JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2025, VOL. 34, NO. 1, 222–238
https://doi.org/10.1080/10618600.2024.2350476

Variance-Reduced Stochastic Optimization for Efficient Inference of Hidden Markov Models

Evan Sidrow[a], Nancy Heckman[a], Alexandre Bouchard-Côté[a], Sarah M. E. Fortune[b], Andrew W. Trites[c], and Marie Auger-Méthé[d]

[a]Department of Statistics, University of British Columbia, Vancouver, Canada; [b]Department of Oceanography, Dalhousie University, Halifax, Canada; [c]Department of Zoology, Institute for the Oceans and Fisheries, University of British Columbia, Vancouver, Canada; [d]Department of Statistics, Institute for the Oceans and Fisheries, University of British Columbia, Vancouver, Canada

ABSTRACT

Hidden Markov models (HMMs) are popular models to identify a finite number of latent states from sequential data. However, fitting them to large datasets can be computationally demanding because most likelihood maximization techniques require iterating through the entire underlying dataset for every parameter update. We propose a novel optimization algorithm that updates the parameters of an HMM without iterating through the entire dataset. Namely, we combine a partial E step with variance-reduced stochastic optimization within the M step. We prove the algorithm converges under certain regularity conditions. We test our algorithm empirically using a simulation study as well as a case study of kinematic data collected using suction-cup attached biologgers from eight northern resident killer whales (*Orcinus orca*) off the western coast of Canada. In both, our algorithm converges in fewer epochs, with less computation time, and to regions of higher likelihood compared to standard numerical optimization techniques. Our algorithm allows practitioners to fit complicated HMMs to large time-series datasets more efficiently than existing baselines. Supplemental materials are available online.

ARTICLE HISTORY
Received October 2023
Accepted April 2024

KEYWORDS
Expectation-maximization algorithm; Maximum likelihood estimation; State space model; Statistical ecology; Stochastic gradient descent

1. Introduction

Hidden Markov models (HMMs) are statistical models for sequential data that are widely used to model time series in fields such as speech recognition (Gales and Young 2007), geology (Bebbington 2007), neuroscience (Kottaram et al. 2019), finance (Mamon and Elliott 2007), and ecology (McClintock et al. 2020). Such models are often used to predict a latent process of interest (e.g., a spoken phrase or an animal's behavioral state) from an observed time series (e.g., raw audio or time-depth data). Many practitioners estimate the parameters of an HMM by maximizing the likelihood function using either gradient-based numerical maximization or the expectation-maximization (EM) algorithm (Baum et al. 1970; Dempster, Laird, and Rubin 1977).

One serious concern for both numerical maximization and the EM algorithm is that every parameter update requires iterating though the full set of observations to calculate either the likelihood or its gradient. This concern is likely to only worsen in the future, as time-series datasets are increasingly collected at high frequencies and contain large numbers of observations (Patterson et al. 2017; Li, Han, and Song 2020). These datasets require progressively complex HMMs which can be computationally expensive to fit (Adam et al. 2019; Sidrow et al. 2022). In addition, many model validation techniques such as cross-validation require repeated parameter estimation which can be prohibitive even for relatively simple HMMs (Pohle et al. 2017).

Many inference techniques for independent datasets do not require iterating through the entire dataset to update a model's parameters. We henceforth refer to these techniques as *sub-linear* methods. One example of a sub-linear method is stochastic gradient descent (SGD), which estimates the gradient of the full likelihood using a random subset of the data (Robbins and Monro 1951). It is ubiquitous in the optimization literature and has inspired several extensions. For example, Johnson and Zhang (2013), Defazio, Bach, and Lacoste-Julien (2014) and Kingma and Ba (2015) reduce the variance of the gradient estimates compared to standard SGD, while Zinkevich et al. (2010) incorporate parallelization into SGD. Similarly, the incremental EM algorithm is a generalization of the EM algorithm that updates only a subset of hidden variables at each E step (Neal and Hinton 1998; Thiesson, Meek, and Heckerman 2001; Karimi et al. 2019). Both stochastic gradient descent and incremental EM assume that the underlying dataset is comprised of independent subsets. Such an assumption is sometimes reasonable for HMMs. For example, Gales and Young (2007) infer spoken words from many relatively short audio files, and they assume that each audio file is independent from one another. However, the assumption of independence is generally violated for HMMs designed for long, sequentially-dependent time series. Long time series are increasingly common in practice and are the focus of this article.

CONTACT Evan Sidrow *evan.sidrow@stat.ubc.ca* Department of Statistics, University of British Columbia, Vancouver, Canada.
Supplementary materials for this article are available online. Please go to *www.tandfonline.com/r/JCGS*.

Some work has been done to apply sub-linear inference methods to HMMs for long time series. For example, Gotoh, Hochberg, and Silverman (1998) divide a sequence of observations into segments and use the incremental EM algorithm to perform inference. This approach assumes that segments are independent of one another, which is not true in general. Alternatively, Ye, Ma, and Qian (2017) define a sufficiently large "buffer" before and after segments of data to minimize the effect of serial dependence. However, the appropriate size of the buffer can be difficult to calculate. More examples of sub-linear inference techniques are given by Khreich et al. (2012), who review on-line and incremental methods for HMM inference. However, most of these methods assume either that the M step of the EM algorithm is tractable (see sec. 3.2.1 of Khreich et al. 2012), or that the emissions of the HMM are discrete (Baldi and Chauvin 1993).

In this article, we introduce a new inference method for HMMs based on variance-reduced stochastic optimization. This inference method updates the HMM parameters without iterating through the entire dataset. Critically, it does not require a closed-form solution for its M step, does not require any buffer tuning, and does not introduce error into the HMM likelihood.

We begin with a formal definition of HMMs, a brief review of standard inference techniques for HMMs, and a description of stochastic optimization algorithms before introducing our algorithm. We then prove that our algorithm converges to a local maximum of the likelihood (under standard regularity assumptions) and note several practical considerations regarding its implementation. Finally, we compare the efficiency of our new algorithm to that of standard optimization techniques using several simulation studies and a kinematic case study of eight northern resident killer whales (*Orcinus orca*) off the western coast of Canada.

2. Background

2.1. Hidden Markov Models

HMMs are common statistical models used to describe time series that exhibit state-switching behavior. An HMM models an observed sequence of length T, $\mathbf{Y} = \{Y_t\}_{t=1}^T$, together with an unobserved (or "hidden") sequence $\mathbf{X} = \{X_t\}_{t=1}^T$. The hidden sequence $\mathbf{X}$ is a Markov chain, and each observation Y_t is a random variable, where Y_t given all other observations $(\mathbf{Y}\setminus\{Y_t\})$ and hidden states $(\mathbf{X})$ depends only on X_t. We assume $X_t \in \{1,\dots,N\}$ for some finite N. The unconditional distribution of X_1 is denoted by the row-vector $\boldsymbol{\delta} = \begin{pmatrix}\delta^{(1)} & \cdots & \delta^{(N)}\end{pmatrix}$, where $\delta^{(i)} = \mathbb{P}(X_1 = i)$. Further, the distribution of X_t for $t > 1$ conditioned on X_{t-1} is denoted by an N-by-N transition probability matrix

$$\boldsymbol{\Gamma}_t = \begin{pmatrix} \Gamma_t^{(1,1)} & \cdots & \Gamma_t^{(1,N)} \\ \vdots & \ddots & \vdots \\ \Gamma_t^{(N,1)} & \cdots & \Gamma_t^{(N,N)} \end{pmatrix}, \tag{1}$$

where $\Gamma_t^{(i,j)} = \mathbb{P}(X_t = j \mid X_{t-1} = i)$. For simplicity, we assume that $\boldsymbol{\Gamma}_t$ does not change over time (i.e., $\boldsymbol{\Gamma}_t = \boldsymbol{\Gamma}$ for all t) unless stated otherwise.

To ensure that all entries are positive and all rows sum to one, it is convenient to reparameterize the transition probability matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{N\times N}$ and initial distribution $\boldsymbol{\delta} \in \mathbb{R}^N$ in terms of auxiliary variables $\boldsymbol{\eta} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{\nu} \in \mathbb{R}^N$:

$$\Gamma^{(i,j)}(\boldsymbol{\eta}) = \frac{\exp(\eta^{(i,j)})}{\sum_{k=1}^N \exp(\eta^{(i,k)})}, \qquad \delta^{(i)}(\boldsymbol{\nu}) = \frac{\exp(\nu^{(i)})}{\sum_{k=1}^N \exp(\nu^{(k)})}, \tag{2}$$

where $i,j = 1,\dots,N$ and $\eta^{(i,i)}$ and $\nu^{(1)}$ are set to zero for identifiability. This formulation simplifies likelihood maximization by removing constraints in the optimization problem. One may also incorporate covariates into $\boldsymbol{\Gamma}$ by setting $\eta_t^{(i,j)}(\boldsymbol{\beta}) = \left(\boldsymbol{\beta}^{(i,j)}\right)^\top \mathbf{z}_t$, where $\mathbf{z}_t$ is a column vector of known covariates at time index t and $\boldsymbol{\beta}^{(i,j)}$ is a column vector of unknown regression coefficients. While $\boldsymbol{\Gamma}$ and $\boldsymbol{\delta}$ are functions of $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$, we abuse notation in future sections and treat $\boldsymbol{\Gamma}$ and $\boldsymbol{\delta}$ as variables since the mappings are bijections.

If $X_t = i$, then we denote the conditional density or probability mass function of Y_t as $f^{(i)}(\cdot;\theta^{(i)})$, where $\theta^{(i)}$ are the parameters describing the state-dependent distribution of Y_t. The collection of all state-dependent parameters is $\boldsymbol{\theta} = \{\theta^{(i)}\}_{i=1}^N$. For brevity, we denote the full set of parameters as $\boldsymbol{\phi} = \{\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\nu}\}$. Figure 1 shows an HMM as a graphical model.

Denote a fixed realization of observations $\mathbf{Y}$ and latent states $\mathbf{X}$ as $\mathbf{y} = \{y_t\}_{t=1}^T$ and $\mathbf{x} = \{x_t\}_{t=1}^T$. The joint likelihood of an HMM given $\mathbf{y}$ and $\mathbf{x}$ is

$$p(\mathbf{x},\mathbf{y};\boldsymbol{\phi}) = \delta^{(x_1)} f^{(x_1)}(y_1;\theta^{(x_1)}) \prod_{t=2}^T \Gamma^{(x_{t-1},x_t)} f^{(x_t)}(y_t;\theta^{(x_t)}). \tag{3}$$

Alternatively, the marginal likelihood of the observed data $\mathbf{y}$ alone is

$$p(\mathbf{y};\boldsymbol{\phi}) = \boldsymbol{\delta} P(y_1;\boldsymbol{\theta}) \prod_{t=2}^T \boldsymbol{\Gamma} P(y_t;\boldsymbol{\theta}) \mathbf{1}_N^\top, \tag{4}$$

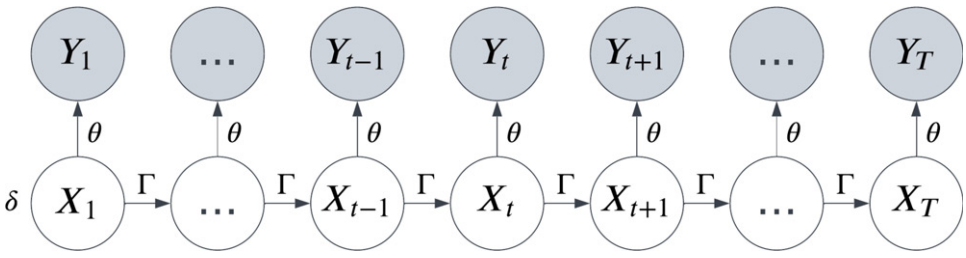

Figure 1. Graphical representation of an HMM. X_t corresponds to an unobserved latent state at time t whose distribution is described by a Markov chain. Y_t corresponds to an observation at time t, where Y_t given all other observations $\mathbf{Y} \setminus \{Y_t\}$ and hidden states $\mathbf{X}$ depends only on X_t.

where $\mathbf{1}_N$ is an N-dimensional row vector of ones and $P(y_t;\boldsymbol{\theta})$ is an $N \times N$ diagonal matrix with entry (i,i) equal to $f^{(i)}(y_t;\theta^{(i)})$. For a more complete introduction to HMMs, see Zucchini, Macdonald, and Langrock (2016).

2.2. State Decoding

One appealing feature of HMMs is that it is simple to determine the distribution of a given hidden state (X_t) conditioned on a set of observations $\mathbf{y}$. Assuming that the HMM parameters $\boldsymbol{\phi}$ are fixed, define the probability density of the observations between times s and t as $p(y_{s:t};\boldsymbol{\phi})$. Likewise, define *forward probabilities* $\alpha_t^{(i)} = p(y_{1:t}, X_t = i;\boldsymbol{\phi})$ (for $i = 1,\dots,N$ and $t = 1,\dots,T$) and *backward probabilities* $\beta_t^{(i)} = p(y_{(t+1):T} \mid X_t = i;\boldsymbol{\phi})$ (for $i = 1,\dots,N$ and $t = 1,\dots,T-1$). By convention, $\beta_T^{(i)} = 1$ for $i = 1,\dots,N$. We thus define the row vectors $\boldsymbol{\alpha}_t = \begin{pmatrix}\alpha_t^{(1)} & \cdots & \alpha_t^{(N)}\end{pmatrix}$ and $\boldsymbol{\beta}_t = \begin{pmatrix}\beta_t^{(1)} & \cdots & \beta_t^{(N)}\end{pmatrix}$. Both $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ can be calculated using the following recursions:

$$\boldsymbol{\alpha}_1 = \boldsymbol{\delta}\, P(y_1;\boldsymbol{\theta}), \qquad \boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\,\boldsymbol{\Gamma}\, P(y_t;\boldsymbol{\theta}), \quad t = 2,\dots,T, \tag{5}$$

$$\boldsymbol{\beta}_T^\top = \mathbf{1}_N^\top, \qquad \boldsymbol{\beta}_t^\top = \boldsymbol{\Gamma}\, P(y_{t+1};\boldsymbol{\theta})\,\boldsymbol{\beta}_{t+1}^\top, \quad t = 1,\dots,T-1. \tag{6}$$

We define conditional probabilities $\gamma_t^{(i)} = \mathbb{P}(X_t = i \mid \mathbf{Y} = \mathbf{y}\,;\,\boldsymbol{\phi})$ and $\xi_t^{(i,j)} = \mathbb{P}(X_{t-1} = i, X_t = j \mid \mathbf{Y} = \mathbf{y}\,;\,\boldsymbol{\phi})$. We also define the row vector $\boldsymbol{\gamma}_t = \begin{pmatrix}\gamma_t^{(1)} & \cdots & \gamma_t^{(N)}\end{pmatrix}$ and the matrix

$$\boldsymbol{\xi}_t = \begin{pmatrix} \xi_t^{(1,1)} & \cdots & \xi_t^{(1,N)} \\ \vdots & \ddots & \vdots \\ \xi_t^{(N,1)} & \cdots & \xi_t^{(N,N)} \end{pmatrix}, \qquad t = 2,\dots,T.$$

Both $\boldsymbol{\gamma}_t$ and $\boldsymbol{\xi}_t$ can be calculated from $\boldsymbol{\alpha}_{t-1}, \boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \boldsymbol{\Gamma}$, and $\boldsymbol{\theta}$. Let $\operatorname{diag}(\cdot)$ map a row vector to the diagonal matrix with that row vector as its diagonal. Then,

$$\gamma_t^{(i)} = \frac{\alpha_t^{(i)}\,\beta_t^{(i)}}{\boldsymbol{\alpha}_t\,\boldsymbol{\beta}_t^\top}, \qquad \boldsymbol{\gamma}_t = \frac{\boldsymbol{\alpha}_t \operatorname{diag}(\boldsymbol{\beta}_t)}{\boldsymbol{\alpha}_t\,\boldsymbol{\beta}_t^\top}, \tag{7}$$

$$\xi_t^{(i,j)} = \frac{\alpha_{t-1}^{(i)}\,\Gamma^{(i,j)}\,f^{(j)}(y_t;\theta^{(j)})\,\beta_t^{(j)}}{\boldsymbol{\alpha}_{t-1}\,\boldsymbol{\Gamma}\,P(y_t;\boldsymbol{\theta})\,\boldsymbol{\beta}_t^\top}, \tag{8}$$

$$\boldsymbol{\xi}_t = \frac{\operatorname{diag}(\boldsymbol{\alpha}_{t-1})\,\boldsymbol{\Gamma}\,P(y_t;\boldsymbol{\theta})\operatorname{diag}(\boldsymbol{\beta}_t)}{\boldsymbol{\alpha}_{t-1}\,\boldsymbol{\Gamma}\,P(y_t;\boldsymbol{\theta})\,\boldsymbol{\beta}_t^\top}. \tag{9}$$

For shorthand, we define the sets $\{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\xi}\} = \{\boldsymbol{\alpha}_t,\boldsymbol{\beta}_t,\boldsymbol{\gamma}_t,\boldsymbol{\xi}_t\}_{t=1}^T$ to summarize the conditional probabilities for all t. In future sections, when $\boldsymbol{\phi}$ is not fixed (e.g., during the parameter estimation procedures), we add an argument to the conditional probabilities and write $\{\boldsymbol{\alpha}(\boldsymbol{\phi}),\boldsymbol{\beta}(\boldsymbol{\phi}),\boldsymbol{\gamma}(\boldsymbol{\phi}),\boldsymbol{\xi}(\boldsymbol{\phi})\} = \{\boldsymbol{\alpha}_t(\boldsymbol{\phi}),\boldsymbol{\beta}_t(\boldsymbol{\phi}),\boldsymbol{\gamma}_t(\boldsymbol{\phi}),\boldsymbol{\xi}_t(\boldsymbol{\phi})\}_{t=1}^T$ to highlight the dependence on $\boldsymbol{\phi}$.

