k moments from $F_X(s)$, we can then construct the PGF $\hat{F}_X(s)$ of a parametric distribution with the same moments. For our purposes, the PGF $\hat{F}_X(s)$ should exist in closed form, be computationally simple, and the high-order derivatives of $\hat{F}_X(s)$ should all exist.

If we only wish for $\hat{F}_X$ to match the first moment of F_X, then we can use a Poisson distribution with mean m_1 for $\hat{F}_X$:

$$\hat{F}_X(s) := e^{m_1(s-1)}, \tag{6.2}$$

where $m_1 = E[X]$. If, however, we wish for $\hat{F}_X$ to match the first two moments of F_X, there are several choices for two-parameter discrete distributions, including the binomial distribution, the negative binomial distribution, the Conway-Maxwell-Poisson (CMP) distribution (Conway & Maxwell, 1962), the generalized Poisson distribution (Consul & Jain, 1973), and many others. Unfortunately, these distributions all have one of two problems: either they cannot handle both overdispersion and underdispersion (binomial and negative binomial), or the PGF (and/or its derivatives) cannot be evaluated efficiently (CMP and generalized Poisson).

As a result, we propose for $\hat{F}_X$ the following casewise discrete distribution parameterized by its mean μ and variance σ^2:

$$\hat{F}_X(s;\mu,\sigma^2) := \begin{cases} [(1-p)+ps]^n, & n = \left\lfloor \frac{\mu^2}{\mu-\sigma^2} \right\rceil, p = \frac{\mu}{n} & \text{if } \sigma^2 \leq \mu \\ e^{\lambda(s-1)}, & \lambda = \mu & \text{if } \sigma^2 = \mu \\ \left(\frac{q}{1-(1-q)s}\right)^r, & r = \frac{\mu^2}{\sigma^2-\mu}, q = \frac{\mu}{\sigma^2} & \text{if } \sigma^2 \geq \mu \end{cases} \tag{6.3}$$

where $\hat{F}_X$ has the PGF of a binomial, Poisson, or negative binomial whenever F_X is underdispersed, equidispersed, or overdispersed, respectively. In all cases, $\hat{F}_X$ is parameterized such that the mean and variance of $\hat{F}_X$ match those of F_X as closely as possible. Indeed if $\sigma^2 \geq \mu$, the mean and variance of $\hat{F}_X$ and F_X will match exactly.

To see why, we consider first the Poisson case: $\hat{F}_X(s;\mu,\sigma^2) = e^{\lambda(s-1)}$. The mean of X is: $E_{\hat{F}}[X] = \lambda$ and the mean is equal to the variance. This is the simplest case, and we simply set $\lambda = \mu$.

In the binomial case, $\hat{F}_X(s;\mu,\sigma^2) = [(1-p)+ps]^n$, the mean and variance of X are:

$$\mu = np,$$
$$\sigma^2 = np(1-p),$$

with the additional restriction that n must be a positive integer. Ignoring the restriction on n for the moment, we can see that p must equal μ/n. Substituting this value for p into the equation for $\mathrm{Var}_{\hat{F}}[X]$ and solving for n gives:

$$
\begin{aligned}
\sigma^2 &= np(1-p) \\
&= n\frac{\mu}{n}\left(1-\frac{\mu}{n}\right) \\
&= \mu\left(1-\frac{\mu}{n}\right) \\
\frac{\mu}{n} &= 1-\frac{\sigma^2}{\mu} \\
n &= \frac{\mu}{1-\frac{\sigma^2}{\mu}} \\
&= \frac{\mu^2}{\mu-\sigma^2}
\end{aligned}
$$

Then, to force n to be integral, we round to the nearest integer:

$$
\begin{aligned}
n &= \left\lfloor \frac{\mu^2}{\mu-\sigma^2} \right\rceil \\
p &= \frac{\mu}{n}
\end{aligned}
$$

Note that if $\sigma^2 < \mu$, it is possible that $\frac{\mu^2}{\mu-\sigma^2}$ will not be integral and will need to be rounded in order to yield a valid binomial distribution. In this case, the mean of $\hat{F}_X$ will still match F_X and $\hat{F}_X$ will still be underdispersed with variance $\hat{\sigma}^2$ bounded by:

$$
\sigma^2 \leq \hat{\sigma}^2 \leq \mu. \tag{6.4}
$$

Finally, for the negative binomial case, $\hat{F}_X(s;\mu,\sigma^2) = \left(\frac{q}{1-(1-q)s}\right)^r$, the mean and variance of X are:

$$
\begin{aligned}
\mu &= \frac{(1-q)r}{q}, \\
\sigma^2 &= \frac{(1-q)r}{q^2}.
\end{aligned}
$$

Solving the first equation for r gives:

$$
r = \frac{q\mu}{1-q}.
$$

Substituting this value of r into the equation for the variance and solving for q gives:

$$\begin{aligned}
\sigma^2 &= \frac{(1-q)\frac{q\mu}{1-q}}{q^2} \\
&= \frac{q\mu}{q^2} \\
&= \frac{\mu}{q} \\
q &= \frac{\mu}{\sigma^2}
\end{aligned}$$

Substituting this value for q back into the equation for r above and simplifying gives the following for r:

$$\begin{aligned}
r &= \frac{\frac{\mu}{\sigma^2}\mu}{1-\frac{\mu}{\sigma^2}} \\
&= \frac{\mu^2}{\sigma^2\left(1-\frac{\mu}{\sigma^2}\right)} \\
&= \frac{\mu^2}{\sigma^2-\mu}
\end{aligned}$$

While the casewise presentation of the distribution suggests it is discontinuous with respect to μ and σ^2, the three cases are closely related as described in the following theorem:

Theorem 6.1. *When parameterized by their mean μ and variance σ^2, both the binomial and negative binomial distributions have the following PGF:*

$$F(s;\mu,\sigma^2) = \left[\frac{\sigma^2}{\mu} + (1-\frac{\sigma^2}{\mu})s\right]^{\frac{\mu^2}{\mu-\sigma^2}}.$$

So long as one of the following sets of conditions holds:

a) $F(s;\mu,\sigma^2)$ is overdispersed ($\sigma^2 > \mu$),

b) or $F(s;\mu,\sigma^2)$ is underdispersed ($\sigma^2 < \mu$) and $\mu^2 = c(\mu - \sigma^2)$ for some $c \in \mathcal{Z}^+$. Furthermore, as $\sigma^2 \to \mu$, both distributions converge in distribution to the Poisson.

Proof. With exception to the restriction in the Binomial PGF that the number of trials, n, be integral, the Binomial and Negative Binomial PGFs are identical when the distributions are parameterized by their mean and variance. To see why, let $\hat{F}_X(s;\hat{n},\hat{p})$ be the PGF of the binomial distribution:

$$\hat{F}_X(s;\hat{n},\hat{p}) = [(1-\hat{p}) + \hat{p}s]^{\hat{n}}$$

and let $\hat{G}_X(s;\hat{r},\hat{q})$ be the PGF of the negative binomial distribution:

$$\hat{G}_X(s;\hat{r},\hat{q}) = \left[\frac{\hat{q}}{1-(1-\hat{q}s)}\right]^{\hat{r}}$$

If, instead of parameterizing the distributions by their natural parameters, we instead parameterize both distributions by their mean μ and variance σ^2 according to the following parameter relations:

$$\begin{aligned}
\hat{n} &= \frac{\mu^2}{\mu-\sigma^2}, \\
\hat{p} &= 1 - \frac{\sigma^2}{\mu}, \\
\hat{r} &= \frac{\mu^2}{\sigma^2-\mu}, \\
\hat{q} &= \frac{\mu}{\sigma^2},
\end{aligned}$$

ignoring for now the restriction that $\hat{n}$ be integral, then the PGFs $\hat{F}_X$ and $\hat{G}_X$ above become:

$$\begin{aligned}
\hat{F}_X(s;\mu,\sigma^2) &= \left[\frac{\sigma^2}{\mu} + \left(1 - \frac{\sigma^2}{\mu}\right)s\right]^{\frac{\mu^2}{\mu-\sigma^2}}, \\
\hat{G}_X(s;\mu,\sigma^2) &= \left[\frac{\mu/\sigma^2}{1-(1-\mu/\sigma^2)s}\right]^{\frac{\mu^2}{\sigma^2-\mu}}
\end{aligned}$$

when parameterized by their mean and variance. If we manipulate $\hat{G}_X$ algebraically as follows:

$$\begin{aligned}
\hat{G}_X(s;\mu,\sigma^2) &= \left[\frac{\mu/\sigma^2}{1-(1-\mu/\sigma^2)s}\right]^{\frac{\mu^2}{\sigma^2-\mu}} \\
&= \left[\frac{1-(1-\mu/\sigma^2)s}{\mu/\sigma^2}\right]^{-1\frac{\mu^2}{\sigma^2-\mu}} \\
&= \left[\frac{\sigma^2}{\mu} - \left(\frac{\sigma^2}{\mu}-1\right)s\right]^{\frac{\mu^2}{\mu-\sigma^2}} \\
&= \left[\frac{\sigma^2}{\mu} + \left(1-\frac{\sigma^2}{\mu}\right)s\right]^{\frac{\mu^2}{\mu-\sigma^2}} \\
&= \hat{F}_X(s;\mu,\sigma^2)
\end{aligned}$$

which is the PGF of $\hat{F}_X$ likewise parameterized by its mean and variance.

Note that in this parameterization, if $\sigma^2 < \mu$ and $\hat{n}$ is not integral, then $\hat{F}_X$ does not define a valid probability distribution as the corresponding pmf $f_X(x)$ (which has infinite support, unlike the Binomial) will necessarily have negative entries (for some $x > \hat{n}$). Interestingly, $\hat{F}_X(1) = \sum_{x\in\mathcal{X}} f_X(x) = 1$, so in some applications, this invalid distribution may still be useful. In our work, however, $\hat{n}$ is always rounded.

To see the convergence of the binomial and negative binomial cases to the Poisson case, consider what happens as $\sigma^2 \to \mu$. For a binomial distribution with mean μ and variance σ^2, the natural parameterization in terms of the number of trials n and the probability of success p is:

$$\begin{aligned}
n &= \frac{\mu}{1-(\sigma^2/\mu)}, \\
p &= 1-\frac{\sigma^2}{\mu}.
\end{aligned}$$

As $\sigma^2 \to \mu^-$, the limits of n and p are:

$$\lim_{\sigma^2 \to \mu^-} \frac{\mu}{1 - (\sigma^2/\mu)} = \infty,$$
$$\lim_{\sigma^2 \to \mu^-} 1 - \frac{\sigma^2}{\mu} = 0.$$

The limiting distribution of a binomial distribution with mean μ as $n \to \infty$ and $p \to 0$ is a Poisson distribution with mean $np = \mu$.

For a negative binomial distribution with mean μ and variance σ^2, the natural parameterization in terms of the number of trials r and the probability of failure p is:

$$r = \frac{\mu^2}{\sigma^2 - \mu},$$
$$p = \frac{\mu}{\sigma^2}.$$

As $\sigma^2 \to \mu^+$, the limits of r and p are:

$$\lim_{\sigma^2 \to \mu^+} \frac{\mu^2}{\sigma^2 - \mu} = \infty,$$
$$\lim_{\sigma^2 \to \mu^+} \frac{\mu}{\sigma^2} = 1.$$

The limiting distribution of a negative binomial distribution with mean μ as $r \to \infty$ and $p \to 1$ is a Poisson distribution with mean μ. □

Theorem 6.1 is important for likelihood optimization as it implies that the approximating distribution changes smoothly as the variance crosses the mean. This observation is supported empirically by our experiments in Section 6.3.

