
VARIANCE-REDUCED STOCHASTIC OPTIMIZATION FOR EFFICIENT INFERENCE OF HIDDEN MARKOV MODELS

Siddharth Nand
Department of Statistics
University of British Columbia
Vancouver, Canada

Nancy Heckman
Department of Statistics
University of British Columbia
Vancouver, Canada

Alexandre Bouchard-Côté
Department of Statistics
University of British Columbia
Vancouver, Canada

Sarah M. E. Fortune
Department of Oceanography
Dalhousie University
Halifax, Canada

Andrew W. Trites
Department of Zoology
Institute for the Oceans and Fisheries
University of British Columbia
Vancouver, Canada

Marie Auger-Méthé
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 data sets can be computationally demanding because most likelihood maximization techniques require iterating through the entire underlying data set for every parameter update. We propose a novel optimization algorithm that updates the parameters of an HMM without iterating through the entire data set. 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 and to regions of higher likelihood compared to standard numerical optimization techniques. Our algorithm allows practitioners to fit complicated HMMs to large time-series data sets more efficiently than existing baselines.

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 Baum-Welch algorithm (Baum et al., 1970). The latter is a special case of the expectation-maximization (EM) algorithm (Dempster et al., 1977).

One serious concern for both numerical maximization and the Baum-Welch algorithm is that every parameter update requires iterating through 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 data sets are increasingly collected at high frequencies and contain large numbers of observations (Patterson et al., 2017; Li et al., 2020). These data sets 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 data sets do not require iterating through the entire data set 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, which evaluates the gradient of the likelihood over a random subset of the data (Robbins and Monro, 1951). It is ubiquitous in the optimization literature and has inspired several extensions (Johnson and Zhang, 2013; Defazio et al., 2014; Kingma and Ba, 2015). 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 et al., 2001; Karimi et al., 2019). However, these methods assume that the underlying data set is comprised of independent subsets. This assumption is violated for HMMs since the underlying data set exhibits sequential dependence.

Some work has been done to apply sub-linear inference methods to HMMs. For example, Gotoh et al. (1998) divide the sequence of observations into subsequences and use the incremental EM algorithm to perform inference. This approach assumes that subsequences are independent of one another, which is not true in general. Alternatively, Ye et al. (2017) define a sufficiently large “buffer” before and after subsequences 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 section 3.2.1 of Khreich et al. (2012)), or that the emissions of the HMM are discrete (Baldi and Chauvin, 1993).

In this paper, 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 data set. 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 $\boldsymbol{\delta} = (\delta^{(1)} \dots \delta^{(N)})$, 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

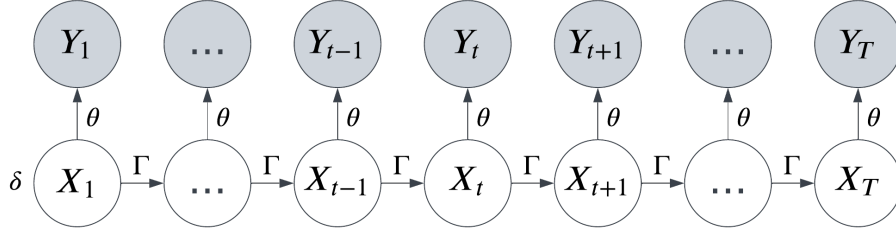


Figure 1: Graphical representation of an HMM. X_t corresponds to an unobserved latent state at time t whose distribution is described by a Markov chain. Y_t corresponds to an observation at time t , where Y_t given all other observations $\mathbf{Y} \setminus \{Y_t\}$ and hidden states \mathbf{X} depends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

probability matrix

$$\mathbf{\Gamma}_t = \begin{pmatrix} \Gamma_t^{(1,1)} & \dots & \Gamma_t^{(1,N)} \\ \vdots & \ddots & \vdots \\ \Gamma_t^{(N,1)} & \dots & \Gamma_t^{(N,N)} \end{pmatrix}, \quad (1)$$

where $\Gamma_t^{(i,j)} = \mathbb{P}(X_t = j \mid X_{t-1} = i)$. For simplicity, we assume that $\mathbf{\Gamma}_t$ does not change over time (i.e. $\mathbf{\Gamma}_t = \mathbf{\Gamma}$ 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 $\mathbf{\Gamma} \in \mathbb{R}^{N \times N}$ and initial distribution $\boldsymbol{\delta} \in \mathbb{R}^N$ in terms of an auxiliary variable $\boldsymbol{\eta}$:

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

where $i, j = 1, \dots, N$ and $\eta^{(i,i)}$ and $\eta^{(1)}$ are set to zero for identifiability. This formulation simplifies likelihood maximization by removing constraints in the optimization problem. One may also incorporate covariates into $\mathbf{\Gamma}$ by setting $\eta_t^{(i,j)}(\boldsymbol{\beta}) = (\boldsymbol{\beta}^{(i,j)})^\top \mathbf{z}_t$, 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 $\mathbf{\Gamma}$ and $\boldsymbol{\delta}$ are functions of $\boldsymbol{\eta}$, we abuse notation in future sections and treat $\mathbf{\Gamma}$ and $\boldsymbol{\delta}$ as variables since the mapping is a bijection.

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 $\boldsymbol{\phi} = \{\boldsymbol{\theta}, \boldsymbol{\eta}\}$. Figure (1) shows an HMM as a graphical model.

The joint likelihood of an HMM given observations \mathbf{Y} and latent states \mathbf{X} is

$$p(\mathbf{X}, \mathbf{Y}; \boldsymbol{\phi}) = \delta^{(X_1)} f^{(X_1)}(Y_1; \theta^{(X_1)}) \prod_{t=2}^T \Gamma^{(X_{t-1}, X_t)} f^{(X_t)}(Y_t; \theta^{(X_t)}). \quad (3)$$

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

$$p(\mathbf{Y}; \boldsymbol{\phi}) = \boldsymbol{\delta} P(Y_1; \boldsymbol{\theta}) \prod_{t=2}^T \mathbf{\Gamma} P(Y_t; \boldsymbol{\theta}) \mathbf{1}_N^\top, \quad (4)$$

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 et al. (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 the 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* $\alpha_t^{(i)} = p(Y_{1:t}, X_t =$

$i; \phi$) (for $i = 1, \dots, N$ and $t = 1, \dots, T$) and *backward probabilities* $\beta_t^{(i)} = p(Y_{(t+1):T} \mid X_t = i; \phi)$ (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 $\alpha_t = \begin{pmatrix} \alpha_t^{(1)} & \dots & \alpha_t^{(N)} \end{pmatrix}$ and $\beta_t = \begin{pmatrix} \beta_t^{(1)} & \dots & \beta_t^{(N)} \end{pmatrix}$. Both α_t and β_t can be calculated using the following recursions:

$$\alpha_1 = \delta P(Y_1; \theta), \quad \alpha_t = \alpha_{t-1} \Gamma P(Y_t; \theta), \quad t = 2, \dots, T, \quad (5)$$

$$\beta_T^\top = \mathbf{1}_N^\top, \quad \beta_t^\top = \Gamma P(Y_{t+1}; \theta) \beta_{t+1}^\top, \quad t = 1, \dots, T-1. \quad (6)$$

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

$$\xi_t = \begin{pmatrix} \xi_t^{(1,1)} & \dots & \xi_t^{(1,N)} \\ \vdots & \ddots & \vdots \\ \xi_t^{(N,1)} & \dots & \xi_t^{(N,N)} \end{pmatrix}, \quad t = 2, \dots, T.$$

Both γ_t and ξ_t can be calculated from α_{t-1} , α_t , β_t , Γ , and θ . Let $\text{diag}(\cdot)$ map a row vector to the diagonal matrix with that row vector as its diagonal. Then,

$$\gamma_t^{(i)} = \frac{\alpha_t^{(i)} \beta_t^{(i)}}{\alpha_t \beta_t^\top}, \quad \gamma_t = \frac{\alpha_t \text{diag}(\beta_t)}{\alpha_t \beta_t^\top}, \quad (7)$$

$$\xi_t^{(i,j)} = \frac{\alpha_{t-1}^{(i)} \Gamma^{(i,j)} f^{(j)}(y_t; \theta^{(j)}) \beta_t^{(j)}}{\alpha_{t-1} \Gamma P(y_t; \theta) \beta_t^\top}, \quad \xi_t = \frac{\text{diag}(\alpha_{t-1}) \Gamma P(y_t; \theta) \text{diag}(\beta_t)}{\alpha_{t-1} \Gamma P(y_t; \theta) \beta_t^\top}. \quad (8)$$

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

2.3 The Baum-Welch Algorithm

The Baum-Welch algorithm is a specific instance of the EM algorithm used to estimate the parameters of an HMM. At iteration k of the EM algorithm, denote the current parameter estimates as ϕ_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 $Q(\phi \mid \phi_k)$ is defined as the expected value of the joint log-likelihood $\log p(\mathbf{X}, \mathbf{Y}; \phi)$ taken with respect to \mathbf{X} , where \mathbf{X} has conditional probability mass function $p(\mathbf{X} \mid \mathbf{Y}; \phi_k)$. For the M step, the next parameter estimate ϕ_{k+1} is found by maximizing $Q(\phi \mid \phi_k)$ with respect to ϕ :

$$Q(\phi \mid \phi_k) = \mathbb{E}_{\phi_k} [\log p(\mathbf{X}, \mathbf{Y}; \phi) \mid \mathbf{Y}], \quad (9)$$

$$\phi_{k+1} = \arg \max_{\phi} Q(\phi \mid \phi_k). \quad (10)$$

For notational convenience, we occasionally denote the set of conditional probabilities $\{\alpha_t(\phi_k), \beta_t(\phi_k), \gamma_t(\phi_k), \xi_t(\phi_k)\}$ as $\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}$. Substituting Equation (3) into Equation (9) and performing some algebra yields a closed form expression for Q :

$$Q(\phi \mid \phi_k) = \sum_{i=1}^N \gamma_{k,1}^{(i)} \log \delta^{(i)}(\eta) + \sum_{t=1}^T \sum_{i=1}^N \gamma_{k,t}^{(i)} \log f^{(i)}(y_t; \theta^{(i)}) + \sum_{t=2}^T \sum_{i=1}^N \sum_{j=1}^N \xi_{k,t}^{(i,j)} \log \Gamma^{(i,j)}(\eta). \quad (11)$$

