Exact Inference for Integer Latent-Variable Models

Kevin Winner [1] **Debora Sujono** [1] **Dan Sheldon** [1 2]

Abstract

Graphical models with latent count variables arise in a number of areas. However, standard inference algorithms do not apply to these models due to the infinite support of the latent variables. Winner & Sheldon (2016) recently developed a new technique using probability generating functions (PGFs) to perform efficient, exact inference for certain Poisson latent variable models. However, the method relies on symbolic manipulation of PGFs, and it is unclear whether this can be extended to more general models. In this paper we introduce a new approach for inference with PGFs: instead of manipulating PGFs symbolically, we adapt techniques from the autodiff literature to compute the higher-order derivatives necessary for inference. This substantially generalizes the class of models for which efficient, exact inference algorithms are available. Specifically, our results apply to a class of models that includes branching processes, which are widely used in applied mathematics and population ecology, and autoregressive models for integer data. Experiments show that our techniques are more scalable than existing approximate methods and enable new applications.

1. Introduction

A key to the success of probabilistic modeling is the pairing of rich probability models with fast and accurate inference algorithms. Probabilistic graphical models enable this by providing a flexible class of probability distributions together with algorithms that exploit the graph structure for efficient inference. However, exact inference algorithms are only available when both the distributions involved and the graph structure are simple enough. However, this situation is rare and consequently, much research today is devoted to general-purpose approximate inference techniques (e.g. Ranganath et al., 2014; Kingma & Welling, 2014; Carpenter et al., 2016).

[1]College of Information and Computer Sciences, University of Massachusetts Amherst [2]Department of Computer Science, Mount Holyoke College. Correspondence to: Kevin Winner <kwinner@cs.umass.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

Despite many advances in probabilistic inference, there remain relatively simple (and useful) models for which exact inference algorithms are not available. This paper considers the case of graphical models with a simple structure but with (unbounded) latent count random variables. These are a natural modeling choice for many real world problems in ecology (Zonneveld, 1991; Royle, 2004; Dail & Madsen, 2011) and epidemiology (Farrington et al., 2003; Panaretos, 2007; Kvitkovicova & Panaretos, 2011). However, they pose a unique challenge for inference: even though algorithms like belief propagation (Pearl, 1986) or variable elimination (Zhang & Poole, 1994) are well defined mathematically, they cannot be implemented in an obvious way because factors have a countably infinite number of entries. As a result, approximations like truncating the support of the random variables or MCMC are applied (Royle, 2004; Gross et al., 2007; Chandler et al., 2011; Dail & Madsen, 2011; Zipkin et al., 2014; Winner et al., 2015).

Recently, Winner & Sheldon (2016) introduced a new technique for exact inference in models with latent count variables. Their approach executes the same operations as variable elimination, but with factors, which are infinite sequences of values, represented in a compact way using probability generating functions (PGFs). They developed an efficient exact inference algorithm for a specific class of Poisson hidden Markov models (HMMs) that represent a population undergoing mortality and immigration, and noisy observations of the population over time.

A key open question is the extent to which PGF-based inference generalizes to a broader class of models. There are two primary considerations. First, for what types of factors can the required operations (multiplication, marginalization, and conditioning) be "lifted" to PGF-based representations? Here, there is significant room for generalization: the mathematical PGF operations developed in (Winner & Sheldon, 2016) already apply to a broad class of non-Poisson immigration models, and we will generalize the models further to allow richer models of population survival and growth. Second, and more significantly, for what types of PGFs can the requisite mathematical operations be *implemented* efficiently? Winner & Sheldon (2016) manipulated PGFs *symbolically*. Their compact symbolic representation seems to rely crucially on properties of the Poisson distribution; it remains unclear whether symbolic PGF inference can be generalized beyond Poisson models.

This paper introduces 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.