Pseudocode for the APGF algorithm is presented in Algorithm 9. The computational bottleneck is in Lines 2 and 3 where the first and second derivatives of the moment generating function are computed. Using the techniques of Chapter 5, this can be performed in a black-box manner using automatic differentiation. Unlike the

Algorithm 9 APGF(F_X)

1: $M_X(t) = F_X(e^t)/F_X(1)$
2: $\mu = M'_X(0)$
3: $\sigma^2 = M''_X(0) - \mu^2$
4: **if** $\sigma^2 < \mu$ **then**
5: $\quad n = \text{round}\left(\frac{\mu^2}{\mu - \sigma^2}\right)$
6: $\quad p = \mu/n$
7: $\quad \hat{F}_X(s; n, p) = (1 - p + ps)^n$
8: **else if** $\sigma^2 == \mu$ **then**
9: $\quad \lambda = \mu$
10: $\quad \hat{F}_X(s; \lambda) = e^{\lambda(s-1)}$
11: **else if** $\sigma^2 > \mu$ **then**
12: $\quad r = \frac{\mu^2}{\sigma^2 - \mu}$
13: $\quad q = \mu/\sigma^2$
14: $\quad \hat{F}_X(s; r, q) = \left(\frac{q}{1-(1-q)s}\right)^r$
15: $\hat{F}_X = \hat{F}_X/F_X(1)$
16: **return** $\hat{F}_X$

high-order derivatives needed in previous algorithms (GDUAL-FORWARD), APGF computes only two derivatives of M_X and can be computed in time proportional to a constant times the time needed to evaluate $F_X(s)$ once.

Also note that the pseudocode in APGF does not assume that F_X is a *normalized* PGF. If F_X is not normalized, then $Z = F_X(1) \neq 1$ is used to renormalize the MGF in line 1 and then in the penultimate line, $\hat{F}_X = \hat{F}_X/F_X(1)$, $\hat{F}_X$ is "renormalized" to match F_X. In this way, APGF may be applied to the PGFs of both normalized and unnormalized distributions.

6.2.1.1 Adding a minimum support constraint to APGF

When using the underdispersed case of the APGF distribution, the resultant distribution (which is binomial) will have finite support with upper limit n. While we found that n was generally likely to be large enough in practice, we did include the option to specify a minimum value for n in our implementation for the sake of robustness:

$$n^* = \max\left(n_{\min}, \left\lfloor \frac{\mu^2}{\mu - \sigma^2} \right\rfloor\right),$$
$$p = \frac{\mu}{n^*}.$$

Note that, like when n is rounded, if this override is triggered, $\hat{F}_X$ will still be underdispersed and will still have the correct mean.

6.2.2 The APGF-FORWARD Algorithm

In Chapter 5, we saw that the PGF recurrence that defined the PGF-FORWARD algorithm led to a deeply nested sequence of high-order derivatives. By using APGF, we can use an assumed density filtering (ADF) (Minka, 2001) approach to repeatedly project the intermediate messages back to the parameteric distribution defined in Equation 6.3.

The primary recurrence for the APGF-FORWARD algorithm is presented in Algorithm 10. Note that although we could validly apply APGF to the A_k PGFs instead of the Γ_k messages (by exchanging the order of Lines 3 and 4 and changing Line 3 to operate on A_k), we will see in Section 6.2.3 that applying APGF immediately before applying the differential operator in Line 4 leads to a significant performance improvement that would not be possible with a different approximation schedule.

The exact effect of applying APGF to the Γ_k PGF in each pass through the PGF-FORWARD recurrence is somewhat subtle. Γ_k is an arbitrary PGF defined recursively in terms of the preceding A_{k-1} PGF. Evaluating APGF(Γ_k) will require computing the first two derivatives of Γ_k (and consequently of A_{k-1} and any more deeply nested PGFs). As we saw in Chapter 5, derivatives of A_{k-1} still require the expensive evaluation of a derivative of order y_{k-1}.

However, the $\hat{\Gamma}_k$ PGF returned by APGF is a simple parametric distribution and will no longer be nested: it no longer contains any reference to Γ_k or A_{k-1}. As a result, in the next loop through the APGF-FORWARD recurrence, the evaluation of

Algorithm 10 APGF-FORWARD

1: **if** $k = 0$ **then**
2: [1:] **return** 1
3: [2:] $\Gamma_k(u_k) = A_{k-1}(F_k(u_k)) \cdot G_k(u_k)$
4: [3:] $\hat{\Gamma}_k = \text{APGF}(\Gamma_k)$
5: [4:] $A_k(s_k) = (s_k\rho_k)^{y_k}/y_k! \cdot \frac{d^{y_k}}{du_k^{y_k}}\hat{\Gamma}_k(u_k)\Big|_{u_k=s_k(1-\rho_k)}$
6: [5:] **return** $A_k(s_k)$

APGF(Γ_{k+1}) will "bottom out" at $\hat{\Gamma}_k$. This collapses the chain of recursion and limits the degree to which high-order derivatives become nested, leading to a significant improvement in runtime complexity:

Theorem 6.2. *The runtime of the* APGF-FORWARD *algorithm is* $\mathcal{O}(K\bar{y}^2 \log \bar{y})$ *where* $\bar{y} = \max_i y_i$ *is the maximum observed count and* K *is the number of observations.*

Proof. The computational bottleneck in each iteration of the APGF-FORWARD recurrence is in line 4 where a derivative of order y_k is taken:

$$\frac{d^{y_k}}{du_k^{y_k}}\hat{\Gamma}_k(u_k)\Big|_{u_k=s_k(1-\rho_k)}.$$

As discussed in the proof of Theorem 5.2, this derivative can be computed in time $\mathcal{O}(y_k^2 \log y_k)$ times the time to evaluate $\hat{\Gamma}_k$, which is constant. As discussed in Section 6.2.1, the call to APGF in line 3 takes constant time with respect to the time to evaluate Γ_k. Therefore, the total complexity of each iteration of the recurrence is $\mathcal{O}(y_k^2 \log y_k)$ times a constant.

If $\bar{y} = \max_k y_k$, then the worst case complexity for each iteration is $\mathcal{O}(\bar{y}^2 \log \bar{y})$ and thus the total worst case complexity to perform all K iterations of the recurrence is $\mathcal{O}(K\bar{y}^2 \log \bar{y})$. □

6.2.3 APGF-FORWARD-S — Improving the APGF-FORWARD Algorithm

In APGF-FORWARD and the PGF inference algorithms in Chapters 4 and 5, the most significant computation challenge has always resulted from the high-order differentiation necessary for conditioning on new evidence. In APGF-FORWARD this happens in line 4 where $\hat{\Gamma}_k$ is differentiated y_k times.

In general, the original PGF Γ_k is an arbitrary PGF and cannot be differentiated efficiently without automatic differentiation. APGF-FORWARD manages to break the nesting of high-order derivatives, but it still requires this application of high-order autodiff and thus the runtime still depends on the magnitude of the counts. In this section, we introduce a significant improvement to APGF-FORWARD that *completely removes the scaling with respect to y* by computing the high-order derivatives in closed form. Mathematically, this improved algorithm is identical to APGF-FORWARD, but the runtime scales only with the number of observations K.

To begin, note that the approximate PGF $\hat{\Gamma}_k$ is known in closed form (see Equation 6.3). However, this particular form of APGF was chosen so that the y-th derivatives can also be written in closed form as:

Lemma 6.1.

$$\frac{d^y}{du^y}\hat{\Gamma}(u;\mu,\sigma^2) := \begin{cases} p^y \frac{n!}{(n-y)!}(1-p+pu)^{n-y}, \textit{for } y \leq n & \textit{if } \sigma^2 \leq \mu \\ \lambda^y e^{\lambda(u-1)}, & \textit{if } \sigma^2 = \mu \\ q^r(1-q)^y \frac{(r+y-1)!}{(r-1)!}\left(1-u(1-q)\right)^{-r-y}, & \textit{if } \sigma^2 \geq \mu \end{cases} \tag{6.5}$$

where n, p, λ, r, and q have the same definition as in Equation 6.3. The y-th derivative of $\hat{\Gamma}$ in all cases consists only of primitive operations and can be computed in constant time.

Proof. Note that in APGF-FORWARD, $\hat{\Gamma}$ is one of the three cases of the APGF PGF given in Equation 6.3. Each of the three cases of $\hat{F}_X$ in Equation 6.3 can be written in

the form $g(f(s) \cdot h)$ where $f(s)$ is a function of s whose derivatives wrt s are constant, i.e.:

$$\frac{d^y}{ds^y} f(s) = c, \forall y \geq 1.$$

Critically, this means we can derive the high-order derivatives of $\hat{\Gamma}_k$ by repeatedly applying the chain rule and without invoking the more general Faà di Bruno's formula.

In the derivations that follow, we use the natural parameterization of the PGFs for readability, but note that the parameterization by μ and σ^2 can be adopted without loss of generality.

In the Poisson/equidispersed case, $\hat{\Gamma}_k(u; \lambda) = e^{\lambda(u-1)}$. Working out the derivatives is relatively straightforward:

$$\begin{aligned}
\frac{d^y}{du^y} \hat{\Gamma}(u; \lambda) &= \frac{d^y}{du^y} e^{\lambda(u-1)} \\
&= e^{-y} \frac{d^y}{du^y} e^{\lambda u} \\
&= e^{-\lambda} \left(\lambda^y e^{\lambda u} \right) \\
&= \lambda^y e^{\lambda(u-1)}
\end{aligned}$$

In the binomial/underdispersed case, $\hat{\Gamma}_k(u; n, p) = (1 - p + pu)^n$ has the form of a polynomial. Again, applying the chain rule to $\hat{\Gamma}_k(u; n, p)$ is fairly straightforward:

$$\begin{aligned}
\frac{d^y}{du^y} \hat{\Gamma}(u; n, p) &= \frac{d^y}{du^y} (1 - p + pu)^n \\
&= \left[\frac{d}{du} f(u) \right]^y \cdot \left[\frac{d^y}{df(u)^y} f(u)^n \right], f(u) = 1 - p + pu \\
&= p^y \cdot \frac{n!}{(n-y)!} f(u)^{n-y} \\
&= p^y \cdot \frac{n!}{(n-y)!} (1 - p + pu)^{n-y}
\end{aligned}$$

so long as $n \in \mathbb{Z}^+, y \in \mathbb{N}^0, n \geq y$.

Finally, in the negative binomial/overdispersed case, $\hat{\Gamma}_k(u; r, q) = \left(\frac{q}{1-u(1-q)}\right)^r$. Once more, applying the chain rule works out simply as follows:

$$\begin{aligned}
\frac{d^y}{du^y}\hat{\Gamma}(u; r, q) &= \frac{d^y}{du^y}\left(\frac{q}{1-u(1-q)}\right)^r \\
&= q^r \frac{d^y}{du^y}(1-u(1-q))^{-r} \\
&= q^r \left[\frac{d}{du}f(u)\right]^y \cdot \left[\frac{d^y}{df(u)^y}f(u)^{-r}\right], f(u) = 1-u(1-q) \\
&= q^r \left[(-1)^y(1-q)^y\right] \cdot \left[(-1)^y \frac{(r+y-1)!}{(r-1)!} f(u)^{-r-y}\right] \\
&= q^r (1-q)^y \frac{(r+y-1)!}{(r-1)!}(1-u(1-q))^{-r-y}
\end{aligned}$$

That the runtime is constant wrt u, y, μ, and σ^2 is trivial. □

We call the version of APGF-FORWARD which uses these closed form derivatives instead of autodiff APGF-FORWARD-S and provide complete pseudocode in Algorithm 11.