2.3. The Baum-Welch Algorithm

The Baum-Welch algorithm is an iterative algorithm used to estimate the parameters of an HMM. It predates the more general EM algorithm (Dempster, Laird, and Rubin 1977), but the two are equivalent when applied to HMMs. At iteration k of the EM algorithm, denote the current parameter estimate as $\boldsymbol{\phi}_k$. One iteration of the EM algorithm consists of an expectation (or E) step, followed by a maximization (or M) step. For the E step, the function value $Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k)$ is defined as the expected value of the joint log-likelihood $\log p(\mathbf{X},\mathbf{y};\boldsymbol{\phi})$ taken with respect to $\mathbf{X}$, where $\mathbf{X}$ has conditional probability mass function $p(\mathbf{X} \mid \mathbf{Y} = \mathbf{y};\boldsymbol{\phi}_k)$. For the M step, the next parameter estimate $\boldsymbol{\phi}_{k+1}$ is found by maximizing $Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k)$ with respect to $\boldsymbol{\phi}$:

$$Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k) = \mathbb{E}_{\boldsymbol{\phi}_k}\left[\log p(\mathbf{X},\mathbf{y};\boldsymbol{\phi}) \mid \mathbf{Y} = \mathbf{y}\right], \tag{10}$$

$$\boldsymbol{\phi}_{k+1} = \arg\max_{\boldsymbol{\phi}} Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k). \tag{11}$$

For notational convenience, we occasionally denote the set of conditional probabilities $\{\boldsymbol{\alpha}_t(\boldsymbol{\phi}_k),\boldsymbol{\beta}_t(\boldsymbol{\phi}_k),\boldsymbol{\gamma}_t(\boldsymbol{\phi}_k),\boldsymbol{\xi}_t(\boldsymbol{\phi}_k)\}$ as $\{\boldsymbol{\alpha}_{k,t},\boldsymbol{\beta}_{k,t},\boldsymbol{\gamma}_{k,t},\boldsymbol{\xi}_{k,t}\}$. Substituting (3) into (10) and performing some algebra yields a closed form expression for Q:

$$\begin{aligned} Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k) = &\sum_{i=1}^N \gamma_{k,1}^{(i)} \log \delta^{(i)}(\boldsymbol{\nu}) + \sum_{t=1}^T \sum_{i=1}^N \gamma_{k,t}^{(i)} \log f^{(i)}(y_t;\theta^{(i)}) \\ &+ \sum_{t=2}^T \sum_{i=1}^N \sum_{j=1}^N \xi_{k,t}^{(i,j)} \log \Gamma^{(i,j)}(\boldsymbol{\eta}). \end{aligned} \tag{12}$$

The conditional probabilities $\gamma_{k,t}^{(i)}$ and $\xi_{k,t}^{(i,j)}$ thus act as weights for $\log \delta^{(i)}(\boldsymbol{\nu})$, $\log f^{(i)}(y_t;\theta^{(i)})$, and $\log \Gamma^{(i,j)}(\boldsymbol{\eta})$ for $i,j = 1,\dots,N$. We thus refer to $\boldsymbol{\gamma}$ and $\boldsymbol{\xi}$ as weights in future sections. Detailed pseudocode for the E and the M step are given in Algorithms 1 and 2. In some simple scenarios, the maximization problem in (11) above has a closed-form solution. However, this maximization problem is not always straightforward and often requires numerical maximization techniques. We thus review different methods for numerical maximization via stochastic optimization.

Algorithm 1 `E-step`($\boldsymbol{\phi}$)

Require: Parameter value $\boldsymbol{\phi} = \{\boldsymbol{\theta},\boldsymbol{\eta},\boldsymbol{\nu}\}$.

$\boldsymbol{\delta} = \boldsymbol{\delta}(\boldsymbol{\nu}), \quad \boldsymbol{\Gamma} = \boldsymbol{\Gamma}(\boldsymbol{\eta})$

$\boldsymbol{\alpha}_1 = \boldsymbol{\delta}\, P(y_1;\boldsymbol{\theta})$

$\boldsymbol{\beta}_T^\top = \mathbf{1}_N^\top$

for $t = 2,\dots,T$ **do**

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\,\boldsymbol{\Gamma}\,P(y_t;\boldsymbol{\theta}), \quad \boldsymbol{\beta}_{T-t+1}^\top = \boldsymbol{\Gamma}\,P(y_{T-t+2};\boldsymbol{\theta})\,\boldsymbol{\beta}_{T-t+2}^\top$$

end for

$\boldsymbol{\gamma}_1 = \frac{\boldsymbol{\alpha}_1 \operatorname{diag}(\boldsymbol{\beta}_1)}{\boldsymbol{\alpha}_1\,\boldsymbol{\beta}_1^\top}$

for $t = 2,\dots,T$ **do**

$$\boldsymbol{\gamma}_t = \frac{\boldsymbol{\alpha}_t \operatorname{diag}(\boldsymbol{\beta}_t)}{\boldsymbol{\alpha}_t\,\boldsymbol{\beta}_t^\top}, \quad \boldsymbol{\xi}_t = \frac{\operatorname{diag}(\boldsymbol{\alpha}_{t-1})\,\boldsymbol{\Gamma}\,P(y_t;\boldsymbol{\theta})\operatorname{diag}(\boldsymbol{\beta}_t)}{\boldsymbol{\alpha}_{t-1}\,\boldsymbol{\Gamma}\,P(y_t;\boldsymbol{\theta})\,\boldsymbol{\beta}_t^\top}$$

end for

return $\{\boldsymbol{\alpha}_t,\boldsymbol{\beta}_t,\boldsymbol{\gamma}_t,\boldsymbol{\xi}_t\}_{t=1}^T$

Algorithm 2 `Baum-Welch`$(\boldsymbol{\phi}_0, K)$

Require: Initial parameter values $\boldsymbol{\phi}_0$, number of iterations K

1: **for** $k = 0, \ldots, K-1$ **do**

2: $\{\boldsymbol{\alpha}_{k,t}, \boldsymbol{\beta}_{k,t}, \boldsymbol{\gamma}_{k,t}, \boldsymbol{\xi}_{k,t}\}_{t=1}^{T} =$ `E-step`$(\boldsymbol{\phi}_k)$ ▷ E step

3: ▷ M step

$$\boldsymbol{\phi}_{k+1} = \arg\max_{\{\boldsymbol{\theta},\boldsymbol{\eta},\boldsymbol{\nu}\}} \sum_{i=1}^{N} \gamma_{k,1}^{(i)} \log \delta^{(i)}(\boldsymbol{\nu}) + \sum_{t=1}^{T}\sum_{i=1}^{N} \gamma_{k,t}^{(i)} \log f^{(i)}(y_t; \theta^{(i)}) + \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \xi_{k,t}^{(i,j)} \log \Gamma^{(i,j)}(\boldsymbol{\eta})$$

4: **end for**

5: **return** $\boldsymbol{\phi}_K$

2.4. Stochastic Optimization

Stochastic optimization involves a class of optimization methods that use random variables to maximize or minimize an objective function. We focus on optimization methods to minimize objective functions that can be written as a sum of many terms (Robbins and Monro 1951), namely to solve

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} F(\boldsymbol{\phi}), \quad \text{where} \quad F(\boldsymbol{\phi}) = \frac{1}{T}\sum_{t=1}^{T} F_t(\boldsymbol{\phi}). \tag{13}$$

We denote the parameter values at step m of an optimization scheme as $\boldsymbol{\phi}^{(m)} = \{\boldsymbol{\theta}^{(m)}, \boldsymbol{\eta}^{(m)}, \boldsymbol{\nu}^{(m)}\}$ to distinguish between the parameter values $\boldsymbol{\phi}_k$ at iteration k of the Baum-Welch algorithm. The bold parameters $\boldsymbol{\theta}^{(m)}$, $\boldsymbol{\eta}^{(m)}$, and $\boldsymbol{\nu}^{(m)}$ correspond to optimization step m and are not to be confused with $\theta^{(i)}$, $\eta^{(i,j)}$, and $\nu^{(i)}$ which correspond to hidden states i and j of the HMM. Standard gradient descent at a given step m with step size λ updates the parameter value $\boldsymbol{\phi}^{(m)}$ by moving in the direction of the negative gradient of F. Formally, the update step is $\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} - \lambda \nabla F(\boldsymbol{\phi}^{(m)})$, or $\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} - (\lambda/T)\sum_{t=1}^{T} \nabla F_t(\boldsymbol{\phi}^{(m)})$ for our problem. The step size λ is user-defined. It should be large enough so $\boldsymbol{\phi}^{(m+1)}$ moves quickly toward a minimum of F, but not so large that $\boldsymbol{\phi}^{(m+1)}$ "overshoots" the minimum. This update requires evaluating a gradient for all $t = 1, \ldots, T$, which can be prohibitively expensive if T is large.

In contrast, stochastic gradient descent (SGD) updates $\boldsymbol{\phi}$ using $\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} - \lambda_m \nabla F_{t_m}(\boldsymbol{\phi}^{(m)})$, where $t_m \in \{1, \ldots, T\}$ is selected uniformly at random at step m of the algorithm (Robbins and Monro 1951). Stochastic gradient descent reduces the amount of time between updates by using an unbiased estimate of the gradient to update $\boldsymbol{\phi}^{(m)}$. However, the gradient estimates can have very high variance, so stochastic gradient descent requires that the step size λ_m is smaller than that of full gradient descent. The step size must also decay to zero as $m \to \infty$ to ensure convergence. Further, SGD has slower convergence rates than full gradient descent (Schmidt, Le Roux, and Bach 2017).

Variance-reduced stochastic optimization techniques such as stochastic average gradient (SAG) (Schmidt, Le Roux, and Bach 2017), stochastic variance reduced gradient (SVRG) (Johnson and Zhang 2013), and stochastic average gradient accelerated (SAGA) (Defazio, Bach, and Lacoste-Julien 2014) enjoy the speed of stochastic gradient descent as well as the convergence rates of full gradient descent. These algorithms involve storing gradient approximations at each gradient step m, $\widehat{\nabla} F_t^{(m)}$ for $t = 1, \ldots, T$, whose average approximates the full gradient $\nabla F(\boldsymbol{\phi}^{(m)})$. The gradient approximations are updated at various stages in the optimization algorithm and are used to reduce the variance of the full gradient estimate. For example, SVRG and SAGA update $\boldsymbol{\phi}^{(m)}$ via $\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} - \lambda_m \left[\nabla F_{t_m}(\boldsymbol{\phi}^{(m)}) - \widehat{\nabla} F_{t_m}^{(m)} + \widehat{\nabla} F^{(m)}\right]$, where $t_m \in \{1, \ldots, T\}$ is chosen uniformly at random and $\widehat{\nabla} F^{(m)} = (1/T)\sum_{t=1}^{T} \widehat{\nabla} F_t^{(m)}$. Like SGD, this stochastic update has the same expectation as that of standard gradient descent, but it empirically exhibits lower variance than SGD and it is guaranteed to converge without decaying the step size λ_m (under certain regularity conditions). After updating the parameters at step m, SAGA updates the gradient approximation $\widehat{\nabla} F_{t_m}^{(m+1)}$ and recalculates the resulting gradient average $\widehat{\nabla} F^{(m+1)}$. Algorithm 3 outlines SVRG and SAGA in pseudocode, and we denote it as *variance-reduced stochastic optimization*, or `VRSO`.

Algorithm 3 `VRSO`$(F, \boldsymbol{\phi}^{(0)}, \lambda, A, M)$

Require: Loss function $F = \frac{1}{T}\sum_{t=1}^{T} F_t$, initial value $\boldsymbol{\phi}^{(0)}$, step size λ, algorithm $A \in \{\text{SVRG}, \text{SAGA}\}$, and number of iterations M.

1: **for** $t = 1, \ldots, T$ **do** ▷ initialize gradients

2: $\widehat{\nabla} F_t^{(0)} = \nabla F_t(\boldsymbol{\phi}^{(0)})$

3: **end for**

4: $\widehat{\nabla} F^{(0)} = (1/T)\sum_{t=1}^{T} \widehat{\nabla} F_t^{(0)}$

5: **for** $m = 0, \ldots, M-1$ **do**

6: Pick $t_m \in \{1, \ldots, T\}$ uniformly at random.

7: ▷ update parameters

$$\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} - \lambda\left[\nabla F_{t_m}(\boldsymbol{\phi}^{(m)}) - \widehat{\nabla} F_{t_m}^{(m)} + \widehat{\nabla} F^{(m)}\right] \tag{14}$$

8: $\widehat{\nabla} F_t^{(m+1)} = \widehat{\nabla} F_t^{(m)}$ for $t = 1, \ldots, T$ ▷ update gradient approximations and average

9: **if** $A =$ SAGA **then**:

$$\widehat{\nabla} F_{t_m}^{(m+1)} = \nabla F_{t_m}(\boldsymbol{\phi}^{(m)}) \tag{15}$$

$$\widehat{\nabla} F^{(m+1)} = \widehat{\nabla} F^{(m)} + \frac{1}{T}\left(\widehat{\nabla} F_{t_m}^{(m+1)} - \widehat{\nabla} F_{t_m}^{(m)}\right) \tag{16}$$

10: **end if**

11: **end for**

12: **return** $\boldsymbol{\phi}^{(M)}$

SAGA involves running Algorithm 3 until convergence. However, the gradient approximations $\widehat{\nabla} F_t^{(m)}$ are never updated when using SVRG within Algorithm 3. As such, SVRG requires repeatedly running Algorithm 3 with M scaling approximately with T so the set of gradient approximations remains up-to-date.

3. Stochastic Optimization for HMM Inference

Both the E step and the M step of the Baum-Welch algorithm are expensive when the length of the observation sequence (T) is large. The E step is expensive because $\boldsymbol{\gamma}_t(\boldsymbol{\phi}_k)$ and $\boldsymbol{\xi}_t(\boldsymbol{\phi}_k)$ must

be calculated for $t = 1, \ldots, T$ to define $Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k)$. If closed-form solutions to (11) are not readily available, then the M step is also expensive because evaluating full gradients of $Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k)$ takes $\mathcal{O}(T)$ time. In this section, we introduce an original algorithm that speeds up both the expensive M step as well as the expensive E step of the Baum-Welch algorithm.