Our algorithmic contribution leads to the first efficient exact algorithms for a class of HMMs that includes many well-known models as special cases, and has many applications. The hidden variables represent a population that undergoes three different processes: *mortality* (or *emigration*), *immigration*, and *growth*. A variety of different distributional assumptions may be made about each process. The models may also be viewed without this interpretation as a flexible class of models for integer-valued time series. Special cases include models from population ecology (Royle, 2004; Gross et al., 2007; Dail & Madsen, 2011), branching processes (Watson & Galton, 1875; Heathcote, 1965), queueing theory (Eick et al., 1993), and integer-valued autoregressive models (McKenzie, 2003). Additional details about the relation to these models are given in Section 2. 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.

2. Model and Problem Statement

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 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. Now, we can describe our model as:

$$N_k = (N_{k-1} \odot X_k) + M_k, \tag{1}$$

$$Y_k = \rho_k \circ N_k. \tag{2}$$

The variable N_k represents the population size at time t_k. The random variable $N_{k-1} \odot X_{k-1} = \sum_{i=1}^{N_{k-1}} X_{k-1,i}$ 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}. 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, 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.

Problem Statement We use lower case variables to denote specific settings of random variables. Let $y_{i:j} = (y_i, \ldots, y_j)$ and $n_{i:j} = (n_i, \ldots, n_j)$. The model above defines a joint probability mass function (pmf) $p(n_{1:K}, y_{1:K}; \theta)$ where we introduce the vector θ containing parameters of all component distributions when necessary. It is clear that the density factors according to a hidden Markov model: $p(n_{1:K}, y_{1:K}) =$

$\prod_{k=1}^{K} p(n_k \mid n_{k-1}) p(y_k \mid n_k)$. We will consider several inference problems that are standard for HMMs, but pose unique challenges when the hidden variables have countably infinite support. Specifically, suppose $y_{1:K}$ are observed, then we seek to:

- 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.

Connections to Other Models This model specializes to capture many different models in the literature. The latent process of Eq. (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 $X_k \sim \text{Bernoulli}(\delta_k)$, i.e., that the offspring process is binomial thinning of the current individuals. 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). For each of these models, we contribute the first known algorithms for exact inference and likelihood calculations when the process is *partially observed*. This allows estimation from data that is noisy and has variability that should not be modeled by the latent process.

Special cases of our model with noisy observations occur in statistical estimation problems in population ecology. 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) for estimating the size of an animal population from repeated counts. With Poisson immigration and Bernoulli offspring, we recover the basic model of Dail & Madsen (2011) for open metapopulations; extended versions with overdispersion and population growth also fall within our framework by using negative-binomial immigration and Poisson offspring. Related models for insect populations also fall within our framework (Zonneveld, 1991; Gross et al., 2007; Winner et al., 2015). The main goal in most of this literature is parameter estimation. Until very recently, no exact algorithms were known to compute the likelihood, so approximations such as truncating the support of the latent variables (Royle, 2004; Fiske & Chandler, 2011; Chandler et al., 2011; Dail & Madsen, 2011) or MCMC (Gross et al., 2007; Winner et al., 2015) were used. Winner & Sheldon (2016) introduced PGF-based exact algorithms for the restricted version of the model with Bernoulli offspring and Poisson immigration. We will build on that work to provide exact inference and likelihood algorithms for all of the aforementioned models.

3. 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). Winner & Sheldon (2016) showed how to implement the forward algorithm using PGFs for models with Bernoulli offspring and Poisson immigration.

Forward Algorithm The forward algorithm recursively computes "messages", which are unnormalized distributions of subsets of the variables. Specifically, define $\alpha_k(n_k) := p(n_k, y_{1:k})$ and $\gamma_k(n_k) := p(n_k, y_{1:k-1})$. These satisfy the recurrence:

$$\gamma_k(n_k) = \sum_{n_{k-1}} \alpha_{k-1}(n_{k-1}) p(n_k \mid n_{k-1}), \tag{3}$$

$$\alpha_k(n_k) = \gamma_k(n_k) p(y_k \mid n_k). \tag{4}$$

We will refer to Equation (3) as the *prediction step* (the value of n_k is predicted based on the observations $y_{1:k-1}$), and Equation (4) 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 (3) and (4). However, if n_k is unbounded, this cannot be done directly; for example, $\alpha_k(n_k)$ is an infinite sequence, and Equation (3) contains an infinite sum.