Theorem 6.3. *The runtime of the* APGF-FORWARD-S *algorithm is* $\mathcal{O}(K)$ *where* K *is the number of observations and does not depend on the magnitude of the observed counts at any time step.*

Proof. The proof of Theorem 6.3 follows directly from the proof of Theorem 6.2 and the replacement of the high-order derivative computation with a constant time routine. □

6.3 Experiments

In this section we investigate the performance tradeoff between approximate inference with APGF-FORWARD-S, approximate inference with the truncated forward algorithm (Dail & Madsen, 2011), and exact inference with GDUAL-FORWARD.

Algorithm 11 APGF-FORWARD-S

if $k = 0$ **then**
  [1:] **return** 1
[2:] $\Gamma_k(u_k) = A_{k-1}(F_k(u_k)) \cdot G_k(u_k)$
[3:] $\hat{\Gamma}_k(\cdot; \mu, \sigma^2) = \text{APGF}(\Gamma_k)$
if $\sigma^2 < \mu$ **then**
  $n = \text{round}\left(\frac{\mu^2}{\mu - \sigma^2}\right)$
  $p = \mu / n$
  [4:] $\hat{A}_k(s_k) = \frac{(s_k \rho_k)^{y_k}}{y_k!} \cdot p_k^{y_k} \frac{n!}{(n-y_k)!}(1 - p + p s_k(1-\rho_k))^{n-y_k}$
else if $\sigma^2 == \mu$ **then**
  $\lambda = \mu$
  [4:] $\hat{A}_k(s_k) = \frac{(s_k \rho_k)^{y_k}}{y_k!} \cdot \lambda e^{\lambda(s_k(1-\rho_k)-1)}$
else if $\sigma^2 > \mu$ **then**
  $r = \frac{\mu^2}{\sigma^2 - \mu}$
  $q = \mu / \sigma^2$
  [4:] $\hat{A}_k(s_k) = \frac{(s_k \rho_k)^{y_k}}{y_k!} \cdot q^r (1-q)^{y_k} \frac{(r+y_k-1)!}{(r-1)!} \left(1 - s_k(1-\rho_k)(1-q)\right)^{-r-y_k}$
[5:] **return** $\hat{A}_k(s_k)$

All experiments in this section were performed using simulated data from a latent branching process (see Chapter 3) with distributions defined by the following PGFs:

$$F_k(u_k) = \underbrace{e^{\gamma(u_k-1)}}_{\text{Poisson}} \cdot \underbrace{((1-\delta) + \delta u_k)}_{\text{Bernoulli}}, \qquad \text{(offspring)}$$

$$G_k(u_k) = e^{\iota_k(u_k-1)}, \qquad \text{(arrivals)}$$

$$y_k \sim \text{Binomial}(n_k, \rho), \qquad \text{(detection)}$$

where γ, δ, ι_k, and ρ define the recruitment rate (the mean offspring per individual per timestep), survival rate, immigration rate, and detection rate respectively. In general, these parameters are not time varying, although the initial arrival rate is parameterized independently and denoted λ (i.e. $\iota_0 = \lambda$) following the standard of Dail & Madsen (2011) and `unmarked` (Fiske & Chandler, 2011).

All experiments were conducted on a 2012 MacBook Pro with a 2.3 GHz Intel Core i7 with 16GB 1600 MHz DDR3 RAM running macOS version 10.14.6. All experiments were conducted in R version 3.4.1.

6.3.1 Likelihood approximation quality and runtime

We evaluated the approximation quality of APGF-FORWARD-S in two ways: how does the accuracy of log-likelihood calculation change as the evaluation point moves further from the generative parameters and how do the accuracy and runtime change as the parameters of the generative model change?

Results of the experiments are shown in Figures 6.2, 6.3, and 6.4. Each sample of data consists of $M = 3$ IID sites with $T = 6$ equally spaced observations. The arrival and offspring distributions in all experiments were Poisson. This corresponds to the TREND dynamics of `unmarked` Fiske & Chandler (2011). We repeated these experiments with other arrival and offspring distributions and got similar results.

In the first set of experiments, shown in Figure 6.2, we took a single sample of data using the following generative model parameters $\theta_0 = \{\lambda_0 = 60, \gamma_0 = 0.95, \rho_0 = 0.5, \iota_0 = 4\}$, where λ_0, γ_0, ρ_0, and ι_0 are the initial arrival rate, the recruitment rate, the detection rate, and the immigration rate, respectively. We then evaluated how the NLL changes as you vary the parameters away from θ_0. Four results are shown in Figure 6.2, each varying one of the four model parameters while keeping the other 3 fixed to the generative values.

In general, the difference in NLL between GDUAL-FORWARD (which computes the exact likelihood) and APGF-FORWARD-S is negligible, suggesting that the approximation in APGF is a reasonable one in this setting. Note that although the offspring and arrival distributions are Poisson, the messages in the forward algorithm are perhaps "Poisson-like" but they are *not* marginally Poisson. If this were the case, APGF would not be an approximation and APGF-FORWARD-S would be exact. Indeed, in practice

we see that the approximating distribution is almost always either the binomial case or the negative binomial case (though again, the true distribution is likely neither of these).

In the second set of experiments, shown in Figures 6.3 and 6.4, we varied one of the parameters of the generative model to produce a new set of parameters θ^*, sampled 30 data samples, and then compared the mean NLL at θ^* on each sample and mean runtime to compute each value using GDUAL-FORWARD and APGF-FORWARD-S. In each figure, 4 results are shown where one of the four model parameters is varied while the rest are kept at $\theta_0 = \{\lambda_0 = 80, \gamma_0 = 0.95, \rho_0 = 0.5, \iota_0 = 8\}$.

Again, the difference in NLL between APGF-FORWARD-S and GDUAL-FORWARD is negligible in all cases. However, APGF-FORWARD-S is as much six times faster than GDUAL-FORWARD in our trials, and demonstrates constant runtime scaling with respect to each of the 4 model parameters, which is consistent with our theoretical analysis. Furthermore, as we demonstrated empirically in Chapter 5, GDUAL-FORWARD is generally significantly faster than the standard approximate method TRUNC, so by transitivity, APGF-FORWARD-S is also significantly faster than TRUNC. We investigate the relationship between APGF-FORWARD-S and TRUNC in more detail in Section 6.3.2 and Chapter 7.

6.3.2 Parameter estimation

We compared the accuracy of maximum likelihood parameter estimates obtained by optimizing the approximate likelihood with APGF-FORWARD-S versus those obtained by optimizing TRUNC.

We varied the detection probability ρ from 0.05 to 0.95 and then learned the other 4 model parameters (listed with their generative values: $\lambda = 20$, $\iota = 3$, $\gamma = 0.7$, and $\delta = 0.25$) simultaneously, treating ρ as fixed to its true (varying) value. In each trial we generated 10 samples. In the first two rows of Figure 6.5 we show the RMSE

of the MLE estimates from each method relative to the generating parameters. In general, there is a small cost paid from using APGF-FORWARD-S, but qualitatively the resultant MLE parameters are very similar between both methods.

In the bottom left panel of Figure 6.5, we took the MLE parameters from each trial and plotted the exact NLL (using GDUAL-FORWARD) of the estimates from TRUNC vs the exact NLL of the estimates from APGF-FORWARD-S. In general, the true NLL of the MLE parameters from both methods are very similar, suggesting that each approximation technique results in small trade offs between the different model parameters.

Finally, in the bottom right panel of Figure 6.5, we show the average runtime of parameter estimation using each method. In these experiments, using APGF-FORWARD-S for parameter estimation was between 3 to 18 times faster than using TRUNC, though in general as population size increases, the gap between the two methods will only widen.

6.3.3 Discussion

Across our experiments, our approximate algorithm, APGF-FORWARD-S, seems to be a very good approximation to the exact algorithm, GDUAL-FORWARD, and performs very similarly to the state of the art truncated algorithm, TRUNC. This is, of course, predicated on the assumption that the APGF distribution in Equation 6.3 is a good approximation to the true factors in GDUAL-FORWARD. In our experience with both real and simulated data, the filtered marginals computed using GDUAL-FORWARD are indeed generally unimodal and approximately Poisson (with potentially significant over- or underdispersion). It is unclear whether the LBP family can be configured in such a way that APGF would be a poor approximation, but if generalizing this technique to other model families, similar care should be taken to evaluate the appropriateness of APGF.

According to our experiments, the cost of adopting APGF-FORWARD-S in terms of approximation quality or in parameter estimation appears to be negligible, while the potential gain in runtime is significant. In fact, the empirical runtime results presented are, if anything, likely misleading to the detriment of APGF-FORWARD-S. The experiments we present here are relatively small in scale, but as the population size increases, the gap between the runtime of APGF-FORWARD-S (which does not depend on population size) and that of all other existing methods will only grow.

There are two important caveats to note about the results in this section. Firstly, note that the truncation hyperparameter $N_{\max}$ is being configured naively. $N_{\max}$ represents the upper bound on the support in TRUNC. The schedule we use for setting $N_{\max} = \max(y) + 20$ is the default behavior in `unmarked` (Fiske & Chandler, 2011), but in practice authors have developed their own techniques for configuring this hyperparameter (Royle, 2004; Gross et al., 2007; Dail & Madsen, 2011). In Chapters 4 and 5 our comparisons with TRUNC used a much more conservative approach to setting K. With a more conservative setting of K, we would expect TRUNC to perform similarly to the other methods at the cost of a potentially significant increase in computation time. We discuss this in more detail in Chapter 7.

Secondly, there are several well-known parameter confounding issues in parameter estimation for LBPs. The most widely-studied of these is the conflation of the overall population size and the detection probability. We explore this tradeoff in more detail in Chapter 7, but address it here by keeping the detection probability, ρ, fixed in our parameter estimation experiments. However, it is still possible for the other model parameters to trade off in complex ways (i.e. the likelihood surface is relatively flat in a reasonably large region around the true MLE parameters). We see this in the relatively chaotic plots of RMSE in Figure 6.5. The plot of exact NLL in that figure, however suggests that both methods do roughly the same in terms of finding parameters that explain the data well, even if the exact MLE estimates from each

method differ slightly. This implies that care should be taken when interpreting MLE estimates from any of these methods in a biological sense.

Overall, APGF-FORWARD-S (and, by extension, APGF-FORWARD) has excellent performance with regards to likelihood approximation. At the same time, APGF-FORWARD-S also offers a *fundamentally different degree of scaling* with population size (constant vs superquadratic) relative to all existing exact or approximate inference algorithms for LBPs.