3.1. Variance-Reduced Stochastic M Step

To speed up the expensive M step, we notice from (12) that Q is a large sum and thus implement variance-reduced stochastic optimization. It is straightforward to reframe the M step of iteration k of the Baum Welch algorithm from (11) so it looks like the minimization problem from (13). To do so, we define $\boldsymbol{\xi}_1 = \emptyset$ and the loss function $F(\cdot \mid \boldsymbol{\gamma}, \boldsymbol{\xi})$ as follows:

$$F(\boldsymbol{\phi} \mid \boldsymbol{\gamma}, \boldsymbol{\xi}) = \frac{1}{T} \sum_{t=1}^{T} F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t), \qquad \text{where} \tag{17}$$

$$F_1(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_1, \boldsymbol{\xi}_1) = -\sum_{i=1}^{N} \gamma_1^{(i)} \log f^{(i)}(y_t; \theta^{(i)}) - \sum_{i=1}^{N} \gamma_1^{(i)} \log \delta^{(i)}(\boldsymbol{v}), \tag{18}$$

$$F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t) = -\sum_{i=1}^{N} \gamma_t^{(i)} \log f^{(i)}(y_t; \theta^{(i)}) - \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_t^{(i,j)} \log \Gamma^{(i,j)}(\boldsymbol{\eta}), \quad t = 2, \ldots, T. \tag{19}$$

The two functions F and Q are closely related to one another, as $F(\boldsymbol{\phi} \mid \boldsymbol{\gamma}(\boldsymbol{\phi}_k), \boldsymbol{\xi}(\boldsymbol{\phi}_k)) = -Q(\boldsymbol{\phi} \mid \boldsymbol{\phi}_k)/T$. However, we make a distinction between the two to bridge the gap between existing EM literature (which uses Q) and stochastic optimization literature (which uses F). At any iteration k of the EM algorithm, the loss function $F(\cdot \mid \boldsymbol{\gamma}(\boldsymbol{\phi}_k), \boldsymbol{\xi}(\boldsymbol{\phi}_k))$ can be minimized using Algorithm 3.

There are additional reasons to use SAGA and SVRG within the Baum-Welch algorithm beyond the standard benefits of variance-reduced stochastic optimization. Traditionally, SAGA is more memory intensive than SVRG because the gradient at every index must be stored. However, the Baum-Welch algorithm involves storing weights for each time index t to define $F(\cdot \mid \boldsymbol{\gamma}(\boldsymbol{\phi}_k), \boldsymbol{\xi}(\boldsymbol{\phi}_k))$, so storing each gradient for SAGA is not considerably more memory intensive than the Baum-Welch algorithm itself. Alternatively, SVRG can be more computationally expensive than SAGA partially because it requires periodically recalculating the full gradient approximation $\widehat{\nabla} F^{(0)}$, and this involves a full pass of the underlying dataset. However, the E step of the Baum-Welch algorithm also involves a full pass of the dataset, so using SVRG is not considerably more computationally expensive than the Baum-Welch algorithm itself. In this way, using either SAGA or SVRG in the M step adds minimal computational and memory complexity to the Baum-Welch algorithm.

3.2. Partial E Step within the M Step

Variance-reduced stochastic optimization reduces the computational cost of the M step, but the E step itself still has a time complexity of $\mathcal{O}(T)$, which can be prohibitive for large T. To decrease this computational burden, Neal and Hinton (1998) justify a partial E step within the EM algorithm for general latent variable models. However, they assume that the optimization of the M step has a closed-form solution. We use their method as inspiration and add a partial E step to the stochastic M step of the Baum-Welch algorithm.

Consider running one iteration of our version of the Baum-Welch algorithm with an initial parameter estimate $\boldsymbol{\phi}^{(0)}$. The E step involves calculating the conditional probabilities $\boldsymbol{\gamma}(\boldsymbol{\phi}^{(0)})$ and $\boldsymbol{\xi}(\boldsymbol{\phi}^{(0)})$, and the M step involves running variance-reduced stochastic optimization with loss function $F(\cdot \mid \boldsymbol{\gamma}(\boldsymbol{\phi}^{(0)}), \boldsymbol{\xi}(\boldsymbol{\phi}^{(0)}))$ and initial parameter value $\boldsymbol{\phi}^{(0)}$. Now, suppose $\boldsymbol{\phi}^{(m)}$ is to be updated using a gradient estimate using a random observation index t_m. The function $F_{t_m}(\cdot \mid \boldsymbol{\gamma}_{t_m}(\boldsymbol{\phi}^{(0)}), \boldsymbol{\xi}_{t_m}(\boldsymbol{\phi}^{(0)}))$ depends on $\boldsymbol{\gamma}_{t_m}(\boldsymbol{\phi}^{(0)})$ and $\boldsymbol{\xi}_{t_m}(\boldsymbol{\phi}^{(0)})$, each of which are vectors of conditional probabilities given $\boldsymbol{\phi}^{(0)}$. However, $\boldsymbol{\phi}^{(0)}$ is an out-of-date parameter estimate since the current parameter estimate is $\boldsymbol{\phi}^{(m)}$. Therefore, it is natural to update $\boldsymbol{\gamma}_{t_m}$ and $\boldsymbol{\xi}_{t_m}$ and redefine $F_{t_m}(\cdot \mid \boldsymbol{\gamma}_{t_m}, \boldsymbol{\xi}_{t_m})$ before calculating $\boldsymbol{\phi}^{(m+1)}$.

A naive method would be to calculate the new conditional probabilities $\boldsymbol{\gamma}_{t_m}(\boldsymbol{\phi}^{(m)})$ and $\boldsymbol{\xi}_{t_m}(\boldsymbol{\phi}^{(m)})$ and then update F_{t_m} as $F_{t_m}(\cdot \mid \boldsymbol{\gamma}_{t_m}(\boldsymbol{\phi}^{(m)}), \boldsymbol{\xi}_{t_m}(\boldsymbol{\phi}^{(m)}))$. This would ensure that $\boldsymbol{\gamma}_{t_m}$ and $\boldsymbol{\xi}_{t_m}$ are completely up-to-date, but evaluating $\boldsymbol{\gamma}_{t_m}(\boldsymbol{\phi}^{(m)})$ and $\boldsymbol{\xi}_{t_m}(\boldsymbol{\phi}^{(m)})$ takes $\mathcal{O}(TN^2)$ time and requires a full E step. Instead, our goal is to update $\boldsymbol{\gamma}_{t_m}$ and $\boldsymbol{\xi}_{t_m}$ in a way that does not scale with T.

To this end, we define the mappings $\widetilde{\boldsymbol{\alpha}}_t$, $\widetilde{\boldsymbol{\beta}}_t$, $\widetilde{\boldsymbol{\gamma}}_t$, and $\widetilde{\boldsymbol{\xi}}_t$ for $t = 1, \ldots, T$ similarly to (5)–(9). If $\mathbf{a} \in \mathbb{R}^N$ and $\mathbf{b} \in \mathbb{R}^N$ are generic row vectors, then:

$$\widetilde{\boldsymbol{\alpha}}_1(\mathbf{a}, \boldsymbol{\phi}) = \boldsymbol{\delta}\, P(y_1; \boldsymbol{\theta}), \qquad \widetilde{\boldsymbol{\alpha}}_t(\mathbf{a}, \boldsymbol{\phi}) = \mathbf{a}\, \boldsymbol{\Gamma}\, P(y_t; \boldsymbol{\theta}), \quad t = 2, \ldots, T, \tag{20}$$

$$\widetilde{\boldsymbol{\beta}}_T^{\top}(\mathbf{b}, \boldsymbol{\phi}) = \mathbf{1}_N^{\top}, \qquad \widetilde{\boldsymbol{\beta}}_t^{\top}(\mathbf{b}, \boldsymbol{\phi}) = \boldsymbol{\Gamma}\, P(y_{t+1}; \boldsymbol{\theta})\, \mathbf{b}^{\top}, \quad t = 1, \ldots, T-1, \tag{21}$$

$$\widetilde{\boldsymbol{\gamma}}_t(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}\, \operatorname{diag}(\mathbf{b})}{\mathbf{a}\, \mathbf{b}^T}, \quad t = 1, \ldots, T, \tag{22}$$

$$\widetilde{\boldsymbol{\xi}}_t(\mathbf{a}, \mathbf{b}, \boldsymbol{\phi}) = \frac{\operatorname{diag}(\mathbf{a})\, \boldsymbol{\Gamma}\, P(y_t; \boldsymbol{\theta})\, \operatorname{diag}(\mathbf{b})}{\mathbf{a}\, \boldsymbol{\Gamma}\, P(y_t; \boldsymbol{\theta})\, \mathbf{b}^{\top}}, \quad t = 2, \ldots, T, \tag{23}$$

all of which take $\mathcal{O}(N^2)$ time to compute. We have designed these mappings so that $\widetilde{\boldsymbol{\alpha}}_1(\mathbf{a}, \boldsymbol{\phi}) = \boldsymbol{\alpha}_1(\boldsymbol{\phi})$, $\widetilde{\boldsymbol{\alpha}}_t(\boldsymbol{\alpha}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi}) = \boldsymbol{\alpha}_t(\boldsymbol{\phi})$, $\widetilde{\boldsymbol{\beta}}_T^{\top}(\mathbf{b}, \boldsymbol{\phi}) = \boldsymbol{\beta}_T^{\top}(\boldsymbol{\phi})$, and $\widetilde{\boldsymbol{\beta}}_t^{\top}(\boldsymbol{\beta}_{t+1}(\boldsymbol{\phi}), \boldsymbol{\phi}) = \boldsymbol{\beta}_t^{\top}(\boldsymbol{\phi})$.

At the beginning of the M step we define conditional probability *approximations* $\widehat{\boldsymbol{\alpha}}_t^{(0)} = \boldsymbol{\alpha}_t(\boldsymbol{\phi}^{(0)})$, $\widehat{\boldsymbol{\beta}}_t^{(0)} = \boldsymbol{\beta}_t(\boldsymbol{\phi}^{(0)})$, $\widehat{\boldsymbol{\gamma}}_t^{(0)} = \boldsymbol{\gamma}_t(\boldsymbol{\phi}^{(0)})$, and $\widehat{\boldsymbol{\xi}}_t^{(0)} = \boldsymbol{\xi}_t(\boldsymbol{\phi}^{(0)})$ for $t = 1, \ldots, T$. This is simply the E step of the Baum-Welch algorithm and takes $\mathcal{O}(TN^2)$ time to compute. Then, at any given step m of the stochastic M step, we update F_{t_m} by first updating $\widehat{\boldsymbol{\alpha}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\alpha}}_{t_m}\left(\widehat{\boldsymbol{\alpha}}_{t_m-1}^{(m)}, \boldsymbol{\phi}^{(m)}\right)$ and $\widehat{\boldsymbol{\beta}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\beta}}_{t_m}\left(\widehat{\boldsymbol{\beta}}_{t_m+1}^{(m)}, \boldsymbol{\phi}^{(m)}\right)$, followed by $\widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\gamma}}_{t_m}\left(\widehat{\boldsymbol{\alpha}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\beta}}_{t_m}^{(m+1)}\right)$ and $\widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\xi}}_{t_m}\left(\widehat{\boldsymbol{\alpha}}_{t_m-1}^{(m+1)}, \widehat{\boldsymbol{\beta}}_{t_m}^{(m+1)}, \boldsymbol{\phi}^{(m)}\right)$. Finally, the loss function at index t_m can be defined as $F_{t_m}\left(\cdot \,\middle|\, \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}\right)$. Updating F_{t_m} in

Algorithm 4 VRSO-PE($\{\widehat{\boldsymbol{\alpha}}_t^{(0)}, \widehat{\boldsymbol{\beta}}_t^{(0)}, \widehat{\boldsymbol{\gamma}}_t^{(0)}, \widehat{\boldsymbol{\xi}}_t^{(0)}\}_{t=1}^T, \boldsymbol{\phi}^{(0)}, \lambda, A, P, M$)

Require: Initial conditional probability approximations $\{\widehat{\boldsymbol{\alpha}}^{(0)}, \widehat{\boldsymbol{\beta}}^{(0)}, \widehat{\boldsymbol{\gamma}}^{(0)}, \widehat{\boldsymbol{\xi}}^{(0)}\}$, initial parameters $\boldsymbol{\phi}^{(0)}$, step size λ, algorithm $A \in \{\text{SVRG, SAGA}\}$, whether to do a partial-E step $P \in \{\texttt{True}, \texttt{False}\}$, and number of iterations M.

for $t = 1, \ldots, T$ **do** ▷ initialize gradients
  $\widehat{\nabla} F_t^{(0)} = \nabla F_t\left(\boldsymbol{\phi}^{(0)} \,\middle|\, \widehat{\boldsymbol{\gamma}}_t^{(0)}, \widehat{\boldsymbol{\xi}}_t^{(0)}\right)$
end for
$\widehat{\nabla} F^{(0)} = (1/T)\sum_{t=1}^T \widehat{\nabla} F_t^{(0)}$
for $m = 0, \ldots, M-1$ **do**:
  Pick $t_m \in \{1, \ldots, T\}$ uniformly at random.
  $\left\{\widehat{\boldsymbol{\alpha}}_t^{(m+1)}, \widehat{\boldsymbol{\beta}}_t^{(m+1)}, \widehat{\boldsymbol{\gamma}}_t^{(m+1)}, \widehat{\boldsymbol{\xi}}_t^{(m+1)}\right\} = \left\{\widehat{\boldsymbol{\alpha}}_t^{(m)}, \widehat{\boldsymbol{\beta}}_t^{(m)}, \widehat{\boldsymbol{\gamma}}_t^{(m)}, \widehat{\boldsymbol{\xi}}_t^{(m)}\right\}$ for $t = 1, \ldots, T$.
  if $P = \texttt{True}$ **then** ▷ partial E step
    $\widehat{\boldsymbol{\alpha}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\alpha}}_{t_m}\left(\widehat{\boldsymbol{\alpha}}_{t_m-1}^{(m)}, \boldsymbol{\phi}^{(m)}\right)$, $\widehat{\boldsymbol{\beta}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\beta}}_{t_m}\left(\widehat{\boldsymbol{\beta}}_{t_m+1}^{(m)}, \boldsymbol{\phi}^{(m)}\right)$
    $\widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\gamma}}_{t_m}\left(\widehat{\boldsymbol{\alpha}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\beta}}_{t_m}^{(m+1)}\right)$, $\widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)} = \widetilde{\boldsymbol{\xi}}_{t_m}\left(\widehat{\boldsymbol{\alpha}}_{t_m-1}^{(m+1)}, \widehat{\boldsymbol{\beta}}_{t_m}^{(m+1)}, \boldsymbol{\phi}^{(m)}\right)$
  end if
  ▷ update parameters

$$\boldsymbol{\phi}^{(m+1)} = \boldsymbol{\phi}^{(m)} - \lambda\left[\nabla F_{t_m}\left(\boldsymbol{\phi}^{(m)} \,\middle|\, \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}\right) - \widehat{\nabla} F_{t_m}^{(m)} + \widehat{\nabla} F^{(m)}\right] \tag{24}$$

  $\widehat{\nabla} F_t^{(m+1)} = \widehat{\nabla} F_t^{(m)}$ for $t = 1, \ldots, T$ ▷ update gradients
  if A = SAGA **then**:

$$\widehat{\nabla} F_{t_m}^{(m+1)} = \nabla F_{t_m}\left(\boldsymbol{\phi}^{(m)} \,\middle|\, \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}\right), \tag{25}$$

$$\widehat{\nabla} F^{(m+1)} = \widehat{\nabla} F^{(m)} + \frac{1}{T}\left(\widehat{\nabla} F_{t_m}^{(m+1)} - \widehat{\nabla} F_{t_m}^{(m)}\right). \tag{26}$$

  end if
end for
return $\boldsymbol{\phi}^{(M)}$

this way take a total of $\mathcal{O}(N^2)$ time, which accomplishes a parameter update step that does not scale with T. Algorithm 4 outlines the M step of the Baum-Welch algorithm with a partial E step integrated in.