3.1. Forward Algorithm with PGFs

Winner & Sheldon (2016) 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. Specifically, define the PGFs $\Gamma_k(u_k)$ and $A_k(s_k)$ of $\gamma_k(n_k)$ and $\alpha_k(n_k)$, respectively, as:

$$\Gamma_k(u_k) := \sum_{n_k=0}^{\infty} \gamma_k(n_k) u_k^{n_k}, \tag{5}$$

$$A_k(s_k) := \sum_{n_k=0}^{\infty} \alpha_k(n_k) s_k^{n_k}. \tag{6}$$

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.

These functions capture all relevant information about the associated distributions. 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 normalizing, 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 nonnegative 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:

$$\Pr(X = r) = F^{(r)}(0)/r!, \tag{7}$$

$$\mathbb{E}[X] = F^{(1)}(1), \tag{8}$$

$$\mathrm{Var}(X) = F^{(2)}(1) - \left[F^{(1)}(1)\right]^2 + F^{(1)}(1). \tag{9}$$

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 of Winner & Sheldon (2016), which we generalize here, is the fact that there is also a recurrence relation among the PGFs.

Proposition 1. *Consider the probability model defined in Equations (1) and (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:*

$$\Gamma_k(u_k) = A_{k-1}\big(F_k(u_k)\big) \cdot G_k(u_k) \tag{10}$$

$$A_k(s_k) = \frac{(s_k \rho_k)^{y_k}}{y_k!} \cdot \Gamma_k^{(y_k)}\big(s_k(1-\rho_k)\big) \tag{11}$$

Proof. A slightly less general version of Equation (10) appeared in Winner & Sheldon (2016); the general version appears in the literature on branching processes with immigration (Heathcote, 1965). Equation (11) follows directly from general PGF operations outlined in (Winner & Sheldon, 2016). □

The PGF recurrence has the same two elements as the pmf recurrence in equations (3) and (4). Equation (10) 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 (11) 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.

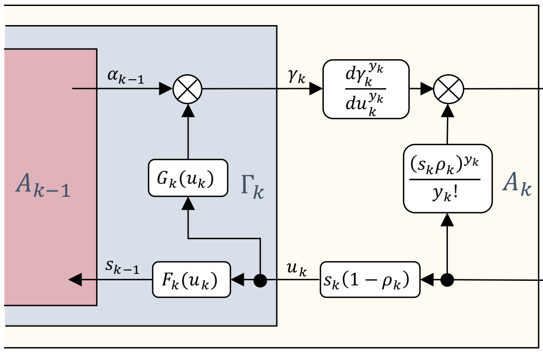

Figure 1. Circuit diagram of $A_k(s_k)$

3.2. Evaluating A_k via Automatic Differentiation

The recurrence reveals structure about A_k and Γ_k but does not immediately imply an algorithm. Winner & Sheldon (2016) 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 (7)–(9)). 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 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.

3.2.1. 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 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 $i \le \ell$, 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:

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

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 $j \le i$, 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 1 (Dual numbers)**.** *The generalized dual number $\langle v_\ell, dv_i \rangle_q$ for $0 \le i \le \ell$ 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 $0 \le i \le \ell$, 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.

3.2.2. 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 2 (Lifted Function)**.** *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a function of variables $x_1, \ldots, x_m$. The qth-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 $\big\langle f(x_1, \ldots, x_m), du \big\rangle_q$.*

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 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 $O(q^2)$ 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 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 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 $O(q^2 \log q)$ 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 $\langle v_\ell, v_i\rangle_q$. A detailed justification of this is given in the supplementary material. 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 $\epsilon^p = 0$ 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 $O(q \log q)$ using the FFT, so the overall running time is $O(q^2 \log q)$. □

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 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:*

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

The differential operator can be applied in $O(q)$ time.