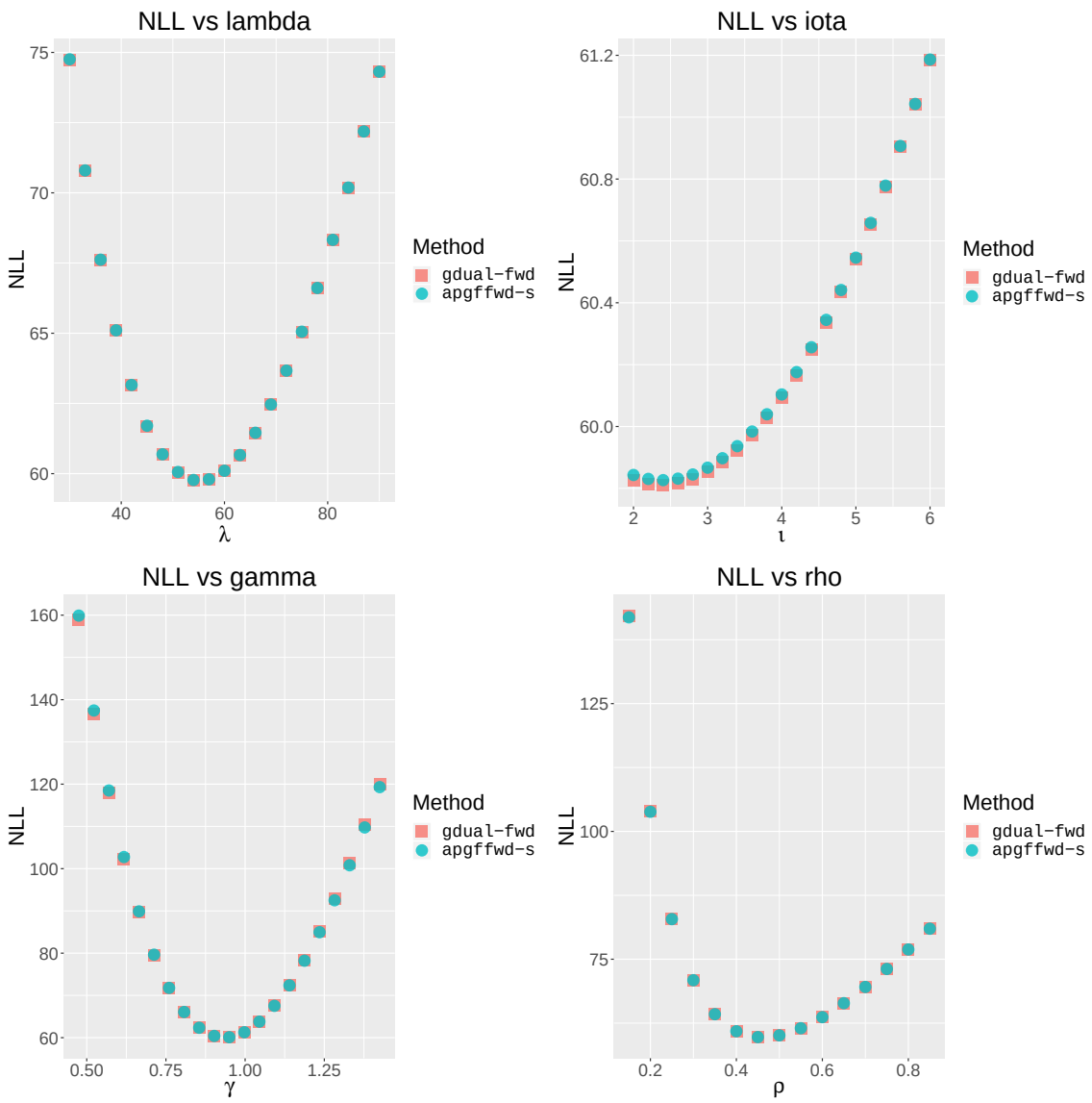

Figure 6.2. Comparison of NLL accuracy between APGF-FORWARD-S and GDUAL-FORWARD on a fixed data sample as the evaluation point is varied. GDUAL-FORWARD shows the exact NLL while APGF-FORWARD-S is an approximation to the true likelihood. In each plot, a different model parameter is varied while the rest are kept at the generative values.

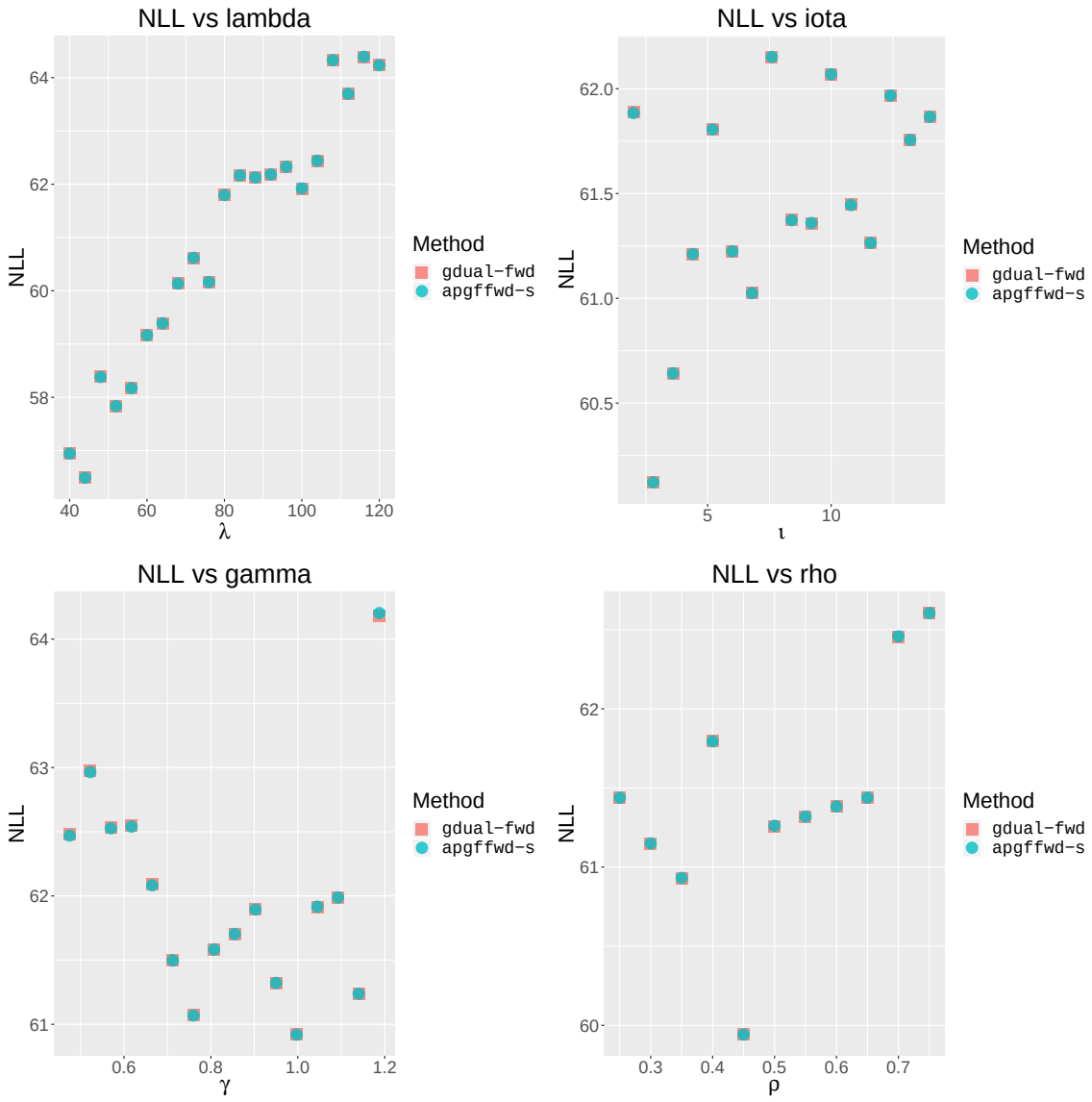

Figure 6.3. Comparison of NLL accuracy between APGF-FORWARD-S and GDUAL-FORWARD at generative parameter values as generative parameters are varied. GDUAL-FORWARD shows the exact NLL while APGF-FORWARD-S is an approximation to the true likelihood.

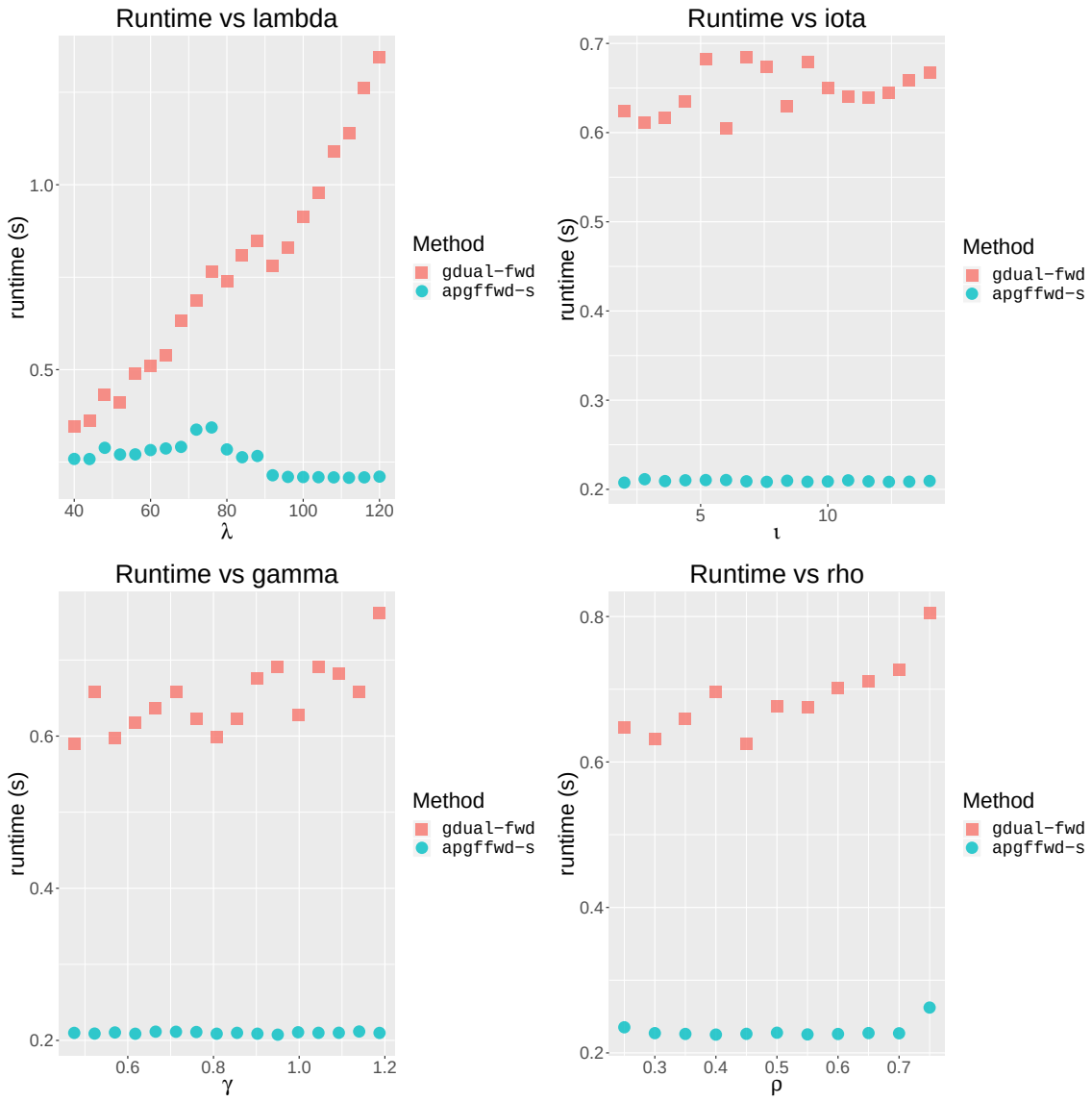

Figure 6.4. Comparison of runtime between APGF-FORWARD-S and GDUAL-FORWARD at generative parameter values as generative parameters are varied. GDUAL-FORWARD shows the exact NLL while APGF-FORWARD-S is an approximation to the true likelihood.