The conditional probabilities $\gamma_{k,t}^{(i)}$ and $\xi_{k,t}^{(i,j)}$ thus act as weights for $\log \delta^{(i)}(\eta)$, $\log f^{(i)}(y_t; \theta^{(i)})$, and $\log \Gamma^{(i,j)}(\eta)$ for $i, j = 1, \dots, N$. We thus refer to γ and ξ as weights in future sections. Detailed pseudocode for the E and the M step are given in Algorithms (1) and (2) below. In some simple scenarios, the maximization problem in Equation (10) 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(ϕ)

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

1: $\delta = \delta(\eta)$, $\Gamma = \Gamma(\eta)$

2: $\alpha_1 = \delta P(y_1; \theta)$

3: $\beta_T^\top = \mathbf{1}_N^\top$

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

5:

$$\alpha_t = \alpha_{t-1} \Gamma P(y_t; \theta), \quad \beta_{T-t+1}^\top = \Gamma P(y_{T-t+2}; \theta) \beta_{T-t+2}^\top$$

6: **end for**

7: $\gamma_1 = \frac{\alpha_1 \text{diag}(\beta_1)}{\alpha_1 \beta_1^\top}$

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

9:

$$\gamma_t = \frac{\alpha_t \text{diag}(\beta_t)}{\alpha_t \beta_t^\top}, \quad \xi_t = \frac{\text{diag}(\alpha_{t-1}) \Gamma P(y_t; \theta) \text{diag}(\beta_t)}{\alpha_{t-1} \Gamma P(y_t; \theta) \beta_t^\top}$$

10: **end for**

11: **return** $\{\alpha_t, \beta_t, \gamma_t, \xi_t\}_{t=1}^T$

Algorithm 2 Baum-Welch(ϕ_0, K)

Require: Initial parameter values ϕ_0 , number of iterations K

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

2: $\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T = \text{E-step}(\phi_k)$

▷ E step

3:

▷ M step

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

4: **end for**

5: **return** ϕ_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

$$\phi^* = \arg \min_{\phi} F(\phi), \quad \text{where} \quad F(\phi) = \frac{1}{T} \sum_{t=1}^T F_t(\phi). \quad (12)$$

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

In contrast, stochastic gradient descent (SGD) updates ϕ using $\phi^{(m+1)} = \phi^{(m)} - \lambda_m \nabla F_{t_m}(\phi^{(m)})$, where $t_m \in \{1, \dots, T\}$ 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 $\phi^{(m)}$. 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 $m \rightarrow \infty$ to ensure convergence. Further, SGD has slower convergence rates than full gradient descent (Schmidt et al., 2017).

Variance-reduced stochastic optimization techniques such as stochastic average gradient descent (SAG) (Schmidt et al., 2017), stochastic variance reduced gradient descent (SVRG) (Johnson and Zhang, 2013), and SAGA (Defazio et al., 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 $t = 1, \dots, T$, whose average approximates the full gradient $\nabla F(\phi^{(m)})$. 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 $\phi^{(m)}$ via $\phi^{(m+1)} = \phi^{(m)} - \lambda_m [\nabla F_{t_m}(\phi^{(m)}) - \widehat{\nabla} F_{t_m}^{(m)} + \widehat{\nabla} F^{(m)}]$, where $t_m \in \{1, \dots, T\}$ 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($F, \phi^{(0)}, \lambda, A, M$)

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

```

1: for  $t = 1, \dots, T$  do ▷ initialize gradients
2:    $\widehat{\nabla} F_t^{(0)} = \nabla F_t(\phi^{(0)})$ 
3: end for
4:  $\widehat{\nabla} F^{(0)} = (1/T) \sum_{t=1}^T \widehat{\nabla} F_t^{(0)}$ 
5: for  $m = 0, \dots, M-1$  do:
6:   Pick  $t_m \in \{1, \dots, T\}$  uniformly at random.
7: ▷ update parameters

```

$$\phi^{(m+1)} = \phi^{(m)} - \lambda [\nabla F_{t_m}(\phi^{(m)}) - \widehat{\nabla} F_{t_m}^{(m)} + \widehat{\nabla} F^{(m)}] \quad (13)$$

```

8:    $\widehat{\nabla} F_t^{(m+1)} = \widehat{\nabla} F_t^{(m)}$  for  $t = 1, \dots, T$  ▷ update gradient approximations and average
9:   if  $A = \text{SAGA}$  then:

```

$$\widehat{\nabla} F_{t_m}^{(m+1)} = \nabla F_{t_m}(\phi^{(m)}) \quad (14)$$

$$\widehat{\nabla} F^{(m+1)} = \widehat{\nabla} F^{(m)} + \frac{1}{T} (\widehat{\nabla} F_{t_m}^{(m+1)} - \widehat{\nabla} F_{t_m}^{(m)}) \quad (15)$$

```

10:  end if
11: end for
12: return  $\phi^{(M)}$ 

```

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 $\gamma_t(\phi_k)$ and $\xi_t(\phi_k)$ must be calculated for $t = 1, \dots, T$ to define $Q(\phi | \phi_k)$. If closed-form solutions to (10) are not readily available, then the M step is also expensive because evaluating full gradients of $Q(\phi | \phi_k)$ 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 Equation (11) that Q is a large sum and thus implement variance-reduced stochastic optimization. It is straightforward to re-frame the M step of iteration k of the Baum Welch algorithm from Equation (10) so it looks like the minimization problem from Equation (12). To do so, we define $\xi_1 = \emptyset$ and the loss function $F(\cdot | \gamma, \xi)$ as follows:

$$F(\phi \mid \gamma, \xi) = \frac{1}{T} \sum_{t=1}^T F_t(\phi \mid \gamma_t, \xi_t), \quad \text{where} \quad (16)$$

$$F_1(\phi \mid \gamma_1, \xi_1) = - \sum_{i=1}^N \gamma_1^{(i)} \log f^{(i)}(y_t; \theta^{(i)}) - \sum_{i=1}^N \gamma_1^{(i)} \log \delta^{(i)}(\eta), \quad (17)$$

$$F_t(\phi \mid \gamma_t, \xi_t) = - \sum_{i=1}^N \gamma_t^{(i)} \log f^{(i)}(y_t; \theta^{(i)}) - \sum_{i=1}^N \sum_{j=1}^N \xi_t^{(i,j)} \log \Gamma^{(i,j)}(\eta), \quad t = 2, \dots, T. \quad (18)$$

The two functions F and Q are closely related to one another, as $F(\phi \mid \gamma(\phi_k), \xi(\phi_k)) = -Q(\phi \mid \phi_k)/T$. 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 $F(\cdot \mid \gamma(\phi_k), \xi(\phi_k))$ 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 $F(\cdot \mid \gamma(\phi_k), \xi(\phi_k))$, 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 re-calculating the full gradient approximation $\widehat{\nabla} F^{(0)}$, and this involves a full pass of the underlying data set. However, the E step of the Baum-Welch algorithm also involves a full pass of the data set, 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 $\phi^{(0)}$. The E step involves calculating the conditional probabilities $\gamma(\phi^{(0)})$ and $\xi(\phi^{(0)})$, and the M step involves running variance-reduced stochastic optimization with loss function $F(\cdot \mid \gamma(\phi^{(0)}), \xi(\phi^{(0)}))$ and initial parameter value $\phi^{(0)}$. Now, suppose $\phi^{(m)}$ is to be updated using a gradient estimate using a random observation index t_m . The function $F_{t_m}(\cdot \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}))$ depends on $\gamma_{t_m}(\phi^{(0)})$ and $\xi_{t_m}(\phi^{(0)})$, each of which are vectors of conditional probabilities given $\phi^{(0)}$. However, $\phi^{(0)}$ is an out-of-date parameter estimate since the current parameter estimate is $\phi^{(m)}$. Therefore, it is natural to update γ_{t_m} and ξ_{t_m} and redefine $F_{t_m}(\cdot \mid \gamma_{t_m}, \xi_{t_m})$ before calculating $\phi^{(m+1)}$.

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

To this end, we define the mappings $\tilde{\alpha}_t, \tilde{\beta}_t, \tilde{\gamma}_t$, and $\tilde{\xi}_t$ for $t = 1, \dots, T$ similarly to Equations (5–8):

$$\tilde{\alpha}_1(\mathbf{a}, \phi) = \delta P(y_1; \theta), \quad \tilde{\alpha}_t(\mathbf{a}, \phi) = \mathbf{a} \Gamma P(y_t; \theta), \quad t = 2, \dots, T, \quad (19)$$

$$\tilde{\beta}_T^\top(\mathbf{b}, \phi) = \mathbf{1}_N^\top, \quad \tilde{\beta}_t^\top(\mathbf{b}, \phi) = \Gamma P(y_{t+1}; \theta) \mathbf{b}^\top, \quad t = 1, \dots, T-1, \quad (20)$$

$$\tilde{\gamma}_t(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \text{diag}(\mathbf{b})}{\mathbf{a} \mathbf{b}^\top}, \quad t = 1, \dots, T, \quad (21)$$

$$\tilde{\xi}_t(\mathbf{a}, \mathbf{b}, \phi) = \frac{\text{diag}(\mathbf{a}) \Gamma P(y_t; \theta) \text{diag}(\mathbf{b})}{\mathbf{a} \Gamma P(y_t; \theta) \mathbf{b}^\top}, \quad t = 2, \dots, T, \quad (22)$$

all of which take $\mathcal{O}(N^2)$ time to compute. Further, at the beginning of the M step we define conditional probability approximations $\hat{\alpha}_t^{(0)} = \alpha_t(\phi^{(0)})$, $\hat{\beta}_t^{(0)} = \beta_t(\phi^{(0)})$, $\hat{\gamma}_t^{(0)} = \gamma_t(\phi^{(0)})$, and $\hat{\xi}_t^{(0)} = \xi_t(\phi^{(0)})$ for $t = 1, \dots, T$. 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 $\hat{\alpha}_{t_m}^{(m+1)} = \tilde{\alpha}_{t_m}(\hat{\alpha}_{t_m-1}^{(m)}, \phi^{(m)})$ and $\hat{\beta}_{t_m}^{(m+1)} = \tilde{\beta}_{t_m}(\hat{\beta}_{t_m+1}^{(m)}, \phi^{(m)})$, followed by $\hat{\gamma}_{t_m}^{(m+1)} = \tilde{\gamma}_{t_m}(\hat{\alpha}_{t_m}^{(m+1)}, \hat{\beta}_{t_m}^{(m+1)})$ and $\hat{\xi}_{t_m}^{(m+1)} = \tilde{\xi}_{t_m}(\hat{\alpha}_{t_m-1}^{(m+1)}, \hat{\beta}_{t_m}^{(m+1)}, \phi^{(m)})$. Finally, the loss function at index t_m can be defined as $F_{t_m} \left(\cdot \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right)$.