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

3.2.3. The gdual-forward Algorithm

We will now use these operations to lift the function A_k to compute $\langle \alpha_k, s_k\rangle_q = \mathcal{L}A(\langle s_k, ds_k\rangle_q)$, i.e., the output of A_k and its derivatives with respect to its input. Algorithm 1 gives a sequence of mathematical operations to compute $A_k(s_k)$. Algorithm 2 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 2. 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 1 contains a non-primitive operation on Line 5: the derivative $d^{y_k}\gamma_k / du_k^{y_k}$. To evaluate this in Algorithm 2, 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 diagram of Figure 1, we calculate derivatives of each nested circuit with respect to its own input, starting with the innermost circuit and working out.

Theorem 2. *$\mathcal{L}A_K$ computes $\langle \alpha_k, ds_k\rangle_q$ in time $O\big(K(q + Y)^2 \log(q + Y)\big)$ 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 $O(KY^2 \log Y)$ time, and the first q moments or the first q entries of the filtered marginals can be computed in time $O\big(K(q + Y)^2 \log(q + Y)\big)$.*

Proof. To see that GDUAL-FORWARD is correct, note that it corresponds to Algorithm 1, 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 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 $O(K)$. 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

Algorithm 1 $A_k(s_k)$

```
    if $k = 0$ then
1:      return $\alpha_k = 1$
    end if
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 $\alpha_k$
```

Algorithm 2 $\mathcal{L}A_k(\langle s_k, ds_k \rangle_q)$ — GDUAL-FORWARD

```
    if $k = 0$ then
1:      return $\langle \alpha_k, ds_k \rangle_q = (1, 0, \ldots, 0)$
    end if
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 $\langle \alpha_k, ds_k \rangle_q$
```

dual numbers is $O(p^2 \log p)$ for dual numbers of order p, so the total is $O(K(q+Y)^2 \log(q+Y))$. $\square$

4. 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.

Running time vs Y. We compared the running time of GDUAL-FORWARD with the PGF-FORWARD algorithm from (Winner & Sheldon, 2016) as well as TRUNC, the standard truncated forward algorithm (Dail & Madsen, 2011). PGF-FORWARD is only applicable to the Poisson HMM from (Winner & Sheldon, 2016), which, in our terminology, 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 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 $O(KY^2)$ and $O(KY^2 \log Y)$, 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 $O(\frac{K}{\rho^2} Y^2 \log Y)$.

Figure 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.

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 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.

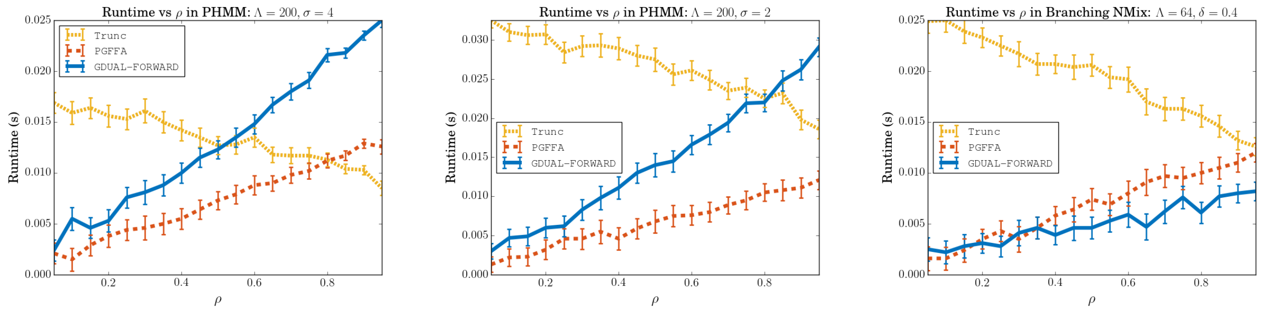

Figure 2. Runtime of GDUAL-FORWARD vs baselines. Left: PHMM. Center: PHMM-peaked. Right: NMix. See text for descriptions.