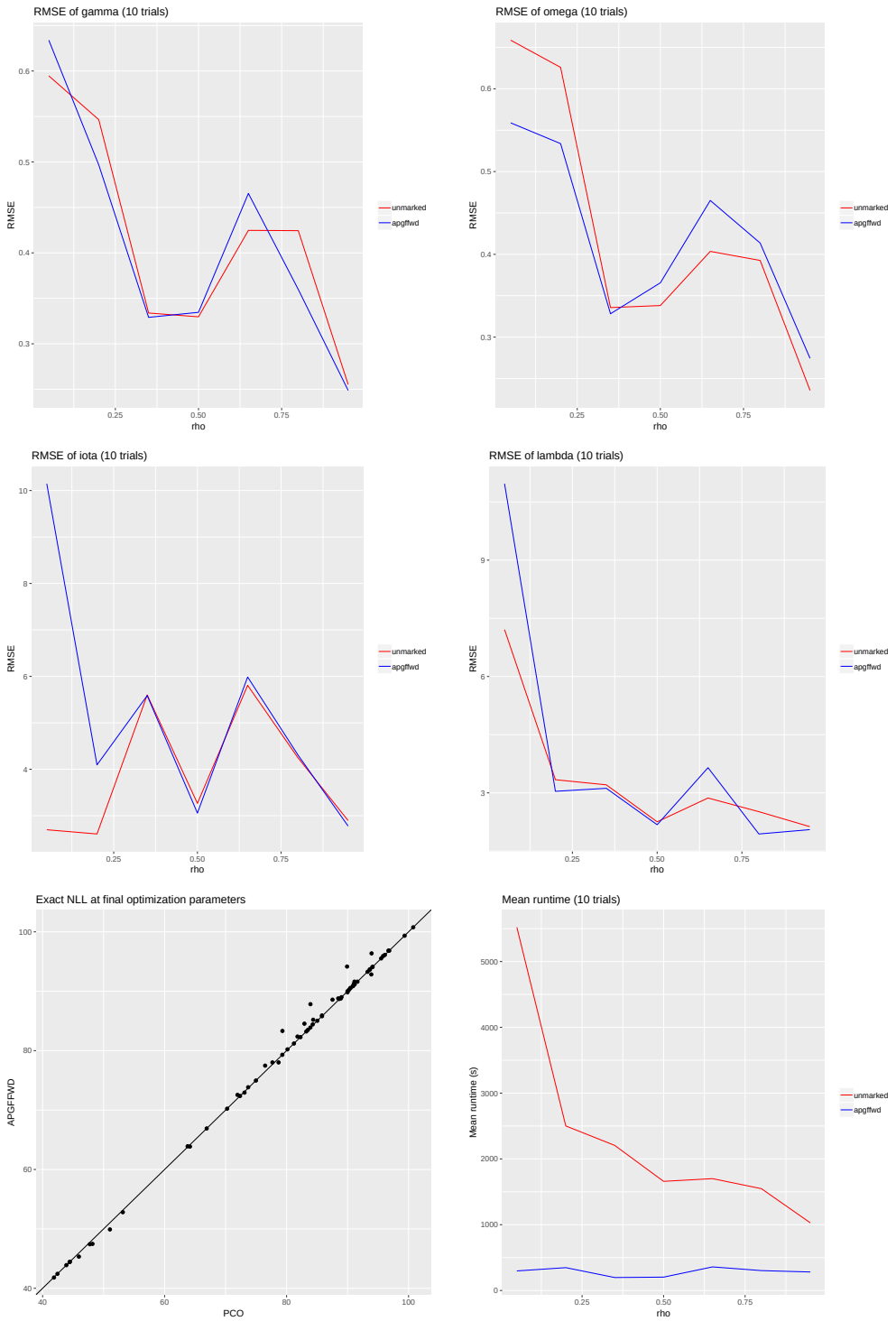

Figure 6.5. Parameter estimation with APGF-FORWARD-S. Top two rows show RMSE for estimates of 4 different model parameters as the detection probability ρ is varied. Bottom left panel plots the exact NLL at the final parameter estimates using TRUNC against those from APGF-FORWARD-S for each trial. Bottom right panel shows the mean runtime.

CHAPTER 7

POPULATION DYNAMICS CASE STUDY WITH `UNMARKED`

7.1 Introduction

The PGF-based techniques we have developed in the preceding chapters for exact and approximate inference in latent branching processes (LBPs) have been shown to have significant advantages over existing techniques in simulated settings. The asymptotic scaling of the APGF-FORWARD-S algorithm from Chapter 6 in particular offers a fundamentally new capability to scale population dynamics models to population sizes far beyond what has been studied using existing methods.

To investigate the capabilities of our PGF-based algorithms in real-world settings, we incorporated GDUAL-FORWARD, APGF-FORWARD, and APGF-FORWARD-S into `unmarked` (Fiske & Chandler, 2011), the most widely used R package for fitting models to populations of unmarked individuals, i.e. populations where individuals cannot be differentiated from one another. In Section 7.2 we show how the models currently supported by `unmarked` relate to the LBP framework we introduced in Chapter 3.

Using our R implementation of APGF-FORWARD-S, we conducted a scalability case study on data from the North American Breeding Bird Survey (BBS) (Pardieck et al., 2019). We fit six LBP models to six years of count data from three highly abundant bird species in Massachusetts and present results in Section 7.3.

It is our hope that adopting our flexible LBP framework will allow future researchers to easily develop and test novel, complex models of population dynamics in a black-box fashion and then deploy them to datasets collected at a variety of scales.

7.1.1 Related Work

Much attention has been given to the related questions of *occupancy* (the presence/absence of a species over a geographic region) and *species richness* (the variety of species in a given region over a particular time) using the BBS record (see e.g. Boulinier et al., 1998; Nichols et al., 2001; Altwegg & Nichols, 2019). Within this literature, several authors have used the BBS (and other studies with large geographic/temporal extent) to estimate species detectability (Boulinier et al., 1998; Nichols et al., 2001) and species accumulation (Dorazio et al., 2006; MacKenzie et al., 2017), directly addressing the imperfect detection of point-count survey data. While these questions are closely related to ours, the treatment of variables are binary presence/absence variables leads to a fundamentally different set of inference problems.

Large-scale dynamics analyses with count-valued variables have been performed with the BBS dataset. Many studies have focused on modeling the trend in the observed counts over time (Geissler & Sauer, 1990; Link & Sauer, 1994; Sauer et al., 1994; Ralph et al., 1995; Link & Sauer, 1998; Hines et al., 1999; Rosenberg et al., 2019). In general, these approaches either performed regression on the observed counts directly, or focused their inference on the issue of estimating detectability, using simple models for the inter-observation population dynamics. Hostetler & Chandler (2015) applied a number of `unmarked` dynamics models to BBS data using the TRUNC method. As the authors note in that paper, computational limitations prevented experimentation with additional models in their case study.

7.2 Model

`Unmarked` (Fiske & Chandler, 2011) supports six models for population dynamics: 'constant', 'notrend', 'trend', 'autoreg', 'Ricker', and 'Gompertz'. All six of these models can be shown to be special cases of the latent branching process model described in Chapter 3. Furthermore, the models with density-independent offspring

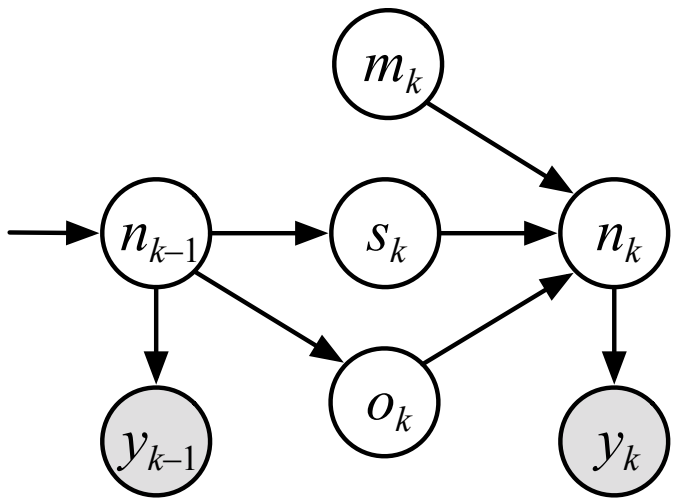

Figure 7.1. The structure of `unmarked` dynamics models. The variables n_k and y_k are the latent abundance and observed abundance, respectively, m_k is the number of immigrants, s_k is the number of survivors, and o_k is the number of offspring.

processes ('constant', 'notrend', 'trend', and 'autoreg') are amenable to automatic inference using our PGF-based methods from Chapters 5 and 6.

The six `unmarked` dynamics models use a common parameterization, though not every model uses all of these parameters. The parameters are:

λ: the rate of initial arrivals (at time $t = 0$),

ι: the rate of immigration (at times $t > 0$),

ω: the survival rate,

γ: the offspring (recruitment) rate, and

ρ: the detection rate.

Similarly, each of the `unmarked` dynamics models is a hidden Markov model (HMM) with a common structure. This structure is described graphically in Figure 7.1. In these models, n_k is the abundance at time k, y_k is the observed count at time k, m_k is the number of immigrants between time $k-1$ and time k, s_k is the

Dynamics	Immigr. (m_k)	Survival (s_k)	Offspring (o_k)
CONSTANT	$\text{Poiss}(\gamma)$	$\text{Binom}(n_{k-1}, \omega)$	0
NOTREND	$\text{Poiss}((1-\omega)\lambda)$	$\text{Binom}(n_{k-1}, \omega)$	0
TREND	$\text{Poiss}(\iota)$	0	$\text{Poiss}(n_{k-1} \cdot \gamma)$
AUTOREG	$\text{Poiss}(\iota)$	$\text{Binom}(n_{k-1}, \omega)$	$\text{Poiss}(n_{k-1} \cdot \gamma)$
RICKER	$\text{Poiss}(\iota)$	0	$\text{Poiss}\left(n_{k-1} \exp\left(\gamma\left(1 - \frac{n_{k-1}}{\omega}\right)\right)\right)$
GOMPERTZ	$\text{Poiss}(\iota)$	0	$\text{Poiss}\left(n_{k-1} \exp\left(\gamma\left(1 - \frac{\log(n_{k-1}+1)}{\log(\omega+1)}\right)\right)\right)$

Table 7.1. Transition distributions for `unmarked` dynamics.

number of survivors from time $k-1$ to time k, and o_k is the number of offspring from time $k-1$ to time k. Note that in `unmarked`, counts may be collected across multiple sites and/or counts may be replicated multiple times at each counting occasion. These extensions to the LBP model are trivial and omitted from our presentation here only for clarity. Our implementation in `unmarked` supports both extensions natively.

In this HMM framework, the dynamics models of each `unmarked` model are defined by the initial distribution of abundance, the transition distributions of m_k, s_k, and o_k, and the observation distribution of y_k. For all six models, the initial arrival distribution is Poisson with rate λ and the observation distribution is Binomial with count n and rate ρ. The transition distributions for each model are shown in Table 7.1.

In all cases, $n_k = m_k + s_k + o_k$. Note that in `unmarked`, the 3 parameters of dynamics (ω, γ, and ι) can vary between sites, but not over time. This restriction is not true in our LBP framework where all parameters may vary across time and between sites.