Updating F_{t_m} in 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 Equations (19 – 22) 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 Equations (19 – 22) only once to approximate $\gamma_{t_m}(\phi^{(m)})$ and $\xi_{t_m}(\phi^{(m)})$. 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. $P = \text{False}$) or if SVRG is used as the optimization algorithm, then either the conditional probability approximations $\left\{ \hat{\alpha}_t^{(m)}, \hat{\beta}_t^{(m)}, \hat{\gamma}_t^{(m)}, \hat{\xi}_t^{(m)} \right\}_{t=1}^T$ or the gradient approximations $\left\{ \hat{\nabla} F_t^{(m)} \right\}_{t=1}^T$ 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. $\log p(\mathbf{y}; \phi_{k,\ell}) \geq \log p(\mathbf{y}; \phi_k)$) 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 $\log p(\mathbf{y}; \phi_{k,\ell}) \geq \log p(\mathbf{y}; \phi_k)$ to exit the while loop of EM-VRSO, since this requirement may cause an infinite loop if it cannot be met. 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 below that $\mathbb{P}(\ell^*(k) < \infty) = 1$. One final concern is whether the sequence $\{\phi_k\}_{k=0}^\infty$ generated by EM-VRSO converges to a local maximum of the likelihood as $K \rightarrow \infty$. Theorem 1 below guarantees such convergence under standard regularity conditions. We prove Theorem 1 in the appendix.

Theorem 1. Suppose the following conditions are met in Algorithm (6) with $P = \text{False}$ and $A = \text{SVRG}$:

1. The parameters ϕ lie in $\Phi = \mathbb{R}^r$ for some dimension r .

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

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

```

1: for  $t = 1, \dots, T$  do ▷ initialize gradients
2:    $\hat{\nabla} F_t^{(0)} = \nabla F_t \left( \phi^{(0)} \mid \hat{\gamma}_t^{(0)}, \hat{\xi}_t^{(0)} \right)$ 
3: end for
4:  $\hat{\nabla} F^{(0)} = (1/T) \sum_{t=1}^T \hat{\nabla} F_t^{(0)}$ 
5: for  $m = 0, \dots, M-1$  do:
6:   Pick  $t_m \in \{1, \dots, T\}$  uniformly at random.
7:    $\left\{ \hat{\alpha}_t^{(m+1)}, \hat{\beta}_t^{(m+1)}, \hat{\gamma}_t^{(m+1)}, \hat{\xi}_t^{(m+1)} \right\} = \left\{ \hat{\alpha}_t^{(m)}, \hat{\beta}_t^{(m)}, \hat{\gamma}_t^{(m)}, \hat{\xi}_t^{(m)} \right\}$  for  $t = 1, \dots, T$ .
8:   if  $P = \text{True}$  then ▷ partial E step
9:      $\hat{\alpha}_{t_m}^{(m+1)} = \tilde{\alpha}_{t_m} \left( \hat{\alpha}_{t_m-1}^{(m)}, \phi^{(m)} \right), \quad \hat{\beta}_{t_m}^{(m+1)} = \tilde{\beta}_{t_m} \left( \hat{\beta}_{t_m+1}^{(m)}, \phi^{(m)} \right)$ 
10:     $\hat{\gamma}_{t_m}^{(m+1)} = \tilde{\gamma}_{t_m} \left( \hat{\alpha}_{t_m}^{(m+1)}, \hat{\beta}_{t_m}^{(m+1)} \right), \quad \hat{\xi}_{t_m}^{(m+1)} = \tilde{\xi}_{t_m} \left( \hat{\alpha}_{t_m-1}^{(m+1)}, \hat{\beta}_{t_m}^{(m+1)}, \phi^{(m)} \right)$ 
11:   end if
12:   ▷ update parameters

$$\phi^{(m+1)} = \phi^{(m)} - \lambda \left[ \nabla F_{t_m} \left( \phi^{(m)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) - \hat{\nabla} F_{t_m}^{(m)} + \hat{\nabla} F^{(m)} \right] \quad (23)$$

13:    $\hat{\nabla} F_t^{(m+1)} = \hat{\nabla} F_t^{(m)}$  for  $t = 1, \dots, T$  ▷ update gradients
14:   if  $A = \text{SAGA}$  then:

$$\hat{\nabla} F_{t_m}^{(m+1)} = \nabla F_{t_m} \left( \phi^{(m)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right), \quad (24)$$


$$\hat{\nabla} F^{(m+1)} = \hat{\nabla} F^{(m)} + \frac{1}{T} \left( \hat{\nabla} F_{t_m}^{(m+1)} - \hat{\nabla} F_{t_m}^{(m)} \right). \quad (25)$$

15:   end if
16: end for
17: return  $\phi^{(M)}$ 

```

Algorithm 5 EM-VRSO($\phi_0, \lambda, A, P, M, K$) (Version 1)

Require: Initial parameters (ϕ_0), step size (λ), algorithm $A \in \{\text{SVRG}, \text{SAGA}\}$, whether to do a partial E step $P \in \{\text{True}, \text{False}\}$, iterations per update (M), and number of updates (K).

```

1: for  $k = 0, \dots, K-1$  do
2:    $\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T = \text{E-step}(\phi_k)$  ▷ E step
3:    $\ell \leftarrow 0$  ▷ M step
4:   while  $\ell = 0$  or  $\log p(\mathbf{y}; \phi_{k,\ell}) < \log p(\mathbf{y}; \phi_k)$  do
5:      $\ell \leftarrow \ell + 1$ 
6:      $\phi_{k,\ell} = \text{VRSO-PE}(\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T, \phi_k, \lambda, A, P, M)$ 
7:   end while
8:    $\phi_{k+1} = \phi_{k,\ell}$ 
9: end for
10: return  $\phi_K$ 

```

2. $\Phi_{\phi_0} = \{\phi \in \Phi : \log p(\mathbf{y}; \phi) \geq \log p(\mathbf{y}; \phi_0)\}$ is compact for all ϕ_0 if $\log p(\mathbf{y}; \phi_0) > -\infty$.
3. $\log p(\mathbf{y}; \phi)$ is differentiable in ϕ for all $\phi \in \Phi$.
4. $F_t(\phi | \gamma_t(\phi'), \xi_t(\phi'))$ is convex with respect to ϕ and $F(\phi | \gamma(\phi'), \xi(\phi'))$ is strongly convex with respect to ϕ for all ϕ' with constant $C > 0$. Namely, for all ϕ, ϕ_0 and ϕ' :

$$F_t(\phi | \gamma_t(\phi'), \xi_t(\phi')) \geq F_t(\phi_0 | \gamma_t(\phi'), \xi_t(\phi')) + \nabla F_t(\phi_0 | \gamma_t(\phi'), \xi_t(\phi'))^T (\phi - \phi_0), \quad (26)$$

$$F(\phi | \gamma(\phi'), \xi(\phi')) \geq F(\phi_0 | \gamma(\phi'), \xi(\phi')) + \nabla F(\phi_0 | \gamma(\phi'), \xi(\phi'))^T (\phi - \phi_0) + \frac{C}{2} \|\phi - \phi_0\|_2^2. \quad (27)$$

5. $F_t(\phi | \gamma_t(\phi'), \xi_t(\phi'))$ is uniformly Lipschitz-smooth with respect to ϕ for all t and ϕ' with constant $L \geq C > 0$. Namely, for all t, ϕ, ϕ_0 and ϕ' :

$$F_t(\phi | \gamma_t(\phi'), \xi_t(\phi')) \leq F_t(\phi_0 | \gamma_t(\phi'), \xi_t(\phi')) + \nabla F_t(\phi_0 | \gamma_t(\phi'), \xi_t(\phi'))^T (\phi - \phi_0) + \frac{L}{2} \|\phi - \phi_0\|_2^2. \quad (28)$$

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

$$\zeta = \frac{1}{C\lambda(1 - 2L\lambda)M} + \frac{2L\lambda}{1 - 2L\lambda} < 1. \quad (29)$$

7. $\nabla F_t(\phi | \gamma_t(\phi'), \xi_t(\phi'))$ is uniformly continuous in (ϕ, ϕ') 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 $\{\phi_k\}_{k=0}^\infty$ are stationary points of $\log p(\mathbf{y}; \cdot)$, and $\log p(\mathbf{y}; \phi_k)$ converges monotonically to $\log p^* = \log p(\mathbf{y}; \phi^*)$ for some stationary point of $\log p(\mathbf{y}; \cdot)$, ϕ^* .

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 Gaussian state-dependent distributions within an HMM. This issue is well-known for maximum likelihood estimation in mixture models (Chen and Li, 2009; Liu et al., 2015). It can be avoided by setting lower bounds on the variance components (Zucchini et al., 2016).

Theorem 1 above applies only to version 2 of EM-VRSO when $P = \text{False}$ and $A = \text{SVRG}$. Unfortunately, convergence analysis for EM-VRSO is more complicated when $P = \text{True}$ than when $P = \text{False}$ because the E and M steps are mixed. However, Theorem 2 below shows that stationary points of the log-likelihood are fixed points of Algorithm (5) for all values of P and A . We prove Theorem 2 in the online appendix.