The partial E step detailed above is closely related to belief propagation, an algorithm that calculates conditional probabilities of variables within graphical models (Pearl 1982). In fact, the E step of the Baum-Welch algorithm is a specific instance of belief propagation, where (20)–(23) above correspond to "passing messages" within the graphical model. Belief propagation can only perform exact inference on acyclic graphical models (including HMMs), but a generalization called loopy belief propagation can perform approximate inference on general graphical models (Pearl 1988). Practitioners balance approximation error and computational complexity to decide how long to run loopy belief propagation. Likewise, we consider computational complexity when running belief propagation and evaluate

Algorithm 5 EM-VRSO($\boldsymbol{\phi}_0, \lambda, A, P, M, K$) (Version 1)

Require: Initial parameters ($\boldsymbol{\phi}_0$), step size (λ), algorithm $A \in \{\text{SVRG, SAGA}\}$, whether to do a partial E step $P \in \{\texttt{True}, \texttt{False}\}$, iterations per update (M), and number of updates (K).

for $k = 0, \ldots, K-1$ **do**
  $\{\boldsymbol{\alpha}_{k,t}, \boldsymbol{\beta}_{k,t}, \boldsymbol{\gamma}_{k,t}, \boldsymbol{\xi}_{k,t}\}_{t=1}^T = \texttt{E-step}(\boldsymbol{\phi}_k)$ ▷ E step
  $\ell \leftarrow 0$ ▷ M step
  while $\ell = 0$ or $\log p(\mathbf{y}; \boldsymbol{\phi}_{k,\ell}) < \log p(\mathbf{y}; \boldsymbol{\phi}_k)$ **do**
    $\ell \leftarrow \ell + 1$
    $\boldsymbol{\phi}_{k,\ell} =$ VRSO-PE($\{\boldsymbol{\alpha}_{k,t}, \boldsymbol{\beta}_{k,t}, \boldsymbol{\gamma}_{k,t}, \boldsymbol{\xi}_{k,t}\}_{t=1}^T, \boldsymbol{\phi}_k, \lambda, A, P, M$)
  end while
  $\boldsymbol{\phi}_{k+1} = \boldsymbol{\phi}_{k,\ell}$
end for
return $\boldsymbol{\phi}_K$

(20)–(23) only once to approximate $\boldsymbol{\gamma}_{t_m}(\boldsymbol{\phi}^{(m)})$ and $\boldsymbol{\xi}_{t_m}(\boldsymbol{\phi}^{(m)})$. This forms the basis of the partial E step within the stochastic M step of our modified Baum-Welch algorithm.

3.3. Full Algorithm

In principal, it is possible to run Algorithm 4 alone without ever performing a full E step. However, if no partial E step is used (i.e., $P = \texttt{False}$) or if SVRG is used as the optimization algorithm, then either the conditional probability approximations $\left\{\widehat{\boldsymbol{\alpha}}_t^{(m)}, \widehat{\boldsymbol{\beta}}_t^{(m)}, \widehat{\boldsymbol{\gamma}}_t^{(m)}, \widehat{\boldsymbol{\xi}}_t^{(m)}\right\}_{t=1}^T$ or the gradient approximations $\left\{\widehat{\nabla} F_t^{(m)}\right\}_{t=1}^T$ will not be updated and become out-of-date. To avoid this issue, Algorithm 5 combines the M step defined in Algorithm 4 with a full E step to complete our new Baum-Welch algorithm for HMMs.

There are two versions of EM-VRSO. Version 1 (Algorithm 5), requires the likelihood to not decrease (i.e., $\log p(\mathbf{y}; \boldsymbol{\phi}_{k,\ell}) \geq \log p(\mathbf{y}; \boldsymbol{\phi}_k)$) in order to exit the while loop of the M step. Version 2 (Algorithm 6) requires the likelihood to *strictly* increase by some threshold to exit the while loop of the M step. We use version 2 to prove theoretical results, but the strict threshold relies on values that are usually not known in practice. Therefore, we use version 1 in our simulation and case studies and defer version 2 to the online appendix. Our simulation and case studies show that version 1 of EM-VRSO converges to local maxima of the log-likelihood function in practice.

At first, it seems troubling to require $\log p(\mathbf{y}; \boldsymbol{\phi}_{k,\ell}) \geq \log p(\mathbf{y}; \boldsymbol{\phi}_k)$ to exit the while loop of EM-VRSO, since this requirement may cause an infinite loop if it cannot be met. In practice, we found that the log-likelihood increased after a pass through the M step in most cases unless the step size λ was very large. As such, we detail how to adaptively select λ in Section 4.4. For a theoretical justification that the while loop terminates, denote $\ell^*(k)$ as the (random) number of runs through the inner loop of EM-VRSO for iteration k (i.e., the maximum value obtained by ℓ for a given value of k). We prove in Theorem 1 that $\mathbb{P}(\ell^*(k) < \infty) = 1$. One final concern is whether the sequence $\{\boldsymbol{\phi}_k\}_{k=0}^\infty$ generated by EM-VRSO converges to a local maximum

of the likelihood as $K \to \infty$. Theorem 1 guarantees such convergence under standard regularity conditions. We prove Theorem 1 in the online appendix.

Theorem 1. Suppose the following conditions are met in Algorithm 6 with $P =$ `False` and $A =$ SVRG:

1. The parameters $\boldsymbol{\phi}$ lie in $\boldsymbol{\Phi} = \mathbb{R}^r$ for some dimension r.
2. $\boldsymbol{\Phi}_{\boldsymbol{\phi}_0} = \{\boldsymbol{\phi} \in \boldsymbol{\Phi} : \log p(\mathbf{y}; \boldsymbol{\phi}) \geq \log p(\mathbf{y}; \boldsymbol{\phi}_0)\}$ is compact for all $\boldsymbol{\phi}_0$ if $\log p(\mathbf{y}; \boldsymbol{\phi}_0) > -\infty$.
3. $\log p(\mathbf{y}; \boldsymbol{\phi})$ is differentiable in $\boldsymbol{\phi}$ for all $\boldsymbol{\phi} \in \boldsymbol{\Phi}$.
4. $F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}'))$ is convex with respect to $\boldsymbol{\phi}$ and $F(\boldsymbol{\phi} \mid \boldsymbol{\gamma}(\boldsymbol{\phi}'), \boldsymbol{\xi}(\boldsymbol{\phi}'))$ is strongly convex with respect to $\boldsymbol{\phi}$ for all $\boldsymbol{\phi}'$ with constant $C > 0$. Namely, for all $\boldsymbol{\phi}$, $\boldsymbol{\phi}_0$ and $\boldsymbol{\phi}'$:

$$F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}')) \geq F_t(\boldsymbol{\phi}_0 \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}')) + \nabla F_t(\boldsymbol{\phi}_0 \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}'))^T(\boldsymbol{\phi} - \boldsymbol{\phi}_0), \tag{27}$$

$$F(\boldsymbol{\phi} \mid \boldsymbol{\gamma}(\boldsymbol{\phi}'), \boldsymbol{\xi}(\boldsymbol{\phi}')) \geq F(\boldsymbol{\phi}_0 \mid \boldsymbol{\gamma}(\boldsymbol{\phi}'), \boldsymbol{\xi}(\boldsymbol{\phi}')) + \nabla F(\boldsymbol{\phi}_0 \mid \boldsymbol{\gamma}(\boldsymbol{\phi}'), \boldsymbol{\xi}(\boldsymbol{\phi}'))^T(\boldsymbol{\phi} - \boldsymbol{\phi}_0) + \frac{C}{2}\|\boldsymbol{\phi} - \boldsymbol{\phi}_0\|_2^2. \tag{28}$$

5. $F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}'))$ is uniformly Lipschitz-smooth with respect to $\boldsymbol{\phi}$ for all t and $\boldsymbol{\phi}'$ with constant $L \geq C > 0$. Namely, for all t, $\boldsymbol{\phi}$, $\boldsymbol{\phi}_0$ and $\boldsymbol{\phi}'$:

$$F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}')) \leq F_t(\boldsymbol{\phi}_0 \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}')) + \nabla F_t(\boldsymbol{\phi}_0 \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}'))^T(\boldsymbol{\phi} - \boldsymbol{\phi}_0) + \frac{L}{2}\|\boldsymbol{\phi} - \boldsymbol{\phi}_0\|_2^2. \tag{29}$$

6. The step size λ is sufficiently small and M is sufficiently large such that

$$\zeta = \frac{1}{C\lambda(1 - 2L\lambda)M} + \frac{2L\lambda}{1 - 2L\lambda} < 1. \tag{30}$$

7. $\nabla F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t(\boldsymbol{\phi}'), \boldsymbol{\xi}_t(\boldsymbol{\phi}'))$ is uniformly continuous in $(\boldsymbol{\phi}, \boldsymbol{\phi}')$ for all t.

Further, let $\mathcal{S} = \{\ell^*(k) < \infty \text{ for all } k\}$. Then, $\mathbb{P}\{\mathcal{S}\} = 1$, and the following holds on $\mathcal{S}$: all limit points of $\{\boldsymbol{\phi}_k\}_{k=0}^{\infty}$ are stationary points of $\log p(\mathbf{y}; \cdot)$, and $\log p(\mathbf{y}; \boldsymbol{\phi}_k)$ converges monotonically to $\log p^* = \log p(\mathbf{y}; \boldsymbol{\phi}^*)$ for some stationary point of $\log p(\mathbf{y}; \cdot)$, $\boldsymbol{\phi}^*$.

Conditions 1–3 are from Wu (1983) and are standard assumptions needed to prove the convergence of the EM algorithm. Likewise, Conditions 4–6 are from Johnson and Zhang (2013) and are standard assumptions used to prove common properties of stochastic optimization algorithms. Condition 7 is needed to prove that SVRG is a continuous mapping.

Condition 2 from Wu (1983) and condition 5 of Johnson and Zhang (2013) can be restrictive and are often violated in common settings. For example, both are violated when estimating the variance of normal state-dependent distributions within an HMM. This issue is well-known for maximum likelihood estimation in mixture models (Chen and Li 2009; Liu, Wu, and Meeker 2015). It can be avoided by setting lower bounds on the variance components (Zucchini, Macdonald, and Langrock 2016).

Condition 4 seems concerning at first because the log-likelihood of a hidden Markov model is usually multi-modal and non-convex. However, the convexity condition applies to F_t rather than the log-likelihood itself. In addition, F_t is a linear combination of probability densities, which are often convex in $\boldsymbol{\theta}$ (Boyd and Vandenberghe 2004), and log-sum-exp functions, which are convex in $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$. Even if F_t is not convex, stochastic gradient methods can escape local optima within the M step more effectively than standard gradient descent (Kleinberg, Li, and Yuan 2018).

Theorem 1 guarantees convergence only to a local optimum of the likelihood for version 2 of `EM-VRSO` with $P =$ `False` and $A =$ SVRG. However, some studies suggest that EM algorithms like `EM-VRSO` may escape local optima of the likelihood faster than direct numerical optimization of the log-likelihood (Zhang, Poupart, and Trimponias 2020).

Unfortunately, the convergence analysis for `EM-VRSO` becomes more complex when $P =$ `True`, as the E and M steps are intertwined. Nonetheless, Theorem 2 below demonstrates that stationary points of the log-likelihood function serve as fixed points of Algorithm 5 for all values of P and A. The proof of Theorem 2 is provided in the online appendix.

Theorem 2. If $\nabla \log p(\mathbf{y}; \boldsymbol{\phi}_0) = 0$, then for all $\lambda \in \mathbb{R}$, $A \in \{\text{SAGA}, \text{SVRG}\}$, $P \in \{\texttt{True}, \texttt{False}\}$, $M \in \mathbb{N}$, and $K \in \mathbb{N}$, $\texttt{EM-VRSO}(\boldsymbol{\phi}_0, \lambda, A, P, M, K) = \boldsymbol{\phi}_0$ with probability 1, where `EM-VRSO` is defined in Algorithm 5.

Theorem 2 is useful because it guarantees that Algorithm 5 does not change $\boldsymbol{\phi}_0$ when it is a stationary point of the likelihood. However, it makes no guarantees that Algorithm 5 will converge if $\nabla \log p(\mathbf{y}; \boldsymbol{\phi}_0) \neq 0$ and either $P =$ `True` or $A =$ SAGA. Nonetheless, we see in our simulation and case studies that Algorithm 5 approaches local maxima of the log-likelihood function faster than existing full-batch baselines for all values of P and A. For theoretical guarantees on convergence, practitioners can set $P =$ `True` or $A =$ SAGA for a predetermined number of iterations, followed by switching to $P =$ `False` and $A =$ SVRG, or a full-gradient method such as BFGS (Fletcher 2000).

4. Practical Considerations

Algorithm 5 outlines a method to perform the Baum-Welch algorithm with a stochastic E and M step. This section outlines implementation details that improve its practical performance.

4.1. Line Search for Step Size Selection

One drawback of stochastic optimization is that the step size can greatly affect the practical performance of the algorithm. Defazio, Bach, and Lacoste-Julien (2014) suggest a step size of $\lambda = 1/(3L)$ for SAGA, where L is the Lipschitz constant defined in Theorem 1. However, the Lipschitz constants are rarely known in practice. Therefore, following Schmidt, Le Roux, and Bach (2017), we initialize an estimate of the Lipschitz constant, $\hat{L}$, and update it if the following inequality does not hold at any step m of the optimization algorithm:

$$F_{t_m}^{(m+1)}\left(\boldsymbol{\phi}^{(m)} - \frac{1}{\hat{L}}\nabla F_{t_m}^{(m+1)}(\boldsymbol{\phi}^{(m)})\right) \leq F_{t_m}^{(m+1)}\left(\boldsymbol{\phi}^{(m)}\right) - \frac{1}{2\hat{L}}\left\|\nabla F_{t_m}^{(m+1)}(\boldsymbol{\phi}^{(m)})\right\|^2, \quad \text{where}$$
$$F_{t_m}^{(m+1)} = F_{t_m}\left(\cdot \,\middle|\, \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}\right). \tag{31}$$

The inequality above is obeyed if $\hat{L} = L$, so if it is violated, then we double the Lipschitz constant estimate $\hat{L}$. We also follow Schmidt, Le Roux, and Bach (2017) and do not check the inequality if $\|\nabla F_{t_m}^{(m+1)}(\boldsymbol{\phi}^{(m)})\|^2 < 10^{-8}$ due to numerical instability.

In addition, the Lipschitz constant L is a global quantity, but the algorithm will likely remain in a neighborhood around a local maximum of F later in the optimization algorithm. Within this local neighborhood, a smaller value of L may apply (Schmidt, Le Roux, and Bach 2017), so the Lipschitz constant estimate is decreased by a small amount after each parameter update: $\hat{L} \leftarrow 2^{-1/T}\,\hat{L}$. Updating $\hat{L}$ after each parameter update allows the step size of the optimization algorithm to adapt to the smoothness of the objective function close to the optimum value.