To express these dynamics models as LBPs, we must convert the transition dynamics above into an offspring PGF F and an arrival PGF G. The offspring PGF describes the number of "offspring" per individual per time step which, notably, may

Dynamics	Offspring PGF	Arrival PGF
CONSTANT	$F_{\text{Bernoulli}}(s;\omega)$	$F_{\text{Poisson}}(s;\gamma)$
NOTREND	$F_{\text{Bernoulli}}(s;\omega)$	$F_{\text{Poisson}}(s;(1-\omega)\lambda)$
TREND	$F_{\text{Poisson}}(s;\gamma)$	$F_{\text{Poisson}}(s;\iota)$
AUTOREG	$F_{\text{Bernoulli}}(s;\omega)\cdot F_{\text{Poisson}}(s;\gamma)$	$F_{\text{Poisson}}(s;\iota)$
RICKER	$F_{\text{Poisson}}\left(s;\exp\left(\gamma\left(1-\frac{n_{k-1}}{\omega}\right)\right)\right)$	$F_{\text{Poisson}}(s;\iota)$
GOMPERTZ	$F_{\text{Poisson}}\left(s;\exp\left(\gamma\left(1-\frac{\log(n_{k-1}+1)}{\log(\omega+1)}\right)\right)\right)$	$F_{\text{Poisson}}(s;\iota)$

Table 7.2. PGFs for `unmarked` dynamics.

include both the recruitment process and the survival process[1]. The arrival PGF describes the number of "immigrants" to the population, which are assumed to be independent of the current population size.

To convert the transition distributions for s_k and o_k into an offspring PGF, the following well-known relationships are useful to remember:

$$\text{if } X=\sum_{i=1}^{n} x_i, \quad \text{and } x_i \sim \text{Bernoulli}(p), \quad \text{then } X \sim \text{Binomial}(n,p).$$

$$\text{if } X=\sum_{i=1}^{n} x_i, \quad \text{and } x_i \sim \text{Poisson}(\lambda), \quad \text{then } X \sim \text{Poisson}(n\lambda).$$

In Table 7.2 we show the corresponding PGFs for each of the `unmarked` dynamics models. Note that while the density-dependent dynamics Ricker and Gompertz fit the LBP framework, it is not currently clear whether they are compatible with our PGF inference methodology. Specifically, the appearance of n_{k-1} in the PGFs for Ricker and Gompertz implies that the individual offspring distributions are not independent of one another and therefore Proposition 4.4 does not apply. As a result, these two dynamics models are not currently supported by our R implementations of

[1]In other words, there is nothing fundamentally different between the two processes except, traditionally, the distribution used for each.

PGF inference, though future work may be able to add support for these and other density dependent dynamics.

7.2.1 Beyond `unmarked`

While the LBP family neatly generalizes the models supported by `unmarked`, the family also extends beyond those models currently in use in `unmarked`. In Chapter 5, we demonstrated LBPs where the immigration distribution was overdispersed (via a negative binomial) and LBPs where the offspring distribution was a geometric distribution. Indeed the LBP framework allows modelers to use virtually any discrete distribution (or combination thereof) for the offspring and immigration distributions. Further work could also extend the family to allow observation distributions besides the Binomial.

7.3 Case Study

To demonstrate the potential of adopting a LBP representation and performing approximate inference with APGF-FORWARD-S, we conducted a case study on data from the North American Breeding Bird Survey (BBS) (Pardieck et al., 2019). We collected count data from Massachusetts for 3 abundant species: House Sparrow (HS; *Passer domesticus*), Ovenbird (OB; *Seiurus aurocapilla*), and Wood Thrush (WT; *Hylocichla mustelina*).

The BBS is a multi-species monitoring program that has been conducted continuously in the United States and Canada since 1966. BBS data is collected once a year (typically in June) at over 4000 sites by a team of volunteer experts. Each site consists of a roughly 24.5 mile long roadside route divided as evenly as possible into 50 stops. At each stop, a 3-minute survey is conducted wherein every bird seen or heard within a 0.25-mile radius is recorded. Most routes in the BBS dataset are reliably recorded in almost every year, but in the experiments that follow, we excluded

in each experiment any route where more than half of the years in question had missing data. For 2013-2018, this meant there were 14, 16, and 12 routes included for house sparrows, ovenbirds, and wood thrushes, respectively and for 1966-2018, there were 17 routes included for all three species. The total counts over all routes for the 2013-2018 dataset were (HS: 3110, OB: 2526, WT: 458) and the average count per route per year was (HS: 35.7, OB: 24.5, WT: 5.1). For the 1966-2018 dataset, the total counts were (HS: 23230, OB: 14359, WT: 9675) and the average counts were (HS: 27.8, OB: 15.8, WT: 11.2).

For each species, we used APGF-FORWARD-S in the framework of `unmarked` and TRUNC to fit several LBPs to the BBS counts. Data from each route was treated as a separate site and routes that were missing data during the study period were dropped. The model parameters were shared across all sites and optimized jointly with the exception of the initial arrival rate, λ. Analysis of each species was performed independently, i.e. no data or parameters were shared across species.

It is well-known in the literature that statistically speaking it can be difficult to differentiate between the case of a large population with small detection probability and the case of a small population with large detection probability (Banks-Leite et al., 2014; Gervasi et al., 2014; Dennis et al., 2015). A number of methods have been developed in recent years to independently estimate the detection probability (e.g. Nichols et al., 2000; Schmidt & Pellet, 2009; Dejean et al., 2012) by modifying the study design or collecting additional covariate data.

The truncated algorithm of Dail & Madsen (2011) (TRUNC) inadvertently avoids this problem by artificially limiting the maximum possible abundance. In our experiments with TRUNC, we tie $N_{\max}$, the maximum abundance to a species specific minimum value for the detection probability ρ, $\rho_{\min}(s)$ by setting $N_{\max}(s, \mathbf{y}) = \frac{\max_k y_k}{\rho_{\min}(s)}$. Intuitively, this is setting $N_{\max}$ to be the maximum possible abundance given some assumption about the lowest possible detection rate we expect. For our three species,

Species	Method	Dynamics	R	ι	ρ	AIC	Mean rt (s)
House Sparrow	APGFFWD	TREND+IMM	1.001	7.300	0.213	725.8	214.9
House Sparrow	TRUNC	TREND+IMM	0.871	3.540	0.618	1560.0	9539.0
Ovenbird	APGFFWD	TREND+IMM	1.022	1.330	0.379	493.7	240.5
Ovenbird	TRUNC	CONSTANT	0.928	1.434	0.703	1071.2	810.0
Wood Thrush	APGFFWD	TREND+IMM	0.856	1.918	0.335	308.7	308.7
Wood Thrush	TRUNC	TREND+IMM	0.651	1.833	0.733	414.2	95.3

Table 7.3. MLE parameters of min AIC models for 6 years of Breeding Bird Survey counts of 3 species fit using APGF-FORWARD-S and TRUNC. In each case, 6 models were fit with both APGF-FORWARD-S and TRUNC. MLE estimates are given for growth rate R, mean immigration rate ι, and detection probability ρ from the model with the lowest AIC. Average runtime of fitting all 6 models is reported along with the AIC as computed by the corresponding algorithm.

we used the following values for $\rho_{\min}$: $\rho_{\min}(s) = \{s = \text{'HS'} : 0.5; s = \text{'OB'} : 0.3; s = \text{'WT'} : 0.1\}$.

Another way to address this identifiability problem is to incorporate covariate data. While covariate data is sometimes available in the BBS record, our focus in this case study is on scalability, so we developed a simpler tactic that applies generically to all species for the purposes of this case study. For both APGF-FORWARD-S and TRUNC, the detection probability ρ was a learned parameter and the initial arrival rate λ_i for each site i was a fixed parameter derived from ρ and the vector of counts $\mathbf{y}_i$ from site i:

$$\lambda_i = \frac{\mathbf{y}_i[1]}{\rho}. \tag{7.1}$$

Setting λ_i this way is essentially setting it to a value which is consistent with the detection probability. While this is not guaranteed to be the maximum likelihood value of λ_i, it allows us to estimate ρ and the rest of the parameters simultaneously given only the count data.

In Table 7.3 we show the MLE estimates of three quantities (R, ι, and ρ) using APGF-FORWARD-S and TRUNC from the model with minimum AIC for each species using data from 2013-2018. We also report the dynamics model used in the minimum

Species	Dynamics	R	ι	ρ	Mean rt (s)
House Sparrow	NOTREND	0.851	0.386	0.386	3885.6
Ovenbird	TREND+IMM	1.002	0.698	0.281	3918.4
Wood Thrush	TREND+IMM	0.965	0.713	0.279	4596.3

Table 7.4. MLE parameters of min AIC models for 53 years of Breeding Bird Survey counts of 3 species fit using APGF-FORWARD-S. In each case, 6 models were fit. MLE estimates are given for growth rate R, mean immigration rate ι, and detection probability ρ from the model with the lowest AIC. Average runtime of fitting all 6 models is reported.

AIC result as well as the average time to fit each of the models. The parameter R is the growth rate (the mean number of offspring per individual per time step), ι is the mean immigration per timestep (in trials with immigration enabled), and ρ is the detection rate.

In Table 7.4 we show the same quantities using all data from 1966-2018 fit with only APGF-FORWARD-S. We excluded TRUNC from this comparison for computational reasons. For wood thrush (the smallest population), fitting a CONSTANT dynamics model (the fewest parameters) with TRUNC, took 6 hours. Fitting the same model to house sparrow data took over 10 hours with TRUNC.

Finally, complete MLE parameters for all models are available in Table 7.5 (for the 2013-2018 dataset) and Table 7.6 (for the 1966-2018 dataset).

7.3.1 Discussion

In general, in the 2013-2018 experiment there was little difference in the time taken by APGF-FORWARD-S to fit each model to the different species. The small variation in average runtime we observed could easily be due to details of the optimization schedule. By comparison, the runtime of TRUNC varied greatly across the three species, with the model fitting on house sparrows taking, on average, 100 times longer than on wood thrushes, even given the much more conservative truncation parameter used for wood thrushes (based on assuming $\rho_{\min}(HS) = 0.5$ and $\rho_{\min}(WT) = 0.1$).

This demonstrates well the scale-dependence of TRUNC on the population size and count magnitude.

The actual MLE parameter estimates returned by the two methods also display interesting disparities. We see for instance in the results from TRUNC significantly higher estimates of the detection probability ρ than with APGF-FORWARD-S. For the smaller populations (ovenbirds and wood thrushes), the MLE estimate of ρ computed by APGF-FORWARD-S was within the range we assumed in our configuration of ρ_{min}, which suggests that this is not just an artifact of our assumptions when setting ρ_{min} but indeed a more fundamental effect of using a truncated distribution in the first place. As we demonstrated in Chapter 6, APGF-FORWARD-S does a good job approximating the true likelihood without significant biases across a range of parameter regimes, so we believe it is preferable for producing trustworthy MLE estimates at these scales. The disparities in AIC computed by each method are also likely due to the values of $\rho_(\text{min})$ as we see smaller discrepancies in the species where $\rho_(\text{min})$ was small (Wood thrushes) than in the species where it was larger (House sparrows).