Theorem 2. *If $\nabla \log p(\mathbf{y}; \phi_0) = 0$, then for all $\lambda \in \mathbb{R}$, $A \in \{\text{SAGA}, \text{SVRG}\}$, $P \in \{\text{True}, \text{False}\}$, $M \in \mathbb{N}$, and $K \in \mathbb{N}$, $\text{EM-VRSO}(\phi_0, \lambda, A, P, M, K) = \phi_0$ with probability 1, where EM-VRSO is defined in Algorithm (5).*

Theorem 2 is useful because it guarantees that Algorithm (5) does not change ϕ_0 when it is a stationary point of the likelihood. However, it makes no guarantees that Algorithm (5) will converge if $\nabla \log p(\mathbf{y}; \phi_0) \neq 0$ and either $P = \text{True}$ or $A = \text{SAGA}$. 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 $P = \text{True}$ or $A = \text{SAGA}$ for a predetermined number of iterations, followed by switching to $P = \text{False}$ and $A = \text{SVRG}$, 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 et al. (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 et al. (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:

$$F_{t_m}^{(m+1)}\left(\phi^{(m)} - \frac{1}{\hat{L}}\nabla F_{t_m}^{(m+1)}(\phi^{(m)})\right) \leq F_{t_m}^{(m+1)}\left(\phi^{(m)}\right) - \frac{1}{2\hat{L}}\left\|\nabla F_{t_m}^{(m+1)}(\phi^{(m)})\right\|^2, \quad \text{where} \\ F_{t_m}^{(m+1)} = F_{t_m}\left(\cdot \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)}\right) \quad (30)$$

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 et al. (2017) and do not check the inequality if $\|\nabla F_{t_m}^{(m+1)}(\phi^{(m)})\| < 10^{-8}$ 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 et al., 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 two separate optimization problems over both θ and η (recall that $\phi = \{\theta, \eta\}$). In particular, we can rewrite F_t as $F_t(\phi \mid \gamma_t, \xi_t) = G_t(\theta \mid \gamma_t, \xi_t) + H_t(\eta \mid \gamma_t, \xi_t)$, where

$$G_t(\theta \mid \gamma_t, \xi_t) = -\sum_{i=1}^N \gamma_t^{(i)} \log f^{(i)}(y_t; \theta^{(i)}), \quad t = 1, \dots, T, \quad (31)$$

$$H_1(\eta \mid \gamma_t, \xi_t) = -\sum_{i=1}^N \gamma_t^{(i)} \log \delta^{(i)}(\eta), \quad (32)$$

$$H_t(\eta \mid \gamma_t, \xi_t) = -\sum_{i=1}^N \sum_{j=1}^N \xi_t^{(i,j)} \log \Gamma^{(i,j)}(\eta), \quad t = 2, \dots, T. \quad (33)$$

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

$$\theta^{(m+1)} = \theta^{(m)} - \lambda_\theta \left[\nabla G_{t_m} \left(\theta^{(m)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) - \hat{\nabla} G_{t_m}^{(m)} + \hat{\nabla} G^{(m)} \right], \quad (34)$$

$$\eta^{(m+1)} = \eta^{(m)} - \lambda_\eta \left[\nabla H_{t_m} \left(\eta^{(m)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) - \hat{\nabla} H_{t_m}^{(m)} + \hat{\nabla} H^{(m)} \right]. \quad (35)$$

where $\lambda_\theta = \lambda_\eta = \lambda$. Note that G_t is a function of θ alone and H_t is a function of η alone for given $\hat{\gamma}_t$ and $\hat{\xi}_t$ and all $t = 1, \dots, T$. As such, we allow $\lambda_\theta \neq \lambda_\eta$ and have each depend upon different Lipschitz constant estimates: $\lambda_\theta = 1/(3\hat{L}_G)$ and $\lambda_\eta = 1/(3\hat{L}_H)$. The line search described in Section 4.1 can then be 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 et al. (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 $\lambda_\theta = 1/(3\hat{L}_G)$ and $\lambda_\eta = 1/(3\hat{L}_H)$ for all experiments and algorithms. However, for Algorithm (5) with $P = \text{True}$, the objective function $F(\cdot | \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)}) = G(\cdot | \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)}) + H(\cdot | \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)})$ itself changes over the course of a single M step as $\hat{\gamma}_{t_m}^{(m+1)}$ and $\hat{\xi}_{t_m}^{(m+1)}$ are updated. As a result, more conservative (i.e. smaller) step sizes λ_θ and λ_η 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 decrease after iteration k and attempt ℓ through the M step of Algorithm (5) with $P = \text{True}$, 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 $\lambda_\theta = 1/(3\hat{L}_G)$ and $\lambda_\eta = 1/(3\hat{L}_H)$ for attempt ℓ through a given M step, and attempt ℓ results in an increase of the log-likelihood, we define new step-sizes $\lambda_\theta \leftarrow 1/(6\hat{L}_G)$ and $\lambda_\eta \leftarrow 1/(6\hat{L}_H)$ 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 re-sample the data set without replacement. Sampling without replacement for SGD is often easier to implement and performs better than sampling with replacement (Gürbüzbalaban et al., 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 data sets. For every experiment and data set, $Y_t | X_t = i$ followed a normal distribution with mean $\mu^{(i)}$ and covariance matrix $\Sigma^{(i)}$. We defined $\mu^{(i)}$ and $\Sigma^{(i)}$ for every data set as $\mu^{(i)} \sim \mathcal{N}(0, I)$ and $\Sigma^{(i)} = \text{diag}[\exp(-2)]$ for $i \in \{1, \dots, N\}$ where I is the identity matrix. We set the transition probability matrices of the generating process to depend upon T to keep the expected number of total transitions at 100. This simulates sequences of observations that are sampled at either low- or high- frequencies. Denote the true transition probability matrix from an experiment with T observations and N hidden states as $\Gamma_{T,N}$. We defined $\Gamma_{10^3,3} \in \mathbb{R}^{3 \times 3}$ to have diagonal elements of 0.9 and off-diagonal elements of 0.05, $\Gamma_{10^3,6} \in \mathbb{R}^{6 \times 6}$ to have diagonal elements of 0.9 and off-diagonal elements of 0.02, $\Gamma_{10^5,3} \in \mathbb{R}^{3 \times 3}$ to have diagonal elements of 0.999 and off-diagonal elements of 5×10^{-4} , and $\Gamma_{10^5,6} \in \mathbb{R}^{6 \times 6}$ to have diagonal elements of 0.999 and off-diagonal elements of 2×10^{-4} . We randomly defined the initial distribution as $\delta \sim \text{dir}(\mathbf{1}_N)$ for every experiment and data set.

5.2 Optimization Procedure

We estimated the parameters of the generating model for all five data sets and all eight experiments using six different versions of Algorithm (5). In particular, we used $A \in \{\text{SVRG}, \text{SAGA}\}$, and for each value of A , we used the combinations $\{P = \text{False}, M = T\}$, $\{P = \text{True}, M = T\}$, and $\{P = \text{True}, M = 10T\}$. Recall that setting $P = \text{True}$ 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 $P = \text{True}$ to test the algorithm when the majority of the computational load was placed on the combined partial E / stochastic M step. We also estimated the HMM parameters using three baseline methods: BFGS (Fletcher, 2000), the conjugate gradient method (Fletcher and Reeves, 1964), and full-batch gradient descent.

We sampled a total of five random parameter initializations for each data set / experiment pair, and then ran all nine optimization algorithms on every parameter initialization. Each parameter initialization was re-used for each algorithm

to ensure consistency. 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 $\boldsymbol{\theta}_0 = \{\mu_0^{(i)}, \Sigma_0^{(i)}\}_{i=1}^N$ as $\mu_0^{(i)} \sim \mathcal{N}(\bar{y}, \text{diag}(\mathbf{Q}))$ and $\Sigma_0^{(i)} = \text{diag}(\mathbf{Q})$ for $i = 1, \dots, N$. We initialized $\boldsymbol{\eta}_0$ as $\eta_0^{(i)} \sim \mathcal{N}(0, 1)$ for $i = 2, \dots, N$ and $\eta_0^{(i,j)} \sim \mathcal{N}(-2, 2^2)$ for $i, j = 1, \dots, N$, where $i \neq j$. Throughout the optimization procedure, we assumed that $\Sigma^{(i)}$ was diagonal for all $i \in \{1, \dots, N\}$, which is in line with the generating model described above. Further, we reparameterized $\Sigma^{(i)}$ as $\Sigma^{(i)} = \text{diag}[\exp(\boldsymbol{\rho}^{(i)})]$ and performed inference on $\boldsymbol{\rho}^{(i)} = (\rho_1^{(i)} \dots \rho_N^{(i)})$ for $i = 1, \dots, N$, which is unconstrained. All six algorithms were initialized with step sizes of $\lambda_{\boldsymbol{\theta}} = 1/3\hat{L}_G$ and $\lambda_{\boldsymbol{\eta}} = 1/3\hat{L}_H$. 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 hours 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 rather than raw computation time. We define one epoch as either T evaluations of Equations (19 – 22) 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 ϕ^* for each data set / experiment pair using the parameters from the best-performing optimization algorithm after 12 hours. We also recorded the epoch of convergence for each optimization algorithm, which 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 hours.

5.3 Simulation Results

The results from the simulation study are shown in Figures (2–4).

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 faster than the baselines for all experiments except for those with $N = 3$. Algorithm (5) with $A = \text{SAGA}$ also converged in significantly fewer epochs when $P = \text{True}$ and $M = 10T$. All methods were prone to converge to local minima, especially when $N = 6$, in which case the likelihood surface was highly multi-modal. However, Algorithm (5) tended to converge to areas of higher likelihood than the baseline for almost all experiments. The sole exception was when $T = 10^3$, $N = 6$, and $d = 6$, where BFGS and the conjugate gradient method were less likely to get stuck in local minima (see Figures (1–2) in Supplement A). However, the aforementioned experiment is a small-data setting, and we are primarily interested in modeling very large data sets. We present figures analogous to Figure (2) for all five data sets corresponding to all experiments in Supplement A. All figures associated experiments with $T = 10^3$ can also be found in Supplement A.