4.2. Multiple Step Sizes

The optimization problem within the M step of the Baum-Welch algorithm can be written as separate optimization problems over $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\nu}$ (recall that $\boldsymbol{\phi} = \{\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\nu}\}$). In particular, we can rewrite F_t as $F_t(\boldsymbol{\phi} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t) = G_t(\boldsymbol{\theta} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t) + H_t(\boldsymbol{\eta}, \boldsymbol{\nu} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t)$, where

$$G_t(\boldsymbol{\theta} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t) = -\sum_{i=1}^{N} \gamma_t^{(i)} \log f^{(i)}(y_t; \theta^{(i)}), \quad t = 1, \ldots, T, \tag{32}$$

$$H_1(\boldsymbol{\eta}, \boldsymbol{\nu} \mid \boldsymbol{\gamma}_1, \boldsymbol{\xi}_1) = -\sum_{i=1}^{N} \gamma_1^{(i)} \log \delta^{(i)}(\boldsymbol{\nu}), \tag{33}$$

$$H_t(\boldsymbol{\eta}, \boldsymbol{\nu} \mid \boldsymbol{\gamma}_t, \boldsymbol{\xi}_t) = -\sum_{i=1}^{N}\sum_{j=1}^{N} \xi_t^{(i,j)} \log \Gamma^{(i,j)}(\boldsymbol{\eta}), \quad t = 2, \ldots, T. \tag{34}$$

To this end, let $\widehat{\nabla} G_t^{(m)}$ be equal to the components of $\widehat{\nabla} F_t^{(m)}$ that correspond to $\boldsymbol{\theta}$, and let $\widehat{\nabla} H_t^{(m)}$ be equal to the components of $\widehat{\nabla} F_t^{(m)}$ that correspond to $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$. Likewise, let $\widehat{\nabla} G^{(m)}$ be equal to the components of $\widehat{\nabla} F^{(m)}$ that correspond to $\boldsymbol{\theta}$, and let $\widehat{\nabla} H^{(m)}$ be equal to the components of $\widehat{\nabla} F^{(m)}$ that correspond to $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$. Then, we can rewrite the gradient step in the stochastic M step of Algorithm 4 as

$$\boldsymbol{\theta}^{(m+1)} = \boldsymbol{\theta}^{(m)} - \lambda_{\boldsymbol{\theta}}\left[\nabla G_{t_m}\left(\boldsymbol{\theta}^{(m)} \,\middle|\, \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}\right) - \widehat{\nabla} G_{t_m}^{(m)} + \widehat{\nabla} G^{(m)}\right], \tag{35}$$

$$\{\boldsymbol{\eta}, \boldsymbol{\nu}\}^{(m+1)} = \{\boldsymbol{\eta}, \boldsymbol{\nu}\}^{(m)} - \lambda_{\boldsymbol{\eta},\boldsymbol{\nu}}\left[\nabla H_{t_m}\left(\boldsymbol{\eta}^{(m)}, \boldsymbol{\nu}^{(m)} \,\middle|\, \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}\right) - \widehat{\nabla} H_{t_m}^{(m)} + \widehat{\nabla} H^{(m)}\right]. \tag{36}$$

where $\lambda_{\boldsymbol{\theta}} = \lambda_{\boldsymbol{\eta},\boldsymbol{\nu}} = \lambda$. Note that G_t is a function of only $\boldsymbol{\theta}$ and H_t is a function of only $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ for given $\widehat{\boldsymbol{\gamma}}_t$ and $\widehat{\boldsymbol{\xi}}_t$ and all $t = 1, \ldots, T$. As such, we allow $\lambda_{\boldsymbol{\theta}} \neq \lambda_{\boldsymbol{\eta},\boldsymbol{\nu}}$ and have each depend upon different Lipschitz constant estimates: $\lambda_{\boldsymbol{\theta}} = 1/(3\hat{L}_G)$ and $\lambda_{\boldsymbol{\eta},\boldsymbol{\nu}} = 1/(3\hat{L}_H)$. The line search described in Section 4.1 was then used to update the estimates $\hat{L}_G$ and $\hat{L}_H$ separately.

4.3. Adaptive Step Size for Fixed Lipschitz Constants

Under certain regularity conditions, Defazio, Bach, and Lacoste-Julien (2014) prove that SAGA converges using a step size of $1/(3L)$ for a given loss function F with Lipschitz constant L. We therefore initialize step sizes of $\lambda_{\boldsymbol{\theta}} = 1/(3\hat{L}_G)$ and $\lambda_{\boldsymbol{\eta},\boldsymbol{\nu}} = 1/(3\hat{L}_H)$ for all experiments and algorithms. However, for Algorithm 5 with $P =$ `True`, the objective function $F(\cdot \mid \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}) = G(\cdot \mid \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}) + H(\cdot \mid \widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}, \widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)})$ itself changes over the course of a single M step as $\widehat{\boldsymbol{\gamma}}_{t_m}^{(m+1)}$ and $\widehat{\boldsymbol{\xi}}_{t_m}^{(m+1)}$ are updated. As a result, more conservative (i.e., smaller) step sizes $\lambda_{\boldsymbol{\theta}}$ and $\lambda_{\boldsymbol{\eta},\boldsymbol{\nu}}$ may be needed, even if the Lipschitz constant estimates $\hat{L}_G$ and $\hat{L}_H$ are accurate. We therefore allow Algorithm 5 to change the step size after each attempt ℓ through the M step of the Baum-Welch algorithm. Namely, if the log-likelihood does not increase after iteration k and attempt ℓ through the M step of Algorithm 5 with $P =$ `True`, we halve the step size (as a function of either $\hat{L}_G$ or $\hat{L}_H$) for attempt $\ell+1$. For example, if the step sizes are $\lambda_{\boldsymbol{\theta}} = 1/(3\hat{L}_G)$ and $\lambda_{\boldsymbol{\eta},\boldsymbol{\nu}} = 1/(3\hat{L}_H)$ for attempt ℓ through a given M step, and attempt ℓ results in a decrease of the log-likelihood, we define new step-sizes $\lambda_{\boldsymbol{\theta}} \leftarrow 1/(6\hat{L}_G)$ and $\lambda_{\boldsymbol{\eta},\boldsymbol{\nu}} \leftarrow 1/(6\hat{L}_H)$ for attempt $\ell + 1$. We also maintain this halved step size for the remainder of Algorithm 5.

4.4. Sampling for SAGA and SVRG Without Replacement

Finally, we sample each random index t *without* replacement within Algorithm 4. If $M > T$, then we sample without replacement until all time indices are sampled, and then resample the dataset without replacement. Sampling without replacement for SGD is often easier to implement and performs better than sampling with replacement (Gürbüzbalaban, Ozdaglar, and Parrilo 2021). Shamir (2016) also gives several convergence results for SVRG when indices are sampled without replacement.

5. Simulation Study

5.1. Simulation Procedure

To test the performance of Algorithm 5, we ran eight simulation experiments. For a given experiment, we simulated $T \in \{10^3, 10^5\}$ observations from an HMM with $N \in \{3, 6\}$ hidden states and observations $y_t \in \mathbb{R}^d$, with $d \in \{3, 6\}$. All possible combinations of T, N, and d comprised a total of $2^3 = 8$ experiments. For each experiment, we simulated five datasets. For every experiment and dataset, $Y_t \mid X_t = i$ followed a normal distribution with mean $\mu^{(i)}$ and covariance matrix $\boldsymbol{\Sigma}^{(i)}$. We defined $\mu^{(i)}$ and $\boldsymbol{\Sigma}^{(i)}$ for every dataset as $\mu^{(i)} \sim \mathcal{N}(0, I)$ and $\boldsymbol{\Sigma}^{(i)} = \text{diag}[\exp(-2)]$ for $i \in \{1, \ldots, N\}$ where I is the

identity matrix. This simulation procedure resulted in relatively well separated state-dependent distributions. For example, when $N = 3$, the Euclidean distance between any pair of means within any of the five datasets ranged between 1.0 and 4.0 for $d = 3$ and between 2.0 and 4.7 for $d = 6$. We set the transition probability matrices of the generating process to depend upon T so that the expected number of total transitions was 100 for all experiments. Our purpose in keeping the expected number of transitions fixed was to induce a high degree of sequential dependence while simultaneously encouraging each hidden state to be visited in the simulated time series. This decision also simulates sequences of observations that are sampled at either low ($T = 10^3$) or high ($T = 10^5$) frequencies. Denote the true transition probability matrix from an experiment with T observations and N hidden states as $\mathbf{\Gamma}_{T,N}$. We defined $\mathbf{\Gamma}_{10^3,3} \in \mathbb{R}^{3\times 3}$ to have diagonal elements of 0.9 and off-diagonal elements of 0.05, $\mathbf{\Gamma}_{10^3,6} \in \mathbb{R}^{6\times 6}$ to have diagonal elements of 0.9 and off-diagonal elements of 0.02, $\mathbf{\Gamma}_{10^5,3} \in \mathbb{R}^{3\times 3}$ to have diagonal elements of 0.999 and off-diagonal elements of 5×10^{-4}, and $\mathbf{\Gamma}_{10^3,6} \in \mathbb{R}^{6\times 6}$ to have diagonal elements of 0.999 and off-diagonal elements of 2×10^{-4}. We randomly defined the initial distribution as $\boldsymbol{\delta} \sim \text{dir}(\mathbf{1}_N)$ for every experiment and dataset.

5.2. Optimization Procedure

We estimated the parameters of the generating model for all five datasets and all eight experiments using six different versions of Algorithm 5. In particular, we used $A \in \{\text{SVRG, SAGA}\}$, and for each value of A, we used the combinations $\{P = \texttt{False}, M = T\}$, $\{P = \texttt{True}, M = T\}$, and $\{P = \texttt{True}, M = 10T\}$. Recall that setting $P = \texttt{True}$ corresponds to integrating a partial E step into the variance-reduced stochastic M step. The variable M corresponds to the number of iterations of SAGA or SVRG that are performed at each M step of the algorithm. It is natural to set $M = T$ to approximately balance the computational load of the E step and the M step. Nonetheless, we ran an experiment with $M = 10T$ and $P = \texttt{True}$ to test the algorithm when the majority of the computational load was placed on the combined partial E / stochastic M step. As a baseline, we also estimated the HMM parameters using direct likelihood maximization via three gradient-based methods: BFGS (Fletcher 2000), the conjugate gradient method (Fletcher and Reeves 1964), and full-batch gradient descent. The model used in our simulation study has closed-form solutions to the M step of the Baum-Welch algorithm. However, we did not include the standard Baum-Welch algorithm as a baseline method because we did not want to rely on closed-form solutions to the M step. If such solutions exist in practice, we recommend using either the standard Baum-Welch algorithm or the Baum-Welch algorithm with a partial E step (Neal and Hinton 1998).

We sampled a total of five random parameter initializations for each dataset/experiment pair, and then ran all nine optimization algorithms on every parameter initialization. Each parameter initialization was reused for each algorithm to ensure consistency. Throughout the optimization procedure, we assumed that $\mathbf{\Sigma}^{(i)}$ was diagonal for all $i \in \{1, \ldots, N\}$, which is in line with the generating model described earlier. Further, we reparameterized $\mathbf{\Sigma}^{(i)}$ as $\mathbf{\Sigma}^{(i)} = \text{diag}\left[\exp(\boldsymbol{\rho}^{(i)})\right]$ and performed inference on $\boldsymbol{\rho}^{(i)} = \begin{pmatrix}\rho_1^{(i)} & \ldots & \rho_d^{(i)}\end{pmatrix}$ for $i = 1, \ldots, N$, which is unconstrained. Let $\bar{y}$ denote the sample mean and $\mathbf{Q}$ denote the sample covariance of the observation sequence $\{y_t\}_{t=1}^T$. We initialized $\boldsymbol{\theta}_0 = \{\mu_0^{(i)}, \boldsymbol{\rho}_0^{(i)}\}_{i=1}^N$ as $\mu_0^{(i)} \sim \mathcal{N}(\bar{y}, \text{diag}(\mathbf{Q}))$ and $\boldsymbol{\rho}_0^{(i)} \sim \mathcal{N}\left(\log(\text{diag}(\mathbf{Q})), 2I\right)$ for $i = 1, \ldots, N$. We initialized $\boldsymbol{\nu}_0$ as $\nu_0^{(i)} \sim \mathcal{N}(0, 1)$ for $i = 2, \ldots, N$ and we initialized $\boldsymbol{\eta}_0$ as $\eta_0^{(i,j)} \sim \mathcal{N}(-2, 2^2)$ for $i, j = 1, \ldots, N$, where $i \neq j$. All six algorithms were initialized with step sizes of $\lambda_{\boldsymbol{\theta}} = 1/3\hat{L}_G$ and $\lambda_{\boldsymbol{\eta},\boldsymbol{\nu}} = 1/3\hat{L}_H$. The Lipschitz constants were initialized as $\hat{L}_G = \hat{L}_H = 100/3$ and updated during the optimization routine according to the procedure from Section 4.1. All algorithms and baselines were run for a total of 12 hr on the Compute Canada Cedar cluster on nodes with 16GB of RAM. All baselines were implemented using the Scipy library in Python (Virtanen et al. 2019), and we implemented Algorithm 5 using a custom Python script.

We employed several measures to fairly compare the performance of each optimization algorithm. To account for differences in speed due to implementation discrepancies, we measured computational complexity in epochs in addition to raw computation time. We define one epoch as either T evaluations of (20)–(23) in the E step of Algorithm 5, T stochastic gradient evaluations in the M step of Algorithm 5, or one gradient evaluation in the full-gradient baseline algorithms. We estimated the true maximum likelihood parameters $\boldsymbol{\phi}^*$ for each dataset / experiment pair using the parameters from the best-performing optimization algorithm and initialization after 12 hr. Convergence was defined as the point when the gradient norm of the log-likelihood (divided by T) was less than 10^{-2}. The tolerance was set to 10^{-2} because it was the lowest tolerance that all algorithms regularly converged to within 12 hr. Even though each algorithm was run for a full 12 hr, we denoted the epoch and time when each algorithm either converged or finished running as the moment of "termination".

5.3. Simulation Results

The results from the simulation study are shown in Figures 2–4. We were primarily focused on the big-data setting and thus only present results from experiments with $T = 10^5$ in the main text. All figures associated with experiments for $T = 10^3$ can be found in Supplement A.

Algorithm 5 with $A = \text{SVRG}$ markedly sped up the optimization procedure, as it usually converged in at most half as many epochs compared to the baselines for each experiment. Algorithm 5 with $A = \text{SAGA}$ also tended to converge in fewer epochs than the baselines for experiments with $N = 6$, particularly when $P = \texttt{True}$ and $M = 10T$. One epoch of each baseline method took less computation time than one epoch of Algorithm 5, but Algorithm 5 with $A = \text{SVRG}$ consistently converged before the baseline methods in terms of raw computation time as well as epoch (Figure 3 and Supplement A). All methods were prone to converge to local minima of the negative log-likelihood, especially when $N = 6$, in which case the likelihood surface was highly multi-modal. However,

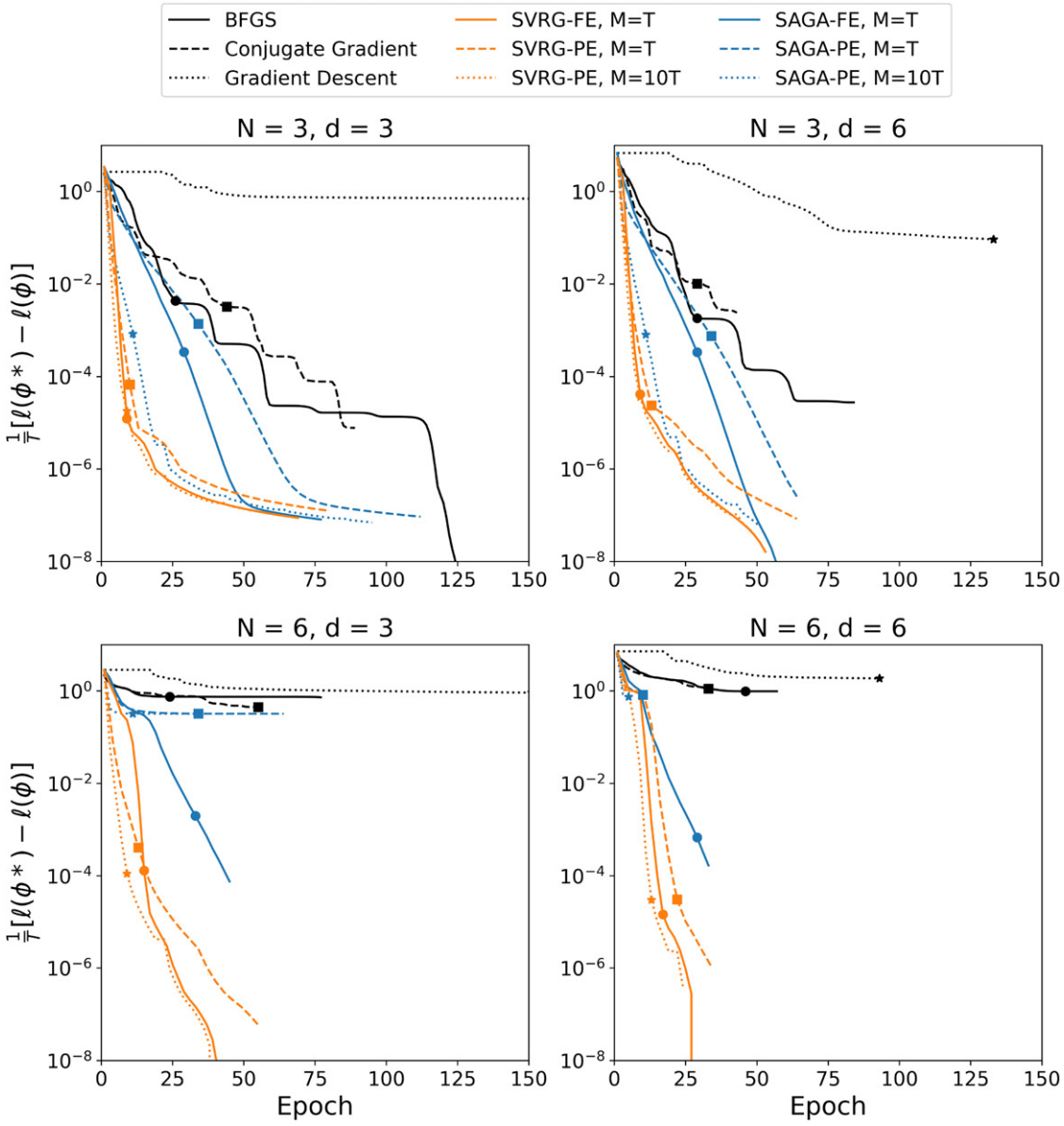

Figure 2. Maximum log-likelihood (across all algorithms and initial values after 12 hr), $\ell(\boldsymbol{\phi}^*)$, minus the log-likelihood at the given epoch, $\ell(\boldsymbol{\phi})$, divided by T for a selected run of each optimization algorithm in the simulation study. Each algorithm was run for 12 hr on a dataset generated with $T = 10^5$, $N \in \{3, 6\}$, or $d \in \{3, 6\}$. For each experiment and optimization algorithm, we display the one random initialization (of five) that resulted in the highest likelihood after 12 hr. FE corresponds to $P =$ `False`, and PE corresponds to $P =$ `True`. The y-axis is on a log-scale. Dots correspond to the epoch and likelihood at termination.