When we increased the amount of available data to include the entire BBS archive (1966-2018), we saw the runtime of APGF-FORWARD-S increase roughly 16-fold. This is fairly consistent with the rougly 9-fold increase in the number of years and roughly 1.2x increase in the number of routes included in the longer study. By comparison, in the cases we tried the runtime of TRUNC increased less consistently with the number of years included, but still took significantly longer than APGF-FORWARD-S. This is somewhat surprising given a key speedup implemented in `unmarked` for TRUNC: the model parameters are not allowed to change over time, even with covariates, meaning that computing the matrix of transition probabilities, the largest computational bottleneck, can be done once and reused across the entire chain. Our PGF techniques are not limited to being time-homogenous, and our implementation therefore does not implement any type of caching of this type. There is a potential improvement

by implementing similar caching strategies in the PGF domain, but we note that our technique still outperforms the standard truncated approach with respect to runtime even without these engineering improvements.

Species	Method	Dynamics	R	ι	ρ	AIC	Runtime (s)
House Sparrow	APGFFWD	NOTREND	0.814	1.391	0.134	934.1	112.1
House Sparrow	TRUNC	NOTREND	0.716	0.946	0.300	2626.1	5404.1
House Sparrow	APGFFWD	CONSTANT	0.995	37.28	0.051	790.8	537.3
House Sparrow	TRUNC	CONSTANT	0.866	4.724	0.585	1795.9	6028.0
House Sparrow	APGFFWD	TREND	1.037	-	0.256	737.3	59.9
House Sparrow	TRUNC	TREND	0.915	-	0.614	1572.9	5365.6
House Sparrow	APGFFWD	AUTOREG	1.037	-	0.256	739.3	119.0
House Sparrow	TRUNC	AUTOREG	0.914	-	0.614	1574.9	12987.9
House Sparrow	APGFFWD	TREND+IMM	1.001	7.300	0.213	725.8	145.8
House Sparrow	TRUNC	TREND+IMM	0.871	3.540	0.618	1560.0	4845.5
House Sparrow	APGFFWD	AUTOREG+IMM	1.001	7.300	0.213	727.8	315.1
House Sparrow	TRUNC	AUTOREG+IMM	0.871	3.538	0.618	1562.0	22602.9
Ovenbird	APGFFWD	NOTREND	0.586	0.709	0.585	562.0	323.6
Ovenbird	TRUNC	NOTREND	0.934	0.098	0.674	1071.9	339.6
Ovenbird	APGFFWD	CONSTANT	0.999	3.095	0.273	534.8	193.2
Ovenbird	TRUNC	CONSTANT	0.928	1.435	0.703	1071.2	398.4
Ovenbird	APGFFWD	TREND	1.044	-	0.378	494.07	121.4
Ovenbird	TRUNC	TREND	0.992	-	0.835	972.2	260.8
Ovenbird	APGFFWD	AUTOREG	1.044	-	0.378	496.07	220.6
Ovenbird	TRUNC	AUTOREG	0.992	-	0.836	974.2	1047.6
Ovenbird	APGFFWD	TREND+IMM	1.022	1.330	0.379	493.7	190.7
Ovenbird	TRUNC	TREND+IMM	0.963	0.889	0.843	971.2	360.4
Ovenbird	APGFFWD	AUTOREG+IMM	1.022	1.330	0.379	495.7	393.2
Ovenbird	TRUNC	AUTOREG+IMM	0.964	0.888	0.843	973.2	2453.2
Wood Thrush	APGFFWD	NOTREND	0.702	0.992	0.300	342.8	398.6
Wood Thrush	TRUNC	NOTREND	0.700	0.383	0.783	462.3	40.6
Wood Thrush	APGFFWD	CONSTANT	0.826	2.885	0.294	323.1	433.6
Wood Thrush	TRUNC	CONSTANT	0.554	2.900	0.639	432.5	45.1
Wood Thrush	APGFFWD	TREND	0.982	-	0.295	310.0	121.0
Wood Thrush	TRUNC	TREND	0.863	-	0.625	421.3	33.7
Wood Thrush	APGFFWD	AUTOREG	0.982	-	0.295	312.0	248.7
Wood Thrush	TRUNC	AUTOREG	0.863	-	0.625	423.3	158.4
Wood Thrush	APGFFWD	TREND+IMM	0.856	1.918	0.335	308.7	245.0
Wood Thrush	TRUNC	TREND+IMM	0.651	1.833	0.733	414.2	33.1
Wood Thrush	APGFFWD	AUTOREG+IMM	0.856	1.918	0.335	310.7	396.9
Wood Thrush	TRUNC	AUTOREG+IMM	0.659	1.791	0.729	416.2	260.8

Table 7.5. MLE parameters of all models for 6 years of Breeding Bird Survey counts of 3 species fit using APGF-FORWARD-S and TRUNC. MLE estimates are given for growth rate R, mean immigration rate ι, and detection probability ρ from each model along with AIC and runtime.

Species	Dynamics	R	ι	ρ	AIC	Runtime (s)
House Sparrow	NOTREND	0.851	0.386	0.386	10804.4	5553.2
House Sparrow	CONSTANT	0.997	2.758	0.102	12216.4	5384.1
House Sparrow	TREND	1.010	-	0.351	6612.0	1309.8
House Sparrow	TREND+IMM	1.010	0.001	0.351	6614.0	2294.3
House Sparrow	AUTOREG+IMM	1.010	0.001	0.351	6616.0	4886.5
Ovenbird	NOTREND	0.911	0.133	0.667	5568.2	3811.0
Ovenbird	CONSTANT	0.978	2.145	0.335	5022.8	8634.8
Ovenbird	TREND	1.016	-	0.258	4150.6	1297.8
Ovenbird	AUTOREG	1.016	-	0.258	4152.6	2812.7
Ovenbird	TREND+IMM	1.003	0.698	0.281	4143.2	2402.8
Ovenbird	AUTOREG+IMM	1.003	0.698	0.281	4145.2	4551.2
Wood Thrush	NOTREND	0.943	0.133	0.429	4901.1	2621.2
Wood Thrush	CONSTANT	0.997	2.758	0.102	12216.4	5384.1
Wood Thrush	TREND	0.983	-	0.258	4047.7	1604.6
Wood Thrush	AUTOREG	0.983	-	0.258	4049.7	4706.0
Wood Thrush	TREND+IMM	0.965	0.713	0.279	4041.0	2284.1
Wood Thrush	AUTOREG+IMM	0.965	0.713	0.279	4043.0	6882.2

Table 7.6. MLE parameters of all models for 53 years of Breeding Bird Survey counts of 3 species fit using APGF-FORWARD-S. MLE estimates are given for growth rate R, mean immigration rate ι, and detection probability ρ from each model along with AIC and runtime.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this work, we have detailed a new representation for count distributions and a set of inference and learning algorithms based on this representation. For models with latent count variables, our PGF representation generally trades an uncomputable operation (marginalization by numerical summation) for simple marginalization (via PGF evaluation, see Proposition 4.1) and difficult but tractable conditioning (via high order derivatives, see Proposition 4.2). Our exact inference algorithms (PGF-FORWARD, PGF-TAIL-ELIMINATE, GDUAL-FORWARD) demonstrate how to implement PGF inference efficiently and were the first exact inference methods for this class of model. Our approximate inference algorithm (APGF-FORWARD-S) represents a fundamental advance in scalability over existing approximate inference algorithms.

While our work has laid the groundwork for using PGFs for probabilistic inference, there is still significant room for further work that builds upon what we have presented. To start, there are numerous additional algorithms which could potentially be adapted to the PGF domain. Even for HMMs, the backward algorithm and the Baum-Welch algorithm are possible direct extensions from this work. In other models with latent counts, algorithms such as Hamiltonian Monte Carlo or Metropolis-Hastings could potentially benefit from a PGF representation.

The Unified Binomial Distribution we present in Chapter 6 may have additional applications. The Latent Branching Process model has a variety of extensions that could be added. The most significant of which is perhaps generalizing the observation process from a strict binomial distribution to an arbitrary observation distribution.

Finally, while the PGF inference toolkit we presented in Chapters 4 and 5 is applicable to multivariate PGFs, we encountered significant implementation and scalability challenges when implementing multivariate versions of these techniques, namely the combinatorial nature of multiple partial derivatives of multivariate PGFs. This is worth revisiting, potentially using some of the advancements that led to the development of APGF-FORWARD-S. Similarly, applying PGF inference to undirected graphical models is theoretically possible, but our initial explorations suggested that it held significant computational challenges.

BIBLIOGRAPHY

Al-Osh, M. A. and Alzaid, A. A. First-order integer-valued autoregressive (INAR(1)) process. *Journal of Time Series Analysis*, 8(3):261–275, 1987.

Altwegg, R. and Nichols, J. D. Occupancy models for citizen-science data. *Methods in Ecology and Evolution*, 10(1):8–21, 2019.

Banks-Leite, C., Pardini, R., Boscolo, D., Cassano, C. R., Püttker, T., Barros, C. S., and Barlow, J. Assessing the utility of statistical adjustments for imperfect detection in tropical conservation science. *Journal of Applied Ecology*, 51(4):849–859, 2014.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153), 2018.

Bickson, D. and Guestrin, C. Inference with Multivariate Heavy-Tails in Linear Models. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Blanghaps, N., Nov, Y., Weiss, G., and Others. Sojourn time estimation in an M/G/∞ queue with partial information. *Journal of Applied Probability*, 50(4): 1044–1056, 2013.

Boulinier, T., Nichols, J. D., Sauer, J. R., Hines, J. E., and Pollock, K. Estimating species richness: the importance of heterogeneity in species detectability. *Ecology*, 79(3):1018–1028, 1998.

Boyen, X. and Koller, D. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 33–42. Morgan Kaufmann Publishers Inc., 1998.

Brintz, B., Fuentes, C., and Madsen, L. An asymptotic approximation to the n-mixture model for the estimation of disease prevalence. *Biometrics*, 74(4):1512–1518, 2018.

Casella, G. and Berger, R. L. *Statistical Inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning, 2002. ISBN 9780534243128.

Chandler, R. No Title. URL http://www.inside-r.org/packages/cran/unmarked/docs/pcountOpen.

Chandler, R. B., Royle, J. A., and King, D. I. Inference about density and temporary emigration in unmarked populations. *Ecology*, 92(7):1429–1435, 2011. ISSN 0012-9658. doi: 10.1890/10-2433.1.

Chandler, R. B., Royle, J. A., et al. Spatially explicit models for inference about density in unmarked or partially marked populations. *The Annals of Applied Statistics*, 7(2):936–954, 2013.

Consul, P. C. and Jain, G. C. A Generalization of the Poisson Distribution. *Technometrics*, 15(4):791–799, 1973. ISSN 15372723. doi: 10.1080/00401706.1973.10489112. URL https://www.jstor.org/stable/pdf/1267389.pdf.

Conway, R. W. and Maxwell, W. L. A queuing model with state dependent service rates. *Journal of Industrial Engineering*, 12(2):132–136, 1962.

Couturier, T., Cheylan, M., Bertolero, A., Astruc, G., and Besnard, A. Estimating abundance and population trends when detection is low and highly variable: A comparison of three methods for the Hermann's tortoise. *Journal of Wildlife Management*, 77(3):454–462, 2013. ISSN 0022541X. doi: 10.1002/jwmg.499.

Dail, D. and Madsen, L. Models for Estimating Abundance from Repeated Counts of an Open Metapopulation. *Biometrics*, 67(2):577–587, jun 2011. ISSN 0006341X. doi: 10.1111/j.1541-0420.2010.01465.x. URL http://doi.wiley.com/10.1111/j.1541-0420.2010.01465.x.