6 Case Study

We additionally tested the performance of our optimization algorithm by modeling the movement of eight northern resident killer whales off the coast of British Columbia, Canada with a data set of over 5000 dives and 80000 depth readings. 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). Researchers also use biologging data sets 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 an HMM to identify three common dive phases (ascent, descent, and bottom) and three dive types that may indicate distinct behaviors of the animal, including resting, 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 hours of attachment. Instruments were retrieved following each deployment using

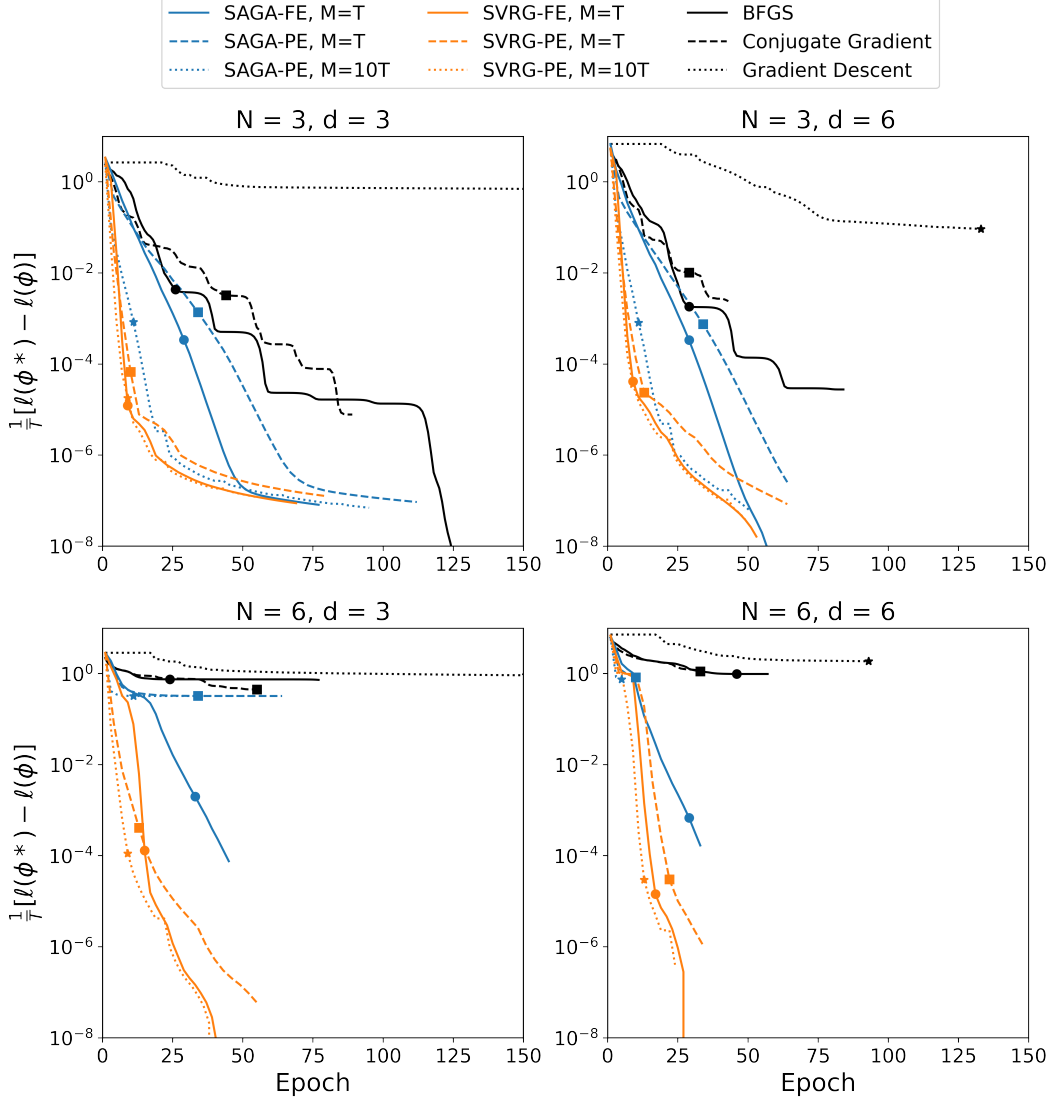


Figure 2: Maximum log-likelihood minus the log-likelihood (all over T) vs epoch for a selected run of each optimization algorithm in the simulation study. Algorithms were run for either 12 hours or 150 epochs (whichever came first) on a data set of experiments with $T = 10^5$, $N \in \{3, 6\}$, and $d \in \{3, 6\}$. FE corresponds to $P = \text{False}$, and PE corresponds to $P = \text{True}$. The log-likelihood is denoted as $\ell(\phi)$, and the y-axis is on a log-scale. We display the random initialization of each algorithm that resulted in the highest likelihood after 12 hours. Dots correspond to the epoch and likelihood at convergence.

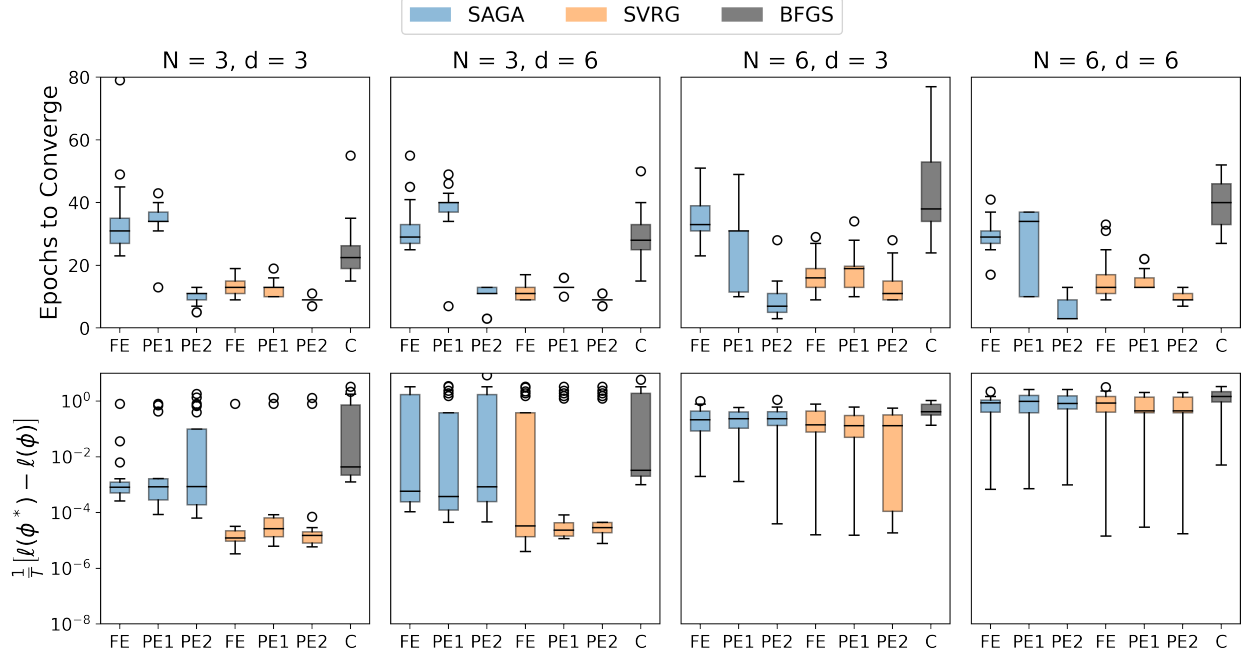


Figure 3: Box plots showing epochs to converge (top) and maximum log-likelihood minus log-likelihood at convergence (all over T , bottom) for each optimization algorithm in the simulation study. Unlike Figure (2), results are displayed for all data sets and initial parameter values. FE corresponds to $\{P = \text{False}, M = T\}$, PE1 corresponds to $\{P = \text{True}, M = T\}$, and PE2 corresponds to $\{P = \text{False}, M = 10T\}$. Results are shown for all simulation studies with $T = 10^5$. The log-likelihood is denoted as $\ell(\phi)$, and the y-axis of the bottom row is on a log-scale.

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 two seconds. We then down-sampled the depth readings to a frequency of 0.5 hertz. The processed data set contained a total of 5858 dives and 89462 depth readings. Figure (5) shows the depth and change in depth for a subset of dives for one whale in the data set.

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 HMM to jointly model dive types and dive phases (Leos Barajas et al., 2017). 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 et al., 2014). This resulted in a total of $N = 9$ hidden states, each corresponding to a different dive phase / dive type combination. Since each dive must begin with the descent phase, we set the initial distribution δ to have the form $\delta = (\delta^{(1)} \ 0 \ 0 \ \delta^{(2)} \ 0 \ 0 \ \delta^{(3)} \ 0 \ 0)$, where $\delta^{(i)}$ represents the probability that a killer whale begins its dive profile with a dive of type i . For this hierarchical HMM, we allowed the dive type to change between dives, but not within a given dive. As such, the form of transition probability matrix changed over time, so we denoted the probability transition matrix as Γ_t . We defined a coarse-scale, inter-dive transition probability matrix $\Gamma^{(c)} \in \mathbb{R}^{3 \times 3}$. For each dive type i , we also defined a fine-scale, intra-dive probability transition matrix $\Gamma^{(f,i)} \in \mathbb{R}^{3 \times 3}$. To ensure that the dive type