the bottom row of Figure 3 shows that in all $T = 10^5$ experiments, negative log-likelihood values at termination were lower for all versions of Algorithm 5 compared to baseline methods. In particular, each optimization algorithm / experiment pair has a corresponding boxplot which summarizes the (re-scaled) negative log-likelihood values at termination from all datasets and initializations. For all experiments, all medians and minimums corresponding to versions of Algorithm 5 were lower than the median and minimum values corresponding to BFGS. The results for experiments with $T = 10^3$ were more varied (see Figure 1 of Supplement A), which indicates that our inference method is more efficient when applied to datasets with large T. We present figures analogous to Figure 2 for all five datasets corresponding to all experiments in Supplement A.

6. Case Study

We tested the performance of our optimization algorithm by modeling the movement of eight northern resident killer whales off the coast of British Columbia, Canada. Biologgers are an essential tool used to understand the behavior of marine mammals. For example, time-depth recorders allow researchers to estimate behavioral states associated with each dive (e.g., foraging, resting, and traveling, Tennessen et al. 2023; McRae et al. 2024). Researchers also use biologging datasets to identify and characterize dive phases, which are important for inferring behavior (e.g., prey capture often occurs in the bottom phase of a foraging dive, Wright et al. 2017; Jensen et al. 2023). As such, we developed a model to identify three common dive phases (ascent, descent, and bottom) and three dive types that may indicate distinct behaviors of the animal, including resting,

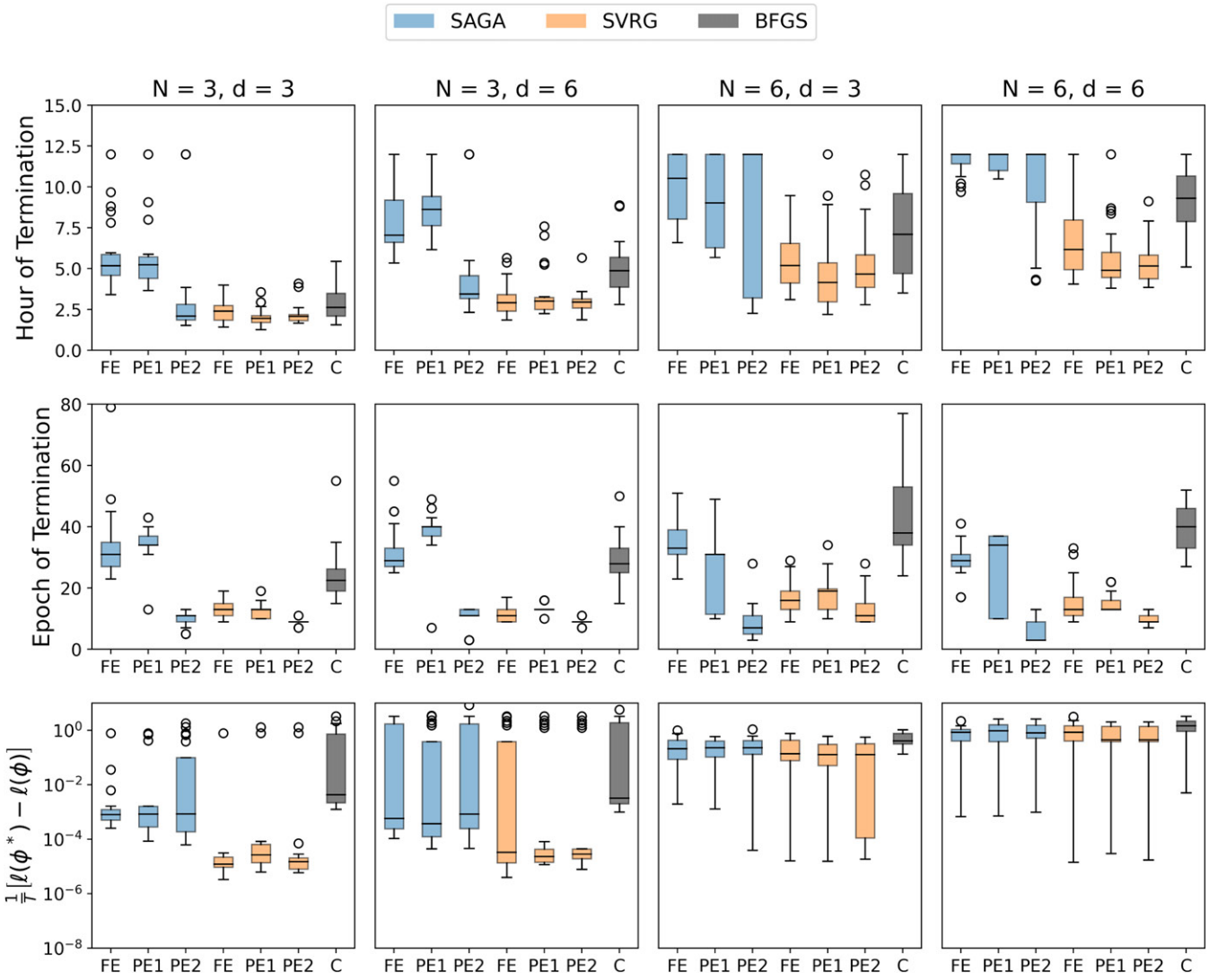

Figure 3. Boxplots showing time to terminate (top), epochs to terminate (middle), and maximum log-likelihood minus log-likelihood at termination (all over T, bottom) for each optimization algorithm in the simulation study. The log-likelihood is denoted as $\ell(\boldsymbol{\phi})$, and the maximum log-likelihood across all algorithms and initial values after 12 hr is denoted as $\ell(\boldsymbol{\phi}^*)$. Unlike Figure 2, results include all five datasets and all five parameter initializations per dataset. FE corresponds to $\{P = \texttt{False}, M = T\}$, PE1 corresponds to $\{P = \texttt{True}, M = T\}$, and PE2 corresponds to $\{P = \texttt{False}, M = 10T\}$. Results are shown for all simulation studies with $T = 10^5$. The y-axis of the bottom row is on a log-scale.

foraging, and traveling. We performed inference on the resulting model using our optimization algorithm in order to illustrate its computational advantages.

6.1. Data Collection and Preprocessing

The data used in this case study were collected in August and September of 2020 using a CATS time-depth recorder, or TDR (Customizable Animal Tracking Solutions, *www.cats.is*). Northern resident killer whales were equipped with suction-cup attached CATs tags in Queen Charlotte Sound using an adjustable 6–8m carbon fiber pole. The tags were programmed to release within 3–24 hr of attachment. Instruments were retrieved following each deployment using a Wildlife Computers 363C SPOT tag (providing Argos satellite positions), goniometer, ultra high frequency receiver, and yagi antenna. The tags included 3D kinematic sensors (accelerometer, magnetometer, gyroscope), time-depth recorder, hydrophone and camera. All sensors were programmed to sample at 50 hertz. However, for the purposes of this study, we focus on the time-depth recorder data to discern behaviorally distinct dives. We calibrated the depth readings using a MATLAB package developed by Cade et al. (2021), and defined a dive as any sequence of depth readings under 0.5 meters that lasted for at least 2 sec. We then down-sampled the depth readings to a frequency of 0.5 hertz. The processed dataset contained a total of 5858 dives and 89,462 depth readings. Figure 5 shows the depth and change in depth for a subset of dives for one whale in the dataset.

6.2. Model Formulation

Dive phases may vary depending upon the animal's behavior. For example, foraging dives tend to be deeper and longer than resting dives, so it is natural to model the phases of foraging dives differently compared to those of resting dives (Tennessen et al. 2019b). As such, we used a hierarchical model to jointly

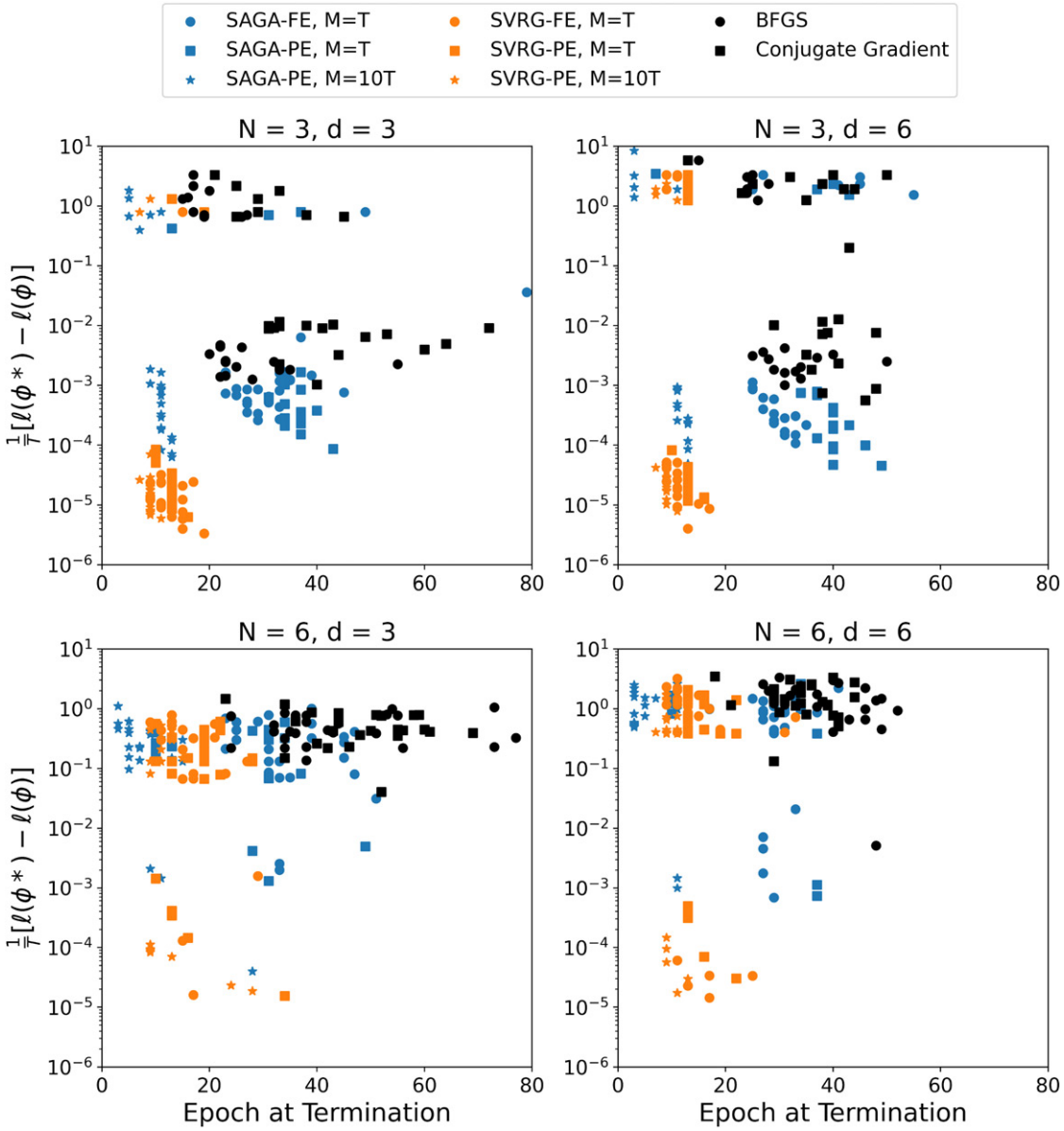

Figure 4. Maximum log-likelihood minus log-likelihood at termination (all over T) versus epochs to terminate for each optimization algorithm in the simulation study with $T = 10^5$, $N \in \{3, 6\}$, and $d \in \{3, 6\}$. The log-likelihood is denoted as $\ell(\boldsymbol{\phi})$, and the maximum log-likelihood across all algorithms and initial values after 12 hr is denoted as $\ell(\boldsymbol{\phi}^*)$. Unlike Figure 2, results include all five datasets and all five parameter initializations per dataset. FE corresponds to $P = \texttt{False}$, and PE corresponds to $P = \texttt{True}$. The y-axis is on a log-scale.

model dive types and dive phases (Leos Barajas et al. 2017; McRae et al. 2024). Hierarchical HMMs are specific instances of traditional HMMs, so the machinery developed here is applicable to perform inference. We assumed there to be three dive types, which is consistent with other studies of cetaceans (e.g., resting, foraging and traveling, Leos Barajas et al. 2017). We also assumed that there are three dive phases per dive type (descent, bottom, and ascent), which is also consistent with other studies of diving birds and mammals (e.g., Viviant, Monestiez, and Guinet 2014). This resulted in a total of $N = 9$ hidden states, each corresponding to a different dive phase / dive type combination. Rather than modeling raw depth every two seconds as the observation sequence, we encoded each two-second window of depth data with summary statistics. Namely, we denoted an observation as $Y_t = \{D_t, E_t\}$, where $D_t \in \mathbb{R}$ is the change in depth in meters and $E_t = 1$ if a dive ended at index t and $E_t = 0$ otherwise. Within dive type i and dive phase j, we assumed D_t followed a normal distribution with mean $\mu^{(i,j)}$ and standard deviation $\sigma^{(i,j)}$, and we assumed that E_t followed a Bernoulli distribution with probability $p^{(i,j)}$. We assumed that dives must end on the ascent phase, so we set $p^{(i,1)} = p^{(i,2)} = 0$ for dive types $i = 1, 2, 3$. Conditioned on the dive type and dive phase, D_t and E_t were assumed to be independent of one another.