Dejean, T., Valentini, A., Miquel, C., Taberlet, P., Bellemain, E., and Miaud, C. Improved detection of an alien invasive species through environmental dna barcoding: the example of the american bullfrog lithobates catesbeianus. *Journal of applied ecology*, 49(4):953–959, 2012.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Dennis, E. B., Morgan, B. J. T., and Ridout, M. S. Computational aspects of N-mixture models. *Biometrics*, 71(1):237–246, 2015. ISSN 1541-0420. doi: 10.1111/biom.12246. URL http://dx.doi.org/10.1111/biom.12246.

Dorazio, R. M., Royle, J. A., Söderström, B., and Glimskär, A. Estimating species richness and accumulation by modeling species occurrence and detectability. *Ecology*, 87(4):842–854, 2006.

Eick, S. G., Massey, W. A., and Whitt, W. The Physics of the $M_t/G/\infty$ Queue. *Operations Research*, 41(4):731–742, 1993.

Farrington, C. P., Kanaan, M. N., and Gay, N. J. Branching process models for surveillance of infectious diseases controlled by mass vaccination. *Biostatistics*, 4 (2):279–295, 2003.

Feller, W. *An Introduction to Probability Theory and Its Applications*. Wiley, 1968.

Fiske, I. and Chandler, R. unmarked: An R package for fitting hierarchical models of wildlife occurrence and abundance. *Journal of Statistical Software*, 43(10):1–23, 2011. URL `http://www.jstatsoft.org/v43/i10/`.

Geissler, P. H. and Sauer, J. R. Topics in route-regression analysis. Technical report, US Fish and Wildlife Service, 1990.

Gelfand, A. E. and Smith, A. F. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

Geman, S. and Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

Gervasi, V., Brøseth, H., Gimenez, O., Nilsen, E. B., and Linnell, J. D. C. The risks of learning: confounding detection and demographic trend when using count-based indices for population monitoring. *Ecology and Evolution*, 4(24):4637–48, 2014.

Ghahramani, Z. An introduction to hidden markov models and bayesian networks. In *Hidden Markov models: applications in computer vision*, pp. 9–41. World Scientific, 2001.

Griewank, A. and Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Griewank, A., Utke, J., and Walther, A. Evaluating higher derivative tensors by forward propagation of univariate taylor series. *Mathematics of Computation*, 69 (231):1117–1130, 2000.

Gross, K., Kalendra, E. J., Hudgens, B. R., and Haddad, N. M. Robustness and uncertainty in estimates of butterfly abundance from transect counts. *Population Ecology*, 49(3):191–200, 2007.

Haines, L. M. Maximum likelihood estimation for n-mixture models. *Biometrics*, 72 (4):1235–1245, 2016.

Heathcote, C. R. A Branching Process Allowing Immigration. *Journal of the Royal Statistical Society. Series B (Methodological)*, 27(1):138–143, 1965. URL `http://www.jstor.org/stable/2984491`.

Heskes, T. and Zoeter, O. Extended version of " expectation propagation for approximate inference in dynamic bayesian networks ". 2003.

Hines, J. E., Boulinier, T., Nichols, J. D., Sauer, J. R., and Pollock, K. H. Comdyn: software to study the dynamics of animal communities using a capture—recapture approach. *Bird study*, 46(sup1):S209–S217, 1999.

Horner, W. G. A new method of solving numerical equations of all orders, by continuous approximation. *Philosophical Transactions of the Royal Society of London*, 109:308–335, 1819.

Hostetler, J. A. and Chandler, R. B. Improved state-space models for inference about spatial and temporal variation in abundance from count data. *Ecology*, 96(6):1713–1723, 2015.

Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. Bayesian updating in causal probabilistic networks by local computations. *Computational statistics quarterly*, 1990.

Jha, A., Gogate, V., Meliou, A., and Suciu, D. Lifted inference seen from the other side: The tractable features. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 973–981, 2010.

Kalaba, R. and Tesfatsion, L. Automatic differentiation of functions of derivatives. *Computers & Mathematics with Applications*, 12(11):1091–1103, 1986.

Kéry, M., Royle, J. A., and Schmid, H. Modeling avian abundance from replicated counts using binomial mixture models. *Ecological applications*, 15(4):1450–1461, 2005.

Kéry, M., Dorazio, R. M., Soldaat, L., Van Strien, A., Zuiderwijk, A., and Royle, J. A. Trend estimation in populations with imperfect detection. *Journal of Applied Ecology*, 46:1163–1172, 2009. ISSN 00218901. doi: 10.1111/j.1365-2664.2009.01724.x.

Kvitkovicova, A. and Panaretos, V. M. Asymptotic Inference For Partially Observed Branching Processes. *Advances in Applied Probability*, 43(4):1166–1190, 2011. ISSN 00018678.

Lauritzen, S. L. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, 1992.

Lauritzen, S. L. and Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 157–224, 1988.

Lin, T.-Y., Winner, K., Bernstein, G., Mittal, A., Dokter, A. M., Horton, K. G., Nilsson, C., Van Doren, B. M., Farnsworth, A., La Sorte, F. A., Maji, S., and Sheldon, D. MistNet: Measuring historical bird migration in the US using archived weather radar data and convolutional neural networks. *Methods in Ecology and Evolution*, 2019. doi: 10.1111/2041-210X.13280.

Link, W. A. and Sauer, J. R. Estimating equations estimates of trends. *Bird Populations*, 2:23–32, 1994.

Link, W. A. and Sauer, J. R. Estimation of population trajectories from count data. *Biometrics*, pp. 488–497, 1997.

Link, W. A. and Sauer, J. R. Estimating population change from count data: application to the north american breeding bird survey. *Ecological applications*, 8(2): 258–268, 1998.

MacKenzie, D. I., Nichols, J. D., Royle, J. A., Pollock, K. H., Bailey, L., and Hines, J. E. *Occupancy estimation and modeling: inferring patterns and dynamics of species occurrence*. Elsevier, 2017.

Mao, Y. and Kschischang, F. R. On factor graphs and the Fourier transform. *IEEE Transactions on Information Theory*, 51(5):1635–1649, 2005. ISSN 0018-9448. doi: 10.1109/TIT.2005.846404.

Maybeck, P. S. *Stochastic models, estimation, and control*, volume 3. Academic press, 1982.

Mazerolle, M. J., Bailey, L. L., Kendall, W. L., Royle, J. A., Converse, S. J., and Nichols, J. D. Making great leaps forward: accounting for detectability in herpetological field studies. *Journal of Herpetology*, 41(4):672–690, 2007.

McKenzie, E. Ch. 16. Discrete variate time series. In *Stochastic Processes: Modelling and Simulation*, volume 21 of *Handbook of Statistics*, pp. 573–606. Elsevier, 2003.

Minka, T. P. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Murphy, K. P. *Machine learning: a probabilistic perspective*. 2012.

Nichols, J., Hines, J., Sauer, J., Fallon, F., Fallon, J., and Heglund, P. A double-observer approach for estimating detection probability and abundance from point counts. *The Auk*, 117(2):393–408, 2000.

Nichols, J. D., Sauer, J. R., Hines, J. E., Boulinier, T., and Pollock, K. H. Estimation of species richness and parameters reflecting community dynamics using data from ecological monitoring programs. 2001.

Panaretos, V. M. Partially observed branching processes for stochastic epidemics. *J. Math. Biol*, 54:645–668, 2007. doi: 10.1007/s00285-006-0062-6.

Pardieck, K., Jr., D. Z., Lutmerding, M., Aponte, V., and Hudson, M.-A. North american breeding bird survey dataset 1966-2018, version 2018.0. Technical report, U.S. Geological Survey, Patuxent Wildlife Research Center, 2019. URL `https://doi.org/10.5066/P9HE8XYJ`.

Pearl, J. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, feb 1989. ISSN 0018-9219. doi: 10.1109/5.18626.

Ralph, C. J., Sauer, J. R., and Droege, S. Monitoring bird populations by point counts. *Gen. Tech. Rep. PSW-GTR-149. Albany, CA: US Department of Agriculture, Forest Service, Pacific Southwest Research Station. 187 p*, 149, 1995.

Resnick, S. I. *Adventures in stochastic processes*. Springer Science & Business Media, 2013.

Rosenberg, K. V., Dokter, A. M., Blancher, P. J., Sauer, J. R., Smith, A. C., Smith, P. A., Stanton, J. C., Panjabi, A., Helft, L., Parr, M., and Marra, P. P. Decline of the north american avifauna. *Science*, 2019. ISSN 0036-8075. doi: 10.1126/science.aaw1313. URL `https://science.sciencemag.org/content/early/2019/09/18/science.aaw1313`.

Royle, J. A. N-mixture models for estimating population size from spatially replicated counts. *Biometrics*, 60(1):108–15, mar 2004. ISSN 0006-341X. doi: 10.1111/j.0006-341X.2004.00142.x. URL `http://www.ncbi.nlm.nih.gov/pubmed/15032780`.

Sauer, J. R., Peterjohn, B. G., and Link, W. A. Observer differences in the north american breeding bird survey. *The Auk*, 111(1):50–62, 1994.

Schmidt, B. R. and Pellet, J. Quantifying abundance: counts, detection probabilities, and estimates. *Amphibian ecology and conservation: a handbook of techniques*, pp. 465–479, 2009.

Sheldon, D., Winner, K., and Sujono, D. Learning in integer latent variable models with nested automatic differentiation. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pp. 4622–4630, 2018.

Shenoy, P. P. and Shafer, G. Axioms for probability and belief-function propagation. In *Uncertainty in Artificial Intelligence*, 1990.

Watson, H. W. and Galton, F. On the Probability of the Extinction of Families. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 4:138–144, 1875. ISSN 1098-6596. doi: 10.2307/2841222. URL `http://www.jstor.org/stable/2841222?origin=crossref`.

Wenger, S. J. and Freeman, M. C. Estimating species occurrence, abundance, and detection probability using zero-inflated distributions. *Ecology*, 89(10):2953–2959, 2008.

Wheeler, F. S. Bell polynomials. *ACM SIGSAM Bulletin*, 21(3):44–53, 1987.

Winner, K. and Sheldon, D. Probabilistic Inference with Generating Functions for Poisson Latent Variable Models. In *Advances in Neural Information Processing Systems 29*, 2016.

Winner, K., Bernstein, G., and Sheldon, D. Inference in a Partially Observed Queueing Model with Applications in Ecology. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pp. 2512–2520, 2015.

Winner, K., Sujono, D., and Sheldon, D. Exact inference for integer latent-variable models. In *International Conference on Machine Learning (ICML)*, pp. 3761–3770, 2017.

Winner, K., Noonan, M. J., Fleming, C. H., Olson, K., Mueller, T., Sheldon, D., and Calabrese, J. M. Statistical inference for home range overlap. *Methods in Ecology and Evolution*, 2018.

Xue, Y., Ermon, S., Lebras, R., Gomes, C. P., and Selman, B. Variable Elimination in Fourier Domain. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 1–10, 2016.

Zipkin, E. F., Thorson, J. T., See, K., Lynch, H. J., Grant, E. H. C., Kanno, Y., Chandler, R. B., Letcher, B. H., and Royle, J. A. Modeling structured population dynamics using data from unmarked individuals. *Ecology*, 95(1):22–29, jan 2014. ISSN 0012-9658. doi: 10.1890/13-1131.1. URL `http://doi.wiley.com/10.1890/13-1131.1`.

Zonneveld, C. Estimating death rates from transect counts. *Ecological Entomology*, 16(1):115–121, 1991.