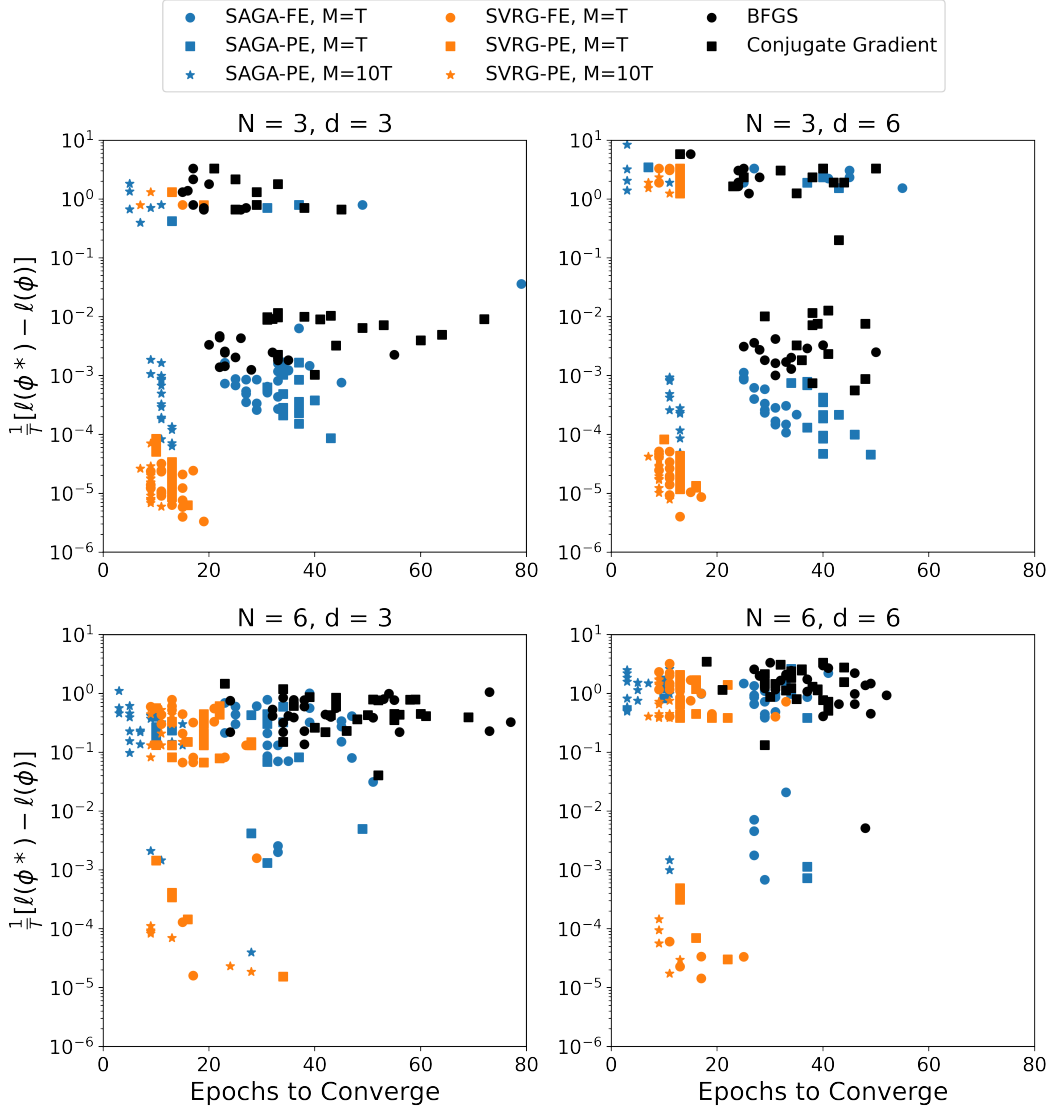


Figure 4: Maximum log-likelihood minus log-likelihood at convergence (all over T) versus epochs to converge for each optimization algorithm in the simulation study with $T = 10^5$, $N \in \{3, 6\}$, and $d \in \{3, 6\}$. FE corresponds to $P = \text{False}$, and PE corresponds to $P = \text{True}$. The log-likelihood is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

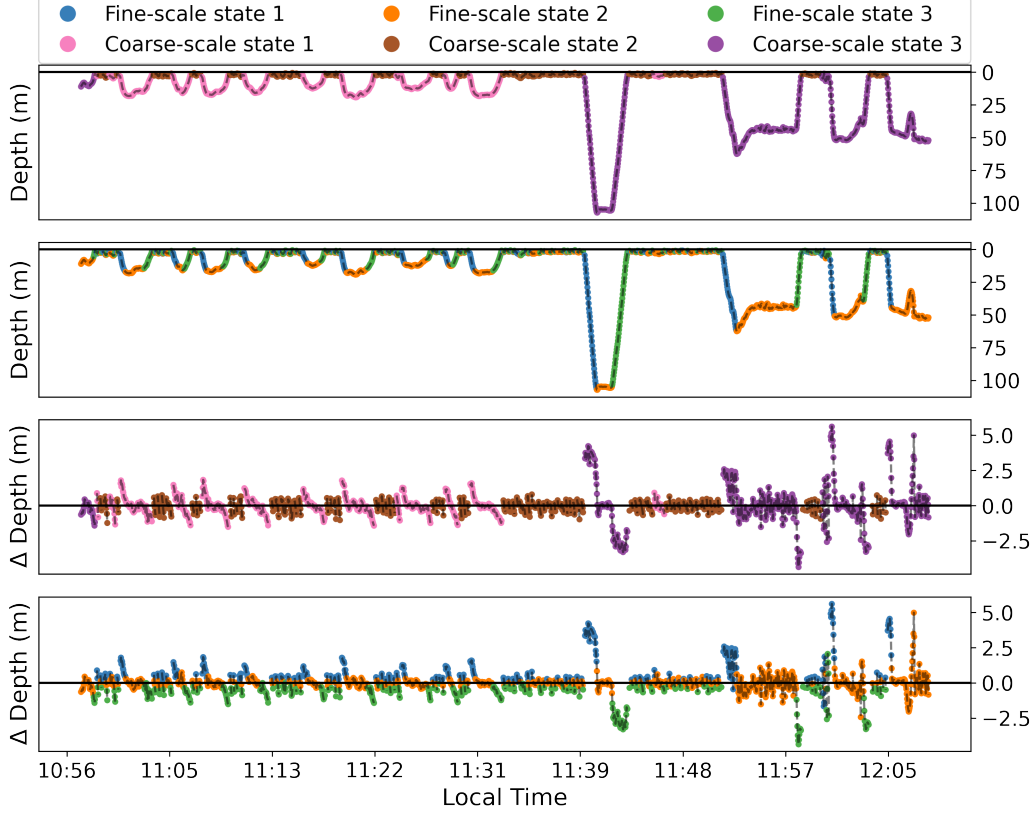


Figure 5: Depth profile and change in depth vs time of day for a selected killer whale (I107, male, born 2004) off the coast of British Columbia, Canada. The data in panels one and three are color-coded according to the most likely hidden coarse-scale state (i.e. dive type) for each dive. The data in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

did not change within a dive, we defined the probability transition matrix *within* a dive to have a block-diagonal form:

$$\mathbf{\Gamma}_t = \begin{pmatrix} \mathbf{\Gamma}^{(f,1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Gamma}^{(f,2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Gamma}^{(f,3)} \end{pmatrix}, \quad (36)$$

where $\mathbf{\Gamma}^{(f,i)}$ is upper-triangular for $i \in \{1, 2, 3\}$. The transition matrices were upper-triangular 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. However, *between* dives, we allowed the coarse-scale dive type to transition, but forced each dive to begin in the descent phase. As such, $\mathbf{\Gamma}_t$ took the following form:

$$\mathbf{\Gamma}_t = \mathbf{\Gamma}^{(c)} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad (37)$$

where \otimes denotes the Kronecker product.

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.

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 standard deviation $\left(\sigma_0^{(i,j)}\right)$ 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 $\eta_k^{(\delta)} \in \mathbb{R}^3$ denote the parameters associated with δ at iteration k of a given optimization algorithm. The reparameterization from $\eta_k^{(\delta)}$ to δ_k is given in Equation (2). We initialized the first element of $\eta_0^{(\delta)}$ as zero for identifiability and the second and third elements with a standard normal distribution, $\mathcal{N}(0, 1)$.

Let $\eta_k^{(c)} \in \mathbb{R}^{3 \times 3}$ denote the parameters associated with the coarse-scale probability transition matrix at iteration k of a given optimization algorithm. The reparameterization from $\eta_k^{(c)}$ to $\Gamma_k^{(c)}$ is given in Equation (2). We initialized the diagonal elements of $\eta_0^{(c)}$ as zeros, and we initialized the off-diagonal elements of $\eta_0^{(c)}$ with a normal distribution with mean -3 and unit variance, $\mathcal{N}(-3, 1)$.

Let $\eta_k^{(f,i)} \in \mathbb{R}^{3 \times 3}$ denote the parameters associated with fine-scale transition probability matrix $\Gamma_k^{(f,i)}$. The reparameterization from $\eta_k^{(f,i)}$ to $\Gamma_k^{(f,i)}$ is given in Equation (2). We initialized all diagonal elements of $\eta_0^{(f,i)}$ as zeros and all off-diagonal elements of $\eta_0^{(f,i)}$ 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 hierarchical HMM using our six inference algorithms ($A \in \{\text{SVRG}, \text{SAGA}\}$ in addition to $(P, M) \in \{\text{False}, T\}, (\text{True}, T), (\text{True}, 10T)\}$) and three baseline algorithms (BFGS, conjugate gradient, and gradient descent). All algorithms were run using 50 random initializations for a total of 12 hours 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 rather than raw computation time and defined an epoch using the definition from the simulation study. We also estimated the maximum likelihood parameters ϕ^* for each data set / experiment pair using the same method as the simulation study. Finally, we recorded the epoch and likelihood of convergence for each optimization algorithm using the same definition of convergence 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 (6–7).

All algorithms occasionally converged to sub-optimal local minima, but Algorithm (5) with $A = \text{SVRG}$ and $P = \text{True}$ and $M = T$ tended to converge in the fewest epochs to regions of highest likelihood; 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 $P = \text{True}$ 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 $\left(\gamma(\phi_k^{(m)}) \text{ and } \xi(\phi_k^{(m)})\right)$ 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

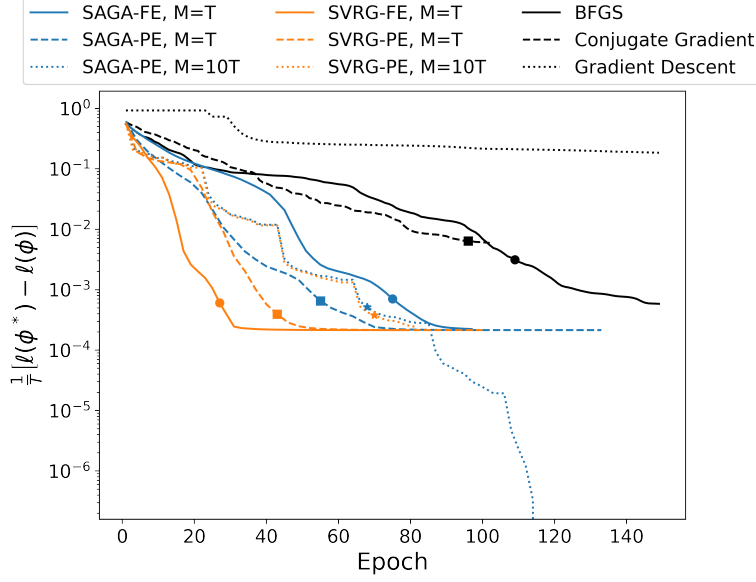


Figure 6: Maximum log-likelihood minus log-likelihood (all over T) vs epoch for 12 hours or 100 epochs (whichever came first) for the HMM from the killer whale case study. FE corresponds to $P = \text{False}$, and PE corresponds to $P = \text{True}$. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale. We display the random initialization of each algorithm that resulted in the highest likelihood after 12 hours. Dots correspond to the epoch and likelihood at convergence for each algorithm.