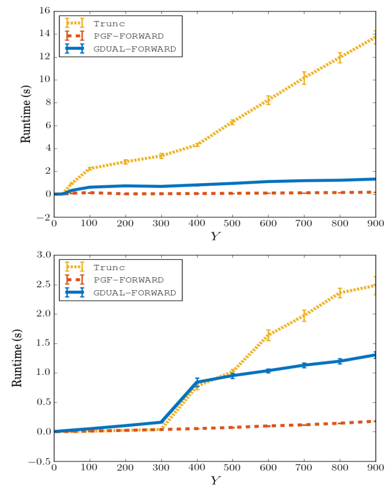

Figure 3. Running time vs. Y. Top: $\rho = 0.15$, bottom: $\rho = 0.85$.

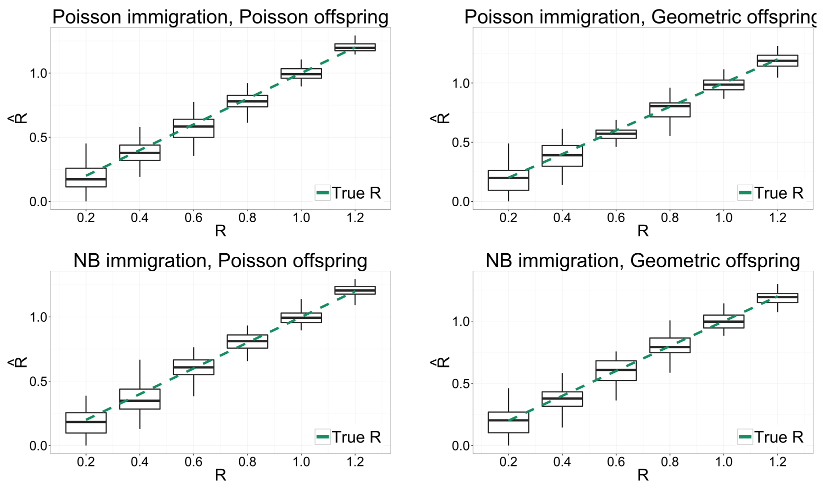

Figure 4. Estimates of R in different models. Titles indicate immigration and offspring distribution. 50 trials summarized as box plot for each model, parameter combination.

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.

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 $\lambda := E[M_k] = 6$ 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. 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 4 shows the distribution of 50 MLE estimates for R vs. the true values for each model. Results for two additional models appear in the supplementary material. 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.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1617533.

References

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. *Journal of Statistical Software*, 20, 2016.

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.

Dail, D. and Madsen, L. Models for estimating abundance from repeated counts of an open metapopulation. *Biometrics*, 67(2):577–587, 2011.

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.

Fiske, I. J. and Chandler, R. B. unmarked: An R package for fitting hierarchical models of wildlife occurrence and abundance. *Journal of Statistical Software*, 43:1–23, 2011.

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

Gross, K., J., Kalendra E., 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.

Heathcote, C. R. A branching process allowing immigration. *Journal of the Royal Statistical Society. Series B (Methodological)*, 27(1):138–143, 1965.

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.

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

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

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. 2014.

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. 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.

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.

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

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.

Ranganath, R., Gerrish, S., and Blei, D. M. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 814–822, 2014.

Royle, J. A. N-Mixture models for estimating population size from spatially replicated counts. *Biometrics*, 60(1): 108–115, 2004.

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.

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)*, pp. 2512–2520, 2015.

Zhang, N. L. and Poole, D. A simple approach to Bayesian network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, 1994.

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, 2014.

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