Since each dive must begin with the descent phase, we set the initial distribution $\boldsymbol{\delta}$ to have the form $\boldsymbol{\delta} = \left(\delta^{(1)} \;\; 0 \;\; 0 \;\; \delta^{(2)} \;\; 0 \;\; 0 \;\; \delta^{(3)} \;\; 0 \;\; 0\right)$, where $\delta^{(i)}$ represents the probability that a killer whale begins its dive profile with a dive of type i. We defined the transition probability matrix at time t to depend upon the previous observation E_{t-1} because the transition matrix will naturally depend upon when a dive begins and ends. For example, the killer whale cannot change dive type mid-dive, it must begin each dive in the descent phase, and it

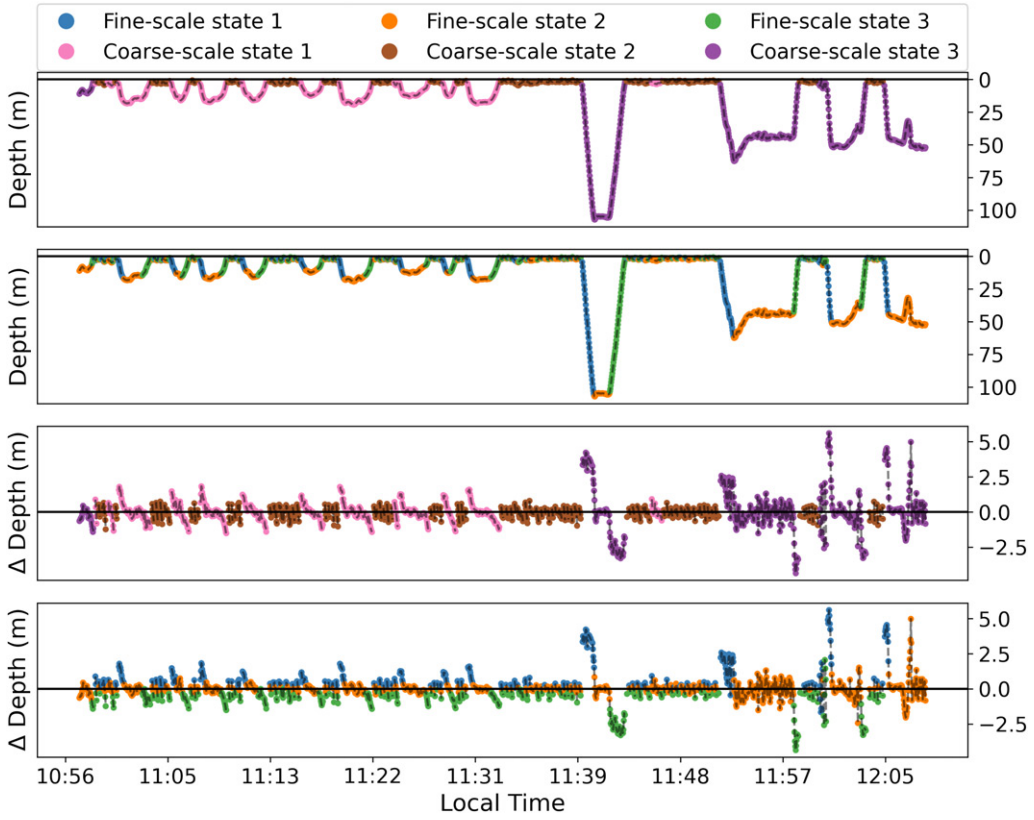

Figure 5. Depth profile and change in depth versus time of day for a selected killer whale (I107, male, born 2004) off the coast of British Columbia, Canada. The data in panels one and three are color-coded according to the most likely hidden coarse-scale state (i.e., dive type) for each dive. The data in panels two and four are color-coded according to the most likely hidden fine-scale state (i.e., dive phase) for each two-second window.

must end each dive in the ascent phase. The structure of this transition probability matrix means that our model is technically a generalization of a standard HMM (namely an HMMSDO, Li 2005). Nonetheless, Li (2005) and Tamposis et al. (2018) show that the likelihood of this model is similar to a traditional HMM, and that the standard Baum-Welch algorithm is still valid to perform inference. To this end, we defined a coarse-scale, inter-dive transition probability matrix $\mathbf{\Gamma}^{(c)} \in \mathbb{R}^{3\times 3}$ as well as a fine-scale, intra-dive probability transition matrix $\mathbf{\Gamma}^{(f,i)} \in \mathbb{R}^{3\times 3}$ for each dive type i. $\mathbf{\Gamma}^{(f,i)}$ was upper-triangular for $i \in \{1,2,3\}$ because the descent and bottom phases of a dive cannot occur after ascent, and the descent phase of a dive cannot occur after the bottom phase. Formally, the transition matrix was a function of E_{t-1} and defined as

$$\mathbf{\Gamma}_t = \begin{pmatrix} \mathbf{\Gamma}^{(f,1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Gamma}^{(f,2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Gamma}^{(f,3)} \end{pmatrix}, \qquad E_{t-1} = 0, \tag{37}$$

$$\mathbf{\Gamma}_t = \mathbf{\Gamma}^{(c)} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad E_{t-1} = 1, \tag{38}$$

where $\otimes$ denotes the Kronecker product. See Supplement B for an expanded version of $\mathbf{\Gamma}_t$.

6.3. Optimization Procedure

We used a procedure similar to the simulation study to initialize the case study parameters. Let $\bar{D}$ denote the sample mean and s denote sample standard deviation of $\{D_t\}_{t=1}^T$. We initialized the initial estimates for the mean $\left(\mu_0^{(i,j)}\right)$ and the log of the standard deviation $\left(\log\left(\sigma_0^{(i,j)}\right)\right)$ of the state-dependent density of D_t as $\mu_0^{(i,j)} \sim \mathcal{N}(\bar{D}, s^2)$ and $\log\left(\sigma_0^{(i,j)}\right) \sim \mathcal{N}(\log(s), 1)$ for $i,j = 1,2,3$, where i refers to the dive type and j to the dive phase. Further, let $\bar{E}$ represent the mean of $\{E_t\}_{t=1}^T$. We initialized the state-dependent probability of observing a dive end as $p_0^{(i,1)} = 0$, $p_0^{(i,2)} = 0$ and $\text{logit}(p_0^{(i,3)}) \sim \mathcal{N}(\text{logit}(\bar{E}), 1)$ for $i = 1,2,3$, where $p_0^{(i,j)}$ is the initial estimate corresponding to the Bernoulli distribution of E_t during dive type i and dive phase j. Dive phase 3 is ascent.

Let $\boldsymbol{\nu}_k \in \mathbb{R}^3$ denote the parameters associated with $\boldsymbol{\delta}$ at iteration k of a given optimization algorithm. We initialized the first element of $\boldsymbol{\nu}_0$ as zero for identifiability and the second and third elements with a standard normal distribution, $\mathcal{N}(0,1)$.

Let $\boldsymbol{\eta}_k^{(c)} \in \mathbb{R}^{3\times 3}$ denote the parameters associated with the coarse-scale probability transition matrix at iteration k of a given optimization algorithm. The reparameterization from $\boldsymbol{\eta}_k^{(c)}$ to $\mathbf{\Gamma}_k^{(c)}$ is given in (2). We initialized the diagonal elements of $\boldsymbol{\eta}_0^{(c)}$ as zeros, and we initialized the off-diagonal elements of $\boldsymbol{\eta}_0^{(c)}$ with a normal distribution with mean -3 and unit variance, $\mathcal{N}(-3,1)$.

Let $\boldsymbol{\eta}_k^{(f,i)} \in \mathbb{R}^{3\times 3}$ denote the parameters associated with fine-scale transition probability matrix $\mathbf{\Gamma}_k^{(f,i)}$. The reparameterization from $\boldsymbol{\eta}_k^{(f,i)}$ to $\mathbf{\Gamma}_k^{(f,i)}$ is given in (2). We initialized all diagonal

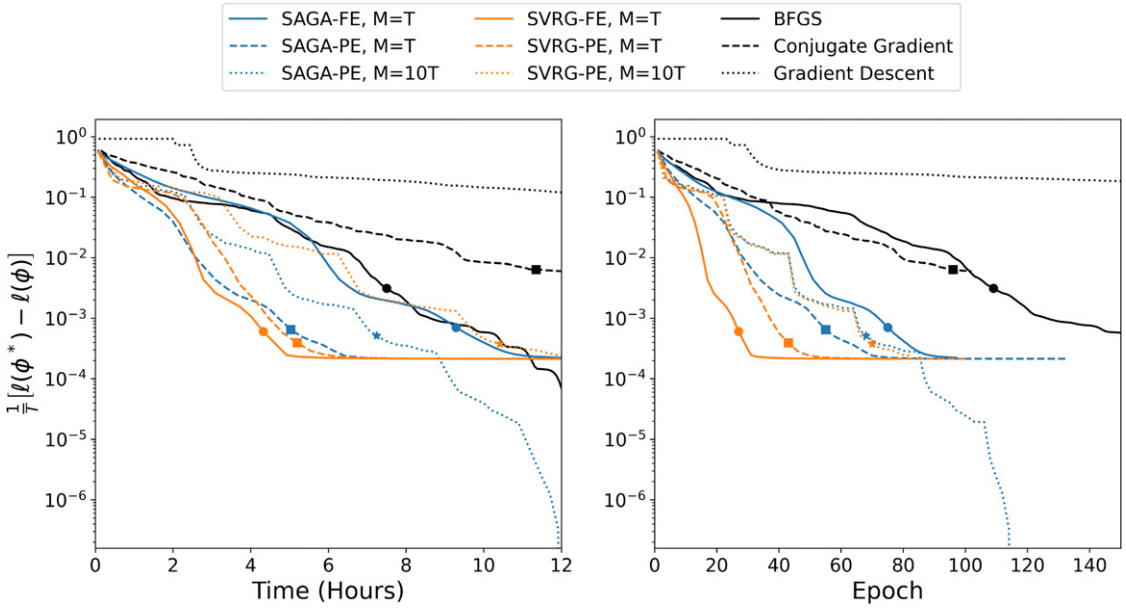

Figure 6. Maximum log-likelihood minus log-likelihood (all over T) versus epoch and computation time for the model from the killer whale case study. All optimization algorithms were run for a total of 12 hr. For each optimization algorithm, we display the one random initialization (of 50) that resulted in the highest likelihood after 12 hr. FE corresponds to $P = \texttt{False}$, and PE corresponds to $P = \texttt{True}$. The log-likelihood of $\boldsymbol{\phi}$ is denoted as $\ell(\boldsymbol{\phi})$ on the y-axis, which is on a log-scale. Dots correspond to the epoch and likelihood at termination for each algorithm.

elements of $\boldsymbol{\eta}_0^{(f,i)}$ as zeros and all off-diagonal elements of $\boldsymbol{\eta}_0^{(f,i)}$ with a normal distribution with mean -1 and unit variance, $\mathcal{N}(-1, 1)$.

Similarly to the simulation study, we estimated the parameters of the model using our six inference algorithms ($A \in \{\text{SVRG}, \text{SAGA}\}$ in addition to $(P, M) \in \{(\texttt{False}, T), (\texttt{True}, T), (\texttt{True}, 10T)\}$) and direct likelihood maximization via three baseline algorithms (BFGS, conjugate gradient, and gradient descent). All algorithms were run using 50 random initializations for a total of 12 hr each on Compute Canada Cedar nodes with 16GB of RAM.

We employed similar measures as those in the simulation study to fairly compare the optimization algorithms. In particular, we measured computational complexity in epochs in addition to raw computation time and defined an epoch using the definition from the simulation study. We also estimated the maximum likelihood parameters $\boldsymbol{\phi}^*$ for each dataset/experiment pair using the same method as the simulation study. Finally, we recorded the epoch and likelihood at termination for each optimization algorithm using the same definition of termination as the simulation study.

6.4. Case Study Results

Our model predicted dive phases and dive types that are in line with previous studies of marine mammal diving behavior. For example, dive types are separated by shallow, medium, and deep depths, which is similar to results from Leos Barajas et al. (2017). Further, each dive has a well-characterized bottom phase that occurs at approximately 70% of the maximum depth for all dive types. This finding is similar to the results from Tennessen et al. (2019a). See Figure 5 and Supplement B for further detail.

Most importantly, this case study demonstrates that all of our stochastic algorithms converged in fewer epochs and to regions of higher likelihood compared to the full-batch baselines; see Figures 6 and 7.

All algorithms occasionally converged to sub-optimal local minima, but Algorithm 5 with $A = \text{SVRG}$, $P = \texttt{True}$, and $M = T$ tended to converge with the highest likelihood relatively quickly in terms of both epoch number and raw computation time (see Figure 7). As with the simulation study, Algorithm 5 with $A = \text{SVRG}$ tended to converge in fewer epochs compared with $A = \text{SAGA}$ (see Figure 6). Setting $P = \texttt{True}$ appears to be of particular use early in the optimization procedure (i.e., the first ≈ 5 epochs, see Figure 6). This behavior is intuitive because the proper weights $\left(\boldsymbol{\gamma}(\boldsymbol{\phi}_k^{(m)})\right.$ and $\left.\boldsymbol{\xi}(\boldsymbol{\phi}_k^{(m)})\right)$ change rapidly early in the optimization procedure.

7. Discussion

The advent of high-frequency sensing technology has allowed researchers to model exceptionally long, high-frequency stochastic processes with increasingly complex HMMs (Patterson et al. 2017). However, these complex models can be computationally expensive to fit (Glennie et al. 2023). We introduce an inference algorithm that speeds up maximum likelihood estimation for HMMs compared to existing batch-gradient methods. We do so without approximating the likelihood, which is required for many existing stochastic inference methods (Gotoh, Hochberg, and Silverman 1998; Ye, Ma, and Qian 2017).

Our method does not require a closed-form solution for the M step, which enables quick inference for a diverse range of HMM models. Such a method is useful in practice because many HMMs lack closed-form solutions for the M step. In finance, Langrock et al. (2018) modeled the relationship between the price of energy and price of the oil in Spain using nonparametric HMMs. In ecology, Lawler et al. (2019) used

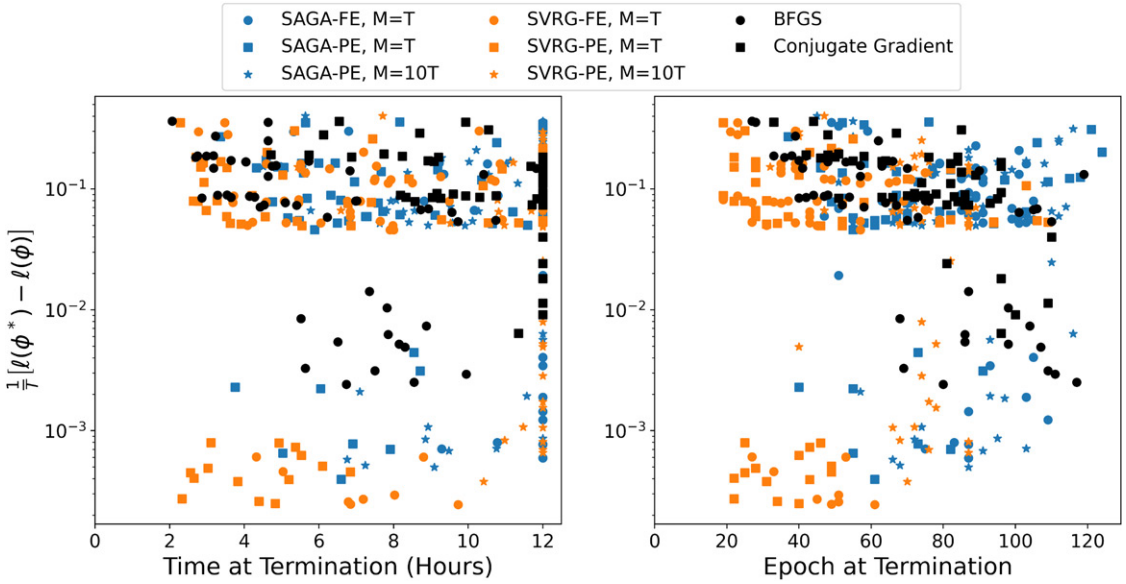

Figure 7. Maximum log-likelihood minus log-likelihood at termination (all over T) versus epochs to terminate for the killer whale case study. Unlike Figure 6, results include all 50 parameter initializations. FE corresponds to $P =$ `False`, and PE corresponds to $P =$ `True`. The log-likelihood of $\boldsymbol{\phi}$ is denoted as $\ell(\boldsymbol{\phi})$ on the y-axis, which is on a log-scale.

autoregression in the state-dependent distributions of an HMM to model the movement of grey seals (*Halichoerus grypus*). Likewise, Pirotta et al. (2018) used covariates in the transition probability matrix of an HMM to determine the effect of fishing boats on the behavior of Northern Fulmars (*Fulmarus glacialis*). Our method will be especially relevant as advances in biologging and tracking technology allow practitioners to collect increasingly large and high-frequency datasets (Patterson et al. 2017).