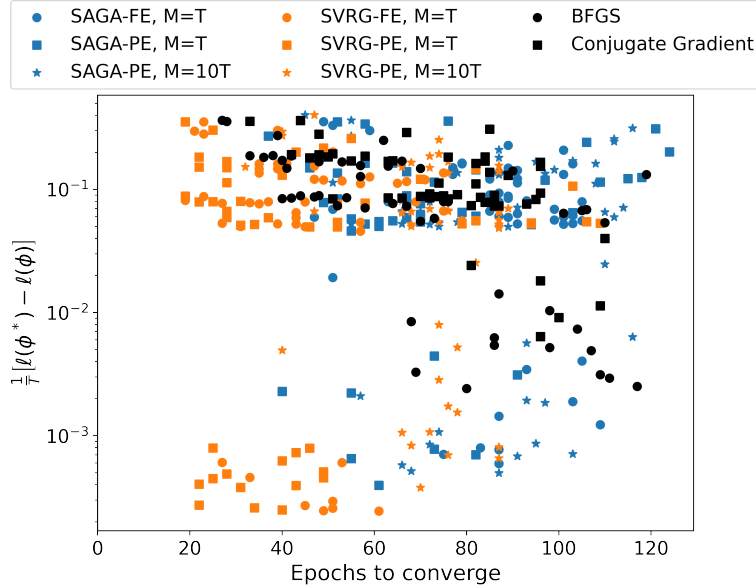


Figure 7: Maximum log-likelihood minus log-likelihood at convergence (all over T) versus epochs to converge for the killer whale case study. FE corresponds to $P = \text{False}$, and PE corresponds to $P = \text{True}$. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

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 et al., 1998; Ye et al., 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. This is useful in practice because many hidden Markov models lack closed-form solutions for the M step. In ecology, Pirodda 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*). Likewise, Lawler et al. (2019) used autoregression in the state-dependent distributions of an HMM to model the movement of grey seals (*Halichoerus grypus*). In finance, Langrock et al. (2018) modelled the relationship between the price of energy and price of the oil in Spain using non-parametric HMMs. The parameters of these models can not be estimated using a naive Baum-Welch algorithm, so our algorithm is a good candidate to speed up inference for complicated HMMs of this type. This will be especially relevant in the future as advances in biologging and tracking technology allow practitioners to collect increasingly large and high-frequency data sets (Patterson et al., 2017).

Our new algorithm was particularly effective when performing inference over large data sets. 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 data sets.

The partial E step variant of our algorithm (i.e. with $P = \text{True}$) 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 data sets 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.

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 et al., 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 data sets for practitioners across a variety of disciplines.

Acknowledgements

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).

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.

Disclosure

The authors report there are no competing interests to declare.

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.
- 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.
- Baldi, P. and Chauvin, Y. (1993), “Smooth On-Line Learning Algorithms for Hidden Markov Models,” *Neural Comput.*, 6.
- 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.
- Bebbington, M. S. (2007), “Identifying Volcanic Regimes using Hidden Markov Models,” *Geophysical Journal International*, 171, 921–942.
- 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.
- Chen, J. and Li, P. (2009), “Hypothesis Test for Normal Mixture Models: The EM Approach,” *The Annals of Statistics*, 37, 2523 – 2542.
- 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.
- 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 (Methodological)*, 39, 1–38.
- Fang, C., Li, C. J., Lin, Z., and Zhang, T. (2018), “SPIDER: Near-optimal Non-convex Optimization via Stochastic Path-integrated Differential Estimator,” *Advances in Neural Information Processing Systems*, 31.
- Fletcher, R. (2000), *Newton-Like Methods*, John Wiley & Sons, Ltd.
- Fletcher, R. and Reeves, C. M. (1964), “Function Minimization by Conjugate Gradients,” *The Computer Journal*, 7, 149–154.
- Gales, M. and Young, S. (2007), “The Application of Hidden Markov Models in Speech Recognition,” *Found. Trends Signal Process.*, 1, 195–304.
- 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.
- 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.
- Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. (2021), “Why Random Reshuffling Beats Stochastic Gradient Descent,” *Mathematical Programming*, 186, 49–84.
- 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.
- 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.
- 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*, volume 32, Curran Associates, Inc.
- Khreich, W., Granger, E., Miri, A., and Sabourin, R. (2012), “Survey of Techniques for Incremental Learning Of HMM Parameters,” *Information Sciences*, 197, 105–130.
- Kingma, D. and Ba, J. (2015), “Adam: A Method for Stochastic Optimization,” in *International Conference on Learning Representations (ICLR)*.
- 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.
- 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.

- 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.
- 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.
- 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.
- Liu, S., Wu, H., and Meeker, W. Q. (2015), “Understanding and Addressing the Unbounded ‘Likelihood’ Problem,” *The American Statistician*, 69, 191–200.
- Mamon, R. S. and Elliott, R. J. (2007), *Hidden Markov Models in Finance*, Springer.
- 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.
- Neal, R. M. and Hinton, G. E. (1998), *A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants*, chapter 12, Dordrecht: Springer Netherlands, 355–368.
- 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.
- 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.
- 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.
- (1988), “Chapter 4: Belief Updating by Network Propagation,” in *Probabilistic Reasoning in Intelligent Systems*, San Francisco (CA): Morgan Kaufmann, 143–237.
- 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.
- 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.
- Robbins, H. and Monro, S. (1951), “A Stochastic Approximation Method,” *The Annals of Mathematical Statistics*, 22, 400 – 407.
- Schmidt, M., Le Roux, N., and Bach, F. (2017), “Minimizing Finite Sums With the Stochastic Average Gradient,” *Mathematical Programming*, 162, 83–112.
- Shamir, O. (2016), “Without-Replacement Sampling for Stochastic Gradient Methods,” in *Advances in Neural Information Processing Systems*, volume 29, Curran Associates, Inc.
- 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.
- 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.
- 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.
- 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.
- Thiesson, B., Meek, C., and Heckerman, D. (2001), “Accelerating EM For Large Databases,” *Machine Learning*, 45, 279–299.
- 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.

- 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.
- 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.
- Wu, C. F. J. (1983), “On the Convergence Properties of the EM Algorithm,” *The Annals of Statistics*, 11, 95–103.
- Ye, F. X., an Ma, Y., and Qian, H. (2017), “Estimate Exponential Memory Decay in Hidden Markov Model and its Applications,” *arXiv preprint arXiv:1710.06078*.
- 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.
- Zucchini, W., Macdonald, I. L., and Langrock, R. (2016), *Hidden Markov Models for Time Series - An Introduction Using R*, CRC Press.

A Version 2 of EM-VRSO

For the purposes of readability, we abuse notation and use the shorthand $F(\phi \mid \gamma(\phi'), \xi(\phi')) \equiv F(\phi \mid \phi')$ and $F_t(\phi \mid \gamma_t(\phi'), \xi_t(\phi')) \equiv F_t(\phi \mid \phi')$ in this appendix. Further, we denote $F^*(\phi') \equiv \inf_{\phi \in \Phi} F(\phi \mid \phi')$.

Algorithm 6 EM-VRSO(ϕ_0, λ, A, M, K) (Version 2)

Require: Initial parameters (ϕ_0), step size (λ), algorithm $A \in \{\text{SVRG}, \text{SAGA}\}$, whether to do a partial E step $P \in \{\text{True}, \text{False}\}$, number of iterations per update (M), and number of updates (K). Assume known Lipschitz constant L and known strong convexity constant C .

- 1: $\zeta = (C\lambda(1 - 2L\lambda)M)^{-1} + (2L\lambda)/(1 - 2L\lambda) < 1$ $\triangleright \zeta < 1$
- 2: **for** $k = 0, \dots, K - 1$ **do**
- 3: $\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T = \text{E-step}(\phi_k)$ \triangleright E step
- 4: $K(\phi_k) = F^*(\phi_k) + [(1 + \zeta)/2] (F(\phi_k \mid \phi_k) - F^*(\phi_k))$
- 5: $\ell \leftarrow 0$ \triangleright M step
- 6: **while** $\ell = 0$ or $F(\phi_{k,\ell} \mid \phi_k) > K(\phi_k)$ **do**
- 7: $\ell \leftarrow \ell + 1$
- 8: $\phi_{k,\ell} = \text{VRSO-PE}(\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T, \phi_k, \lambda, A, P, M)$
- 9: **end while**
- 10: $\phi_{k+1} = \phi_{k,\ell}$
- 11: **end for**
- 12: **return** ϕ_K

One potential concern with Algorithm (6) is that it requires quantities that are not known in practice, namely ζ and $F^*(\phi_k)$. However, these quantities are only used in the while loop to ensure a strict increase in the likelihood for every iteration of VRSO-PE. In addition, version 1 of the algorithm is intuitively preferable since it will accept new parameters that increase the likelihood by any amount. We only use version 2 above to prove Theorem 1.

Another concern with Algorithm (6) is that $F^*(\phi_k)$ is defined using an infimum, which can be problematic in some settings. However, we prove in Lemma 1 that if the conditions of Theorem 1 hold, then $F^*(\phi')$ is finite, has a unique minimizer, and is uniformly continuous.