Our new algorithm was particularly effective when performing inference over large datasets. For example, when applied to simulations with $T = 10^5$ observations, our algorithm converged in approximately half as many epochs and tended to converge to regions of higher likelihood compared to existing baselines. Our case study of killer whale kinematic data showed similar improvements, demonstrating how our optimization procedure makes complex hierarchical HMMs less computationally expensive to fit on large biologging datasets.

The partial E step variant of our algorithm (i.e., with $P =$ `True`) outperforms baselines particularly well early in the optimization procedure (in the first ≈ 5 epochs). As such, using a partial E step may be particularly advantageous when researchers are modeling large datasets with relatively low convergence thresholds.

One method that is particularly aligned with our algorithm is that of Zhu et al. (2017), who implement variance-reduced stochastic optimization to perform the M step of the EM algorithm on high-dimensional latent-variable models. Their method obtains a sub-linear computational complexity in the length of the observation sequence as well as a linear convergence rate. However, they focus primarily on mixture models rather than HMMs, and they do not combine the variance-reduced stochastic M step with a partial E step, which is an extension that we implement here. Further, their theoretical results assume independence between observations, which we do not rely on here.

Future work can explore the performance of our new algorithm when applied to increasingly complex HMMs. Parameter inference for these complicated models may add difficulties beyond those presented in our simulation and case studies. For example, HMMs with covariates in the transition matrix are less stable than time-homogeneous HMMs, and optimization algorithms can easily get stuck in local optima of the likelihood surface. Our optimization algorithm was less prone to getting stuck in local optima than the baseline methods in this work, so the same may be true for more complicated HMMs.

While we use SVRG and SAGA in our analysis, there are other variance-reduced stochastic optimization algorithms that could be applied within our framework. For example, SARAH recursively updates the control variate in the inner loop of the optimization algorithm (Nguyen et al. 2017), SPIDER uses a path-integrated differential estimator of the gradient (Fang et al. 2018), and LiSSA is a second-order variance-reduced stochastic optimization scheme (Agarwal, Bullins, and Hazan 2017). Future work can integrate these algorithms within our framework and evaluate the resulting performance. While we focus on an ecological case study here, the inference procedures we developed can unlock more complicated HMMs with larger latent spaces and bigger datasets for practitioners across a variety of disciplines.

Supplemental Materials

Appendix: Contains alternate version of Algorithm 5 and proofs for all theorems. (`appendix.pdf`, pdf file)

Supplement A: Additional results from the simulation study, including plots of likelihood versus computation time, plots for all five simulated datasets, and results for experiments with $T = 10^3$. (`supp_A.pdf`, pdf file)

Supplement B: Additional results from the case study, including parameter estimates of the final model and dive profiles for all eight killer whales. (`supp_B.pdf`, pdf file)

Code: Code and data to run experiments and plot results from the manuscript can be found at *https://github.com/evsi8432/sublinear-HMM-inference*.

Acknowledgments

All killer whale data was collected under University of British Columbia Animal Care Permit no. A19-0053 and Fisheries and Oceans Canada Marine Mammal Scientific License for Whale Research no. XMMS 6 2019. Tags were deployed by Mike deRoos, the tagging boat skipper was Chris Hall, and photo-ID was done by Taryn Scarff. This research was enabled in part by support provided by WestGrid (www.westgrid.ca) and Compute Canada (www.computecanada.ca). We also thank the reviewers for their constructive comments.

Disclosure Statement

The authors report there are no competing interests to declare.

Funding

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) as well as the support of Fisheries and Oceans Canada (DFO). This project was supported by a financial contribution from the DFO and NSERC (Whale Science for Tomorrow). This work was also supported by the NSERC Discovery program under grant RGPIN-2020-04629; the Canadian Research Chairs program for Statistical Ecology; the BC Knowledge Development fund; the Canada Foundation for Innovation (John R. Evans Leaders Fund) under grant 37715; and the University of British Columbia via the Four-Year Doctoral Fellowship program.

ORCID

Evan Sidrow https://orcid.org/0000-0001-5280-8376
Marie Auger-Méthé https://orcid.org/0000-0003-3550-4930

References

Adam, T., Griffiths, C., Leos Barajas, V., Meese, E., Lowe, C., et al. (2019), "Joint Modelling of Multi-scale Animal Movement Data using Hierarchical Hidden Markov Models," *Methods in Ecology and Evolution*, 10, 1536–1550. [222]

Agarwal, N., Bullins, B., and Hazan, E. (2017), "Second-order Stochastic Optimization for Machine Learning in Linear Time," *The Journal of Machine Learning Research*, 18, 4148–4187. [236]

Baldi, P., and Chauvin, Y. (1993), "Smooth On-Line Learning Algorithms for Hidden Markov Models," *Neural Computation*, 6, 307–318. [223]

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970), "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, 41, 164–171. [222]

Bebbington, M. S. (2007), "Identifying Volcanic Regimes using Hidden Markov Models," *Geophysical Journal International*, 171, 921–942. [222]

Boyd, S., and Vandenberghe, L. (2004), *Convex Optimization*, Cambridge: Cambridge University Press. [228]

Cade, D. E., Gough, W. T., Czapanskiy, M. F., Fahlbusch, J. A., Kahane-Rapport, S. R., Linsky, J. M. J., et al. (2021), "Tools for Integrating Inertial Sensor Data With Video Bio-Loggers, Including Estimation of Animal Orientation, Motion, and Position," *Animal Biotelemetry*, 9, 34. [232]

Chen, J., and Li, P. (2009), "Hypothesis Test for Normal Mixture Models: The EM Approach," *The Annals of Statistics*, 37, 2523–2542. [228]

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014), "SAGA: A Fast Incremental Gradient Method with Support for Non-Strongly Convex Composite Objectives," in *Advances in Neural Information Processing Systems*, NIPS '14. [222,225,228,229]

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, Series B, 39, 1–38. [222,224]

Fang, C., Li, C. J., Lin, Z., and Zhang, T. (2018), "SPIDER: Near-optimal Non-convex Optimization via Stochastic Path-integrated Differential Estimator," in *Advances in Neural Information Processing Systems* (Vol. 31). [236]

Fletcher, R. (2000), *Newton-Like Methods*, Wiley. [228,230]

Fletcher, R., and Reeves, C. M. (1964), "Function Minimization by Conjugate Gradients," *The Computer Journal*, 7, 149–154. [230]

Gales, M., and Young, S. (2007), "The Application of Hidden Markov Models in Speech Recognition," *Foundations Trends Signal Processing*, 1, 195–304. [222]

Glennie, R., Adam, T., Leos-Barajas, V., Michelot, T., Photopoulou, T., and McClintock, B. T. (2023), "Hidden Markov Models: Pitfalls and Opportunities in Ecology," *Methods in Ecology and Evolution*, 14, 43–56. [235]

Gotoh, Y., Hochberg, M., and Silverman, H. (1998), "Efficient Training Algorithms for HMMs Using Incremental Estimation," *IEEE Transactions on Speech and Audio Processing*, 6, 539–548. [223,235]

Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. (2021), "Why Random Reshuffling Beats Stochastic Gradient Descent," *Mathematical Programming*, 186, 49–84. [229]

Jensen, F. H., Tervo, O. M., Heide-Jørgensen, M. P., and Ditlevsen, S. (2023), "Detecting Narwhal Foraging Behaviour From Accelerometer and Depth Data Using Mixed-Effects Logistic Regression," *Animal Biotelemetry*, 11, 14. [231]

Johnson, R., and Zhang, T. (2013), "Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, neurIPS, Red Hook, NY, USA: Curran Associates Inc. [222,225,228]

Karimi, B., Wai, H.-T., Moulines, E., and Lavielle, M. (2019), "On the Global Convergence of (Fast) Incremental Expectation Maximization Methods," in *Advances in Neural Information Processing Systems* (Vol. 32), Curran Associates, Inc. [222]

Khreich, W., Granger, E., Miri, A., and Sabourin, R. (2012), "Survey of Techniques for Incremental Learning Of HMM Parameters," *Information Sciences*, 197, 105–130. [223]

Kingma, D., and Ba, J. (2015), "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*. [222]

Kleinberg, B., Li, Y., and Yuan, Y. (2018), "An Alternative View: When Does SGD Escape Local Minima?" in *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of Proceedings of Machine Learning Research, eds. J. Dy and A. Krause, PMLR. [228]

Kottaram, A., Johnston, L., Cocchi, L., Ganella, E., Everall, I., Pantelis, C., et al. (2019), "Brain Network Dynamics in Schizophrenia: Reduced Dynamism of the Default Mode Network," *Human Brain Mapping*, 40, 2212–2228. [222]

Langrock, R., Adam, T., Leos Barajas, V., Mews, S., Miller, D. L., and Papastamatiou, Y. P. (2018), "Spline-Based Nonparametric Inference in General State-Switching Models," *Statistica Neerlandica*, 72, 179–200. [235]

Lawler, E., Whoriskey, K., Aeberhard, W. H., Field, C., and Mills Flemming, J. (2019), "The Conditionally Autoregressive Hidden Markov Model (CarHMM): Inferring Behavioural States from Animal Tracking Data Exhibiting Conditional Autocorrelation," *Journal of Agricultural, Biological and Environmental Statistics*, 24, 651–668. [235]

Leos Barajas, V., Gangloff, E. J., Adam, T., Langrock, R., van Beest, F. M., Nabe-Nielsen, J., and Morales, J. M. (2017), "Multi-Scale Modeling of Animal Movement and General Behavior Data Using Hidden Markov Models with Hierarchical Structures," *Journal of Agricultural, Biological and Environmental Statistics*, 22, 232–248. [233,235]

Li, Y. (2005), "Hidden Markov Models with States Depending on Observations," *Pattern Recognition Letters*, 26, 977–984. [234]

Li, Z., Han, J., and Song, Y. (2020), "On the Forecasting of High-Frequency Financial Time Series Based on Arima Model Improved by Deep Learning," *Journal of Forecasting*, 39, 1081–1097. [222]

Liu, S., Wu, H., and Meeker, W. Q. (2015), "Understanding and Addressing the Unbounded 'Likelihood' Problem," *The American Statistician*, 69, 191–200. [228]

Mamon, R. S., and Elliott, R. J. (2007), *Hidden Markov Models in Finance*, New York: Springer. [222]

McClintock, B. T., Langrock, R., Gimenez, O., Cam, E., Borchers, D. L., Glennie, R., and Patterson, T. A. (2020), "Uncovering Ecological State Dynamics With Hidden Markov Models," *Ecology Letters*, 23, 1878–1903. [222]

McRae, T., Volpov, B., E. Sidrow, S. F., Auger-Méthé, M., Heckman, N., and Trites, A. (2024), "Killer Whale Respiration Rates," *PloS One*, 19, 1–26. [231,233]

Neal, R. M., and Hinton, G. E. (1998), *A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants*, chapter 12, pp. 355–368, Dordrecht: Springer Netherlands. [222,226,230]

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. (2017), "SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient," in *International Conference on Machine Learning*, PMLR. [236]

Patterson, T. A., Parton, A., Langrock, R., Blackwell, P. G., Thomas, L., and King, R. (2017), "Statistical Modelling of Individual Animal Movement: An Overview of Key Methods and a Discussion of Practical Challenges," *Advances in Statistical Analysis*, 101, 399–438. [222,235,236]

Pearl, J. (1982), "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach," in *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, AAAI Press. [227]

——— (1988), "Chapter 4: Belief Updating by Network Propagation," in *Probabilistic Reasoning in Intelligent Systems*, San Francisco (CA): Morgan Kaufmann, pp. 143–237. [227]

Pirotta, E., Edwards, E. W. J., New, L., and Thompson, P. M. (2018), "Central Place Foragers and Moving Stimuli: A Hidden-State Model to Discriminate the Processes Affecting Movement," *Journal of Animal Ecology*, 87, 1116–1125. [236]

Pohle, J., Langrock, R., van Beest, M., and Schmidt, N. M. (2017), "Selecting the Number of States in Hidden Markov Models: Pragmatic Solutions Illustrated Using Animal Movement," *Journal of Agricultural, Biological and Environmental Statistics*, 22, 1–24. [222]

Robbins, H., and Monro, S. (1951), "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, 22, 400–407. [222,225]

Schmidt, M., Le Roux, N., and Bach, F. (2017), "Minimizing Finite Sums With the Stochastic Average Gradient," *Mathematical Programming*, 162, 83–112. [225,228,229]

Shamir, O. (2016), "Without-Replacement Sampling for Stochastic Gradient Methods," in *Advances in Neural Information Processing Systems* (Vol. 29), Curran Associates, Inc. [229]

Sidrow, E., Heckman, N., Fortune, S. M. E., Trites, A. W., Murphy, I., and Auger-Méthé, M. (2022), "Modelling Multi-Scale, State-Switching Functional Data with Hidden Markov Models," *Canadian Journal of Statistics*, 50, 327–356. [222]

Tamposis, I. A., Theodoropoulou, M. C., Tsirigos, K. D., and Bagos, P. G. (2018), "Extending Hidden Markov Models to Allow Conditioning on Previous Observations," *Journal of Bioinformatics and Computational Biology*, 16, 1850019. [234]

Tennessen, J. B., Holt, M. M., Hanson, M. B., Emmons, C. K., Giles, D. A., and Hogan, J. T. (2019a), "Kinematic Signatures of Prey Capture From Archival Tags Reveal Sex Differences in Killer Whale Foraging Activity," *Journal of Experimental Biology*, 222, jeb191874. [235]

Tennessen, J. B., Holt, M. M., Ward, E. J., Hanson, M. B., Emmons, C. K., Giles, D. A., and Hogan, J. T. (2019b), "Hidden Markov Models Reveal Temporal Patterns and Sex Differences in Killer Whale Behavior," *Scientific Reports*, 9, 14951. [232]

Tennessen, J. B., Holt, M. M., Wright, B. M., Hanson, M. B., Emmons, C. K., Giles, D. A., Hogan, J. T., Thornton, S. J., and Deecke, V. B. (2023), "Divergent Foraging Strategies Between Populations of Sympatric Matrilineal Killer Whales," *Behavioral Ecology*, 34, 373–386. [231]

Thiesson, B., Meek, C., and Heckerman, D. (2001), "Accelerating EM For Large Databases," *Machine Learning*, 45, 279–299. [222]

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2019), "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," arXiv e-prints, arXiv:1907.10121. [230]

Viviant, M., Monestiez, P., and Guinet, C. (2014), "Can We Predict Foraging Success in a Marine Predator from Dive Patterns Only? Validation with Prey Capture Attempt Data," *PloS One*, 9, 1–10. [233]

Wright, B. M., Ford, J. K. B., Ellis, G. M., Deecke, V. B., Shapiro, A. D., Battaile, B. C., and Trites, A. W. (2017), "Fine-Scale Foraging Movements by Fish-Eating Killer Whales (*Orcinus Orca*) Relate to the Vertical Distributions and Escape Responses of Salmonid Prey (*Oncorhynchus* Spp.)," *Movement Ecology*, 5, 3. [231]

Wu, C. F. J. (1983), "On the Convergence Properties of the EM Algorithm," *The Annals of Statistics*, 11, 95–103. [228]

Ye, F. X., Ma, Y., and Qian, H. (2017), "Estimate Exponential Memory Decay in Hidden Markov Model and its Applications," arXiv preprint arXiv:1710.06078. [223,235]

Zhang, G., Poupart, P., and Trimponias, G. (2020), "Comparing EM with GD in Mixture Models of Two Components," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of Proceedings of Machine Learning Research, eds. R. P. Adams and V. Gogate, PMLR. [228]

Zhu, R., Wang, L., Zhai, C., and Gu, Q. (2017), "High-Dimensional Variance-Reduced Stochastic Gradient Expectation-Maximization Algorithm," in *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of Proceedings of Machine Learning Research, PMLR. [236]

Zinkevich, M., Weimer, M., Li, L., and Smola, A. (2010), "Parallelized Stochastic Gradient Descent," in *Advances in Neural Information Processing Systems* (Vol. 23), eds. J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Curran Associates, Inc. [222]

Zucchini, W., Macdonald, I. L., and Langrock, R. (2016), *Hidden Markov Models for Time Series - An Introduction Using R*, Boca Raton, FL: CRC Press. [224,228]