Lemma 1. *If the conditions of Theorem 1 hold, then for all $\phi_k \in \Phi$, $F^*(\phi_k) = \inf_{\phi \in \Phi} F(\phi \mid \phi_k)$ is finite and has a unique minimizer, so $F^*(\phi_k) = \min_{\phi \in \Phi} F(\phi \mid \phi_k) = F(\phi_{k+1}^* \mid \phi_k)$ for a unique $\phi_{k+1}^* \in \Phi$. Further, $F^*(\phi_k)$ is uniformly continuous for all $\phi_k \in \Phi$.*

Proof. For any fixed ϕ_k , if the conditions of Theorem 1 hold, then there exists a unique minimizer of $F(\cdot \mid \phi_k)$ over Φ_{ϕ_k} because Φ_{ϕ_k} is compact and F is strongly convex and continuous. Denote this minimizer as $\phi_{k+1}^* = \arg \min_{\phi \in \Phi_{\phi_k}} F(\phi \mid \phi_k)$. We show that ϕ_{k+1}^* minimizes $F(\cdot \mid \phi_k)$ over the entirety of Φ using two cases:

1. For any $\phi \in \Phi_{\phi_k}$ with $\phi \neq \phi_{k+1}^*$, we have that $F(\phi \mid \phi_k) > F(\phi_{k+1}^* \mid \phi_k)$ because ϕ_{k+1}^* is the unique minimizer over Φ_{ϕ_k} .
2. For any $\phi \notin \Phi_{\phi_k}$, $\log p(\mathbf{y}; \phi) < \log p(\mathbf{y}; \phi_k) \leq \log p(\mathbf{y}; \phi_{k+1}^*)$ by the definition of Φ_{ϕ_k} . This implies that $F(\phi \mid \phi_k) > F(\phi_k \mid \phi_k) \geq F(\phi_{k+1}^* \mid \phi_k)$ by basic properties of the EM algorithm (Dempster et al., 1977).

Therefore, ϕ_{k+1}^* is a global minimizer of $F(\cdot \mid \phi_k)$, and $F^*(\phi_k) = F(\phi_{k+1}^* \mid \phi_k)$.

We now prove that $F^*(\phi_k)$ is uniformly continuous. Let $\epsilon > 0$ be fixed. Then, because $F(y \mid x)$ is uniformly continuous in $(x, y) \in \Phi \times \Phi$, there exists some δ such that for all (y, x) and (y, x_0) with $|x - x_0| < \delta$ we have that $|F(y \mid x) - F(y \mid x_0)| < \epsilon$. Now, we fix arbitrary x and x_0 with $|x - x_0| < \delta$ and show that $|F^*(x) - F^*(x_0)| < \epsilon$. To this end, pick y^* such that $F(y^* \mid x) = \min_{y \in \Phi} F(y \mid x) = F^*(x)$ and pick y_0^* such that $F(y_0^* \mid x_0) = \min_{y \in \Phi} F(y \mid x_0) = F^*(x_0)$. Then:

$$\begin{aligned} F^*(x_0) &= F(y_0^* \mid x_0) \leq F(y^* \mid x_0) < F(y^* \mid x) + \epsilon = F^*(x) + \epsilon, \\ F^*(x_0) &= F(y_0^* \mid x_0) > F(y_0^* \mid x) - \epsilon \geq F(y^* \mid x) - \epsilon = F^*(x) - \epsilon, \end{aligned}$$

so $|F^*(x) - F^*(x_0)| < \epsilon$. □

Johnson and Zhang (2013)	Our Notation
α	ζ
η	λ
L	L
γ	C
m	M
\tilde{w}_0	ϕ_k
\tilde{w}_1	$\phi_{k,\ell}$
w_*	ϕ_{k+1}^*
P	$F(\cdot \mid \phi_k)$
ψ_i	$F_t(\cdot \mid \phi_k)$

Table 1: Legend connecting this paper’s notation to that of Johnson and Zhang (2013).

B Proof of Theorem 1

We begin by introducing notation needed for the proof. Let $I \in \{1, \dots, T\}^M$ be an M -dimensional vector of indices. Let $\text{VRSO-PE}^*(I, \lambda, \phi')$ correspond to the mapping that results from performing M steps of SVRG on objective function $F(\cdot \mid \phi')$ starting at ϕ' with step size λ and random realization I . That is, $\text{VRSO-PE}^*(I, \lambda, \phi')$ equals $\text{VRSO-PE}(\{\alpha_t(\phi'), \beta_t(\phi'), \gamma_t(\phi'), \xi_t(\phi')\}_{t=1}^T, \phi', \lambda, \text{SVRG}, \text{False}, M)$ when the random vector of indices $(t_0 \dots t_{M-1})$ is equal to I . We prove that $\text{VRSO-PE}^*(I, \lambda, \phi')$ is uniformly continuous in $\phi' \in \Phi$ for fixed I and λ in Lemma 2 below.

Lemma 2. *Suppose all conditions from Theorem 1 hold. Then $\text{VRSO-PE}^*(I, \lambda, \phi')$ is uniformly continuous in $\phi' \in \Phi$ for fixed I and λ .*

Proof. Define $\phi'^{(0)} = \phi'$. The function $\text{VRSO-PE}^*(I, \lambda, \phi')$ involves calculating $\phi'^{(0)}, \phi'^{(1)}, \dots, \phi'^{(m)}, \dots, \phi'^{(M-1)}$ by iteratively applying the following mapping (with $t_m = I[m+1]$) and returning $\phi'^{(M)} \equiv \phi$:

$$\phi'^{(m+1)}(\phi'^{(m)}, \phi'^{(0)}) = \lambda \left[\nabla F_{t_m}(\phi'^{(m)} \mid \phi'^{(0)}) - \nabla F_{t_m}(\phi'^{(0)} \mid \phi'^{(0)}) + \frac{1}{T} \sum_{t=1}^T \nabla F_t(\phi'^{(0)} \mid \phi'^{(0)}) \right].$$

Note that $\nabla F_t(\phi \mid \phi')$ is uniformly continuous in (ϕ, ϕ') for each t by condition (7) of Theorem 1. Therefore, the mapping above is also uniformly continuous in $(\phi'^{(m)}, \phi'^{(0)})$, and so $\text{VRSO-PE}^*(I, \lambda, \phi')$ is uniformly continuous in ϕ' for fixed I and λ . \square

If the conditions of Theorem 1 are met, iteration k of Algorithm (6) is equivalent to drawing a sequence of index vectors $I_{k,0}, I_{k,1}, \dots$ and calculating $\phi_{k,\ell} = \text{VRSO-PE}^*(I_{k,\ell}, \lambda, \phi_k)$ until the condition to exit the while loop is satisfied, which happens in finite time by Lemma 3 below. Lemma 3 depends upon Theorem 1 of Johnson and Zhang (2013), so Table (1) identifies how our notation corresponds to theirs.

Lemma 3. *Suppose all conditions from Theorem 1 hold. Then, the while loop within Algorithm (6) almost surely terminates in finite time, i.e. $\mathbb{P}\{\ell^*(k) < \infty\} = 1$ for all $k \geq 0$. Furthermore, $\mathbb{P}\{\forall k : \ell^*(k) < \infty\} = 1$.*

Proof. By Lemma 1, $F^*(\phi_k) = \inf_{\phi \in \Phi} F(\phi \mid \phi_k)$ is finite and uniformly continuous in ϕ_k . Further, Theorem 1 of Johnson and Zhang (2013) applies for one iteration through M steps of SVRG. In particular, for all $\ell \geq 0$:

$$\mathbb{E} \left[F(\phi_{k,\ell} \mid \phi_k) - F^*(\phi_k) \mid \phi_k \right] \leq \zeta \left(F(\phi_k \mid \phi_k) - F^*(\phi_k) \right), \quad (38)$$

where ζ is defined in condition (6) of Theorem 1. Using (38) in Markov’s inequality gives

$$\mathbb{P} \left[F(\phi_{k,\ell} \mid \phi_k) - F^*(\phi_k) \geq \frac{1+\zeta}{2} (F(\phi_k \mid \phi_k) - F^*(\phi_k)) \mid \phi_k \right] \leq \frac{\zeta (F(\phi_k \mid \phi_k) - F^*(\phi_k))}{\frac{1+\zeta}{2} (F(\phi_k \mid \phi_k) - F^*(\phi_k))} \quad (39)$$

$$= \frac{2}{1+1/\zeta} < 1. \quad (40)$$

Taking the complement of the above expression and rearranging terms gives

$$\mathbb{P}\left[F(\phi_{k,\ell} \mid \phi_k) \leq F^*(\phi_k) + \frac{1+\zeta}{2} \left(F(\phi_k \mid \phi_k) - F^*(\phi_k)\right) \mid \phi_k\right] \geq \frac{1-\zeta}{1+\zeta} > 0, \quad (41)$$

so with some probability greater than or equal to $(1-\zeta)/(1+\zeta) > 0$, the condition to exit the while loop at iteration ℓ of Algorithm (6) is satisfied.

Further, repeatedly calling VRSO-PE within the inner while loop of Algorithm (6) corresponds to drawing *independent* samples of $\phi_{k,\ell}$ conditioned on ϕ_k . The draws are independent because VRSO-PE* is only random due to $I_{k,\lambda}$ after conditioning on ϕ_k , and each iteration of the inner while loop of Algorithm (6) involves drawing independent samples of $I_{k,\lambda}$. As a result, $\ell^*(k)$ follows a geometric distribution with some positive success probability $p_k \geq (1-\zeta)/(1+\zeta)$.

Therefore, $\mathbb{P}\{\ell^*(k) < \infty\} = \sum_{\ell \geq 1} p_k(1-p_k)^{\ell-1} = 1$, and

$$\mathbb{P}\{\forall k : \ell^*(k) < \infty\} = 1 - \mathbb{P}\{\exists k : \ell^*(k) = \infty\} = 1.$$

□

Finally, we introduce $R_{M,\lambda}$ as a point-to-set map corresponding to one iteration of Algorithm (6) with M steps of step size λ . First, recall that

$$K(\phi') = F^*(\phi') + [(1+\zeta)/2] \left(F(\phi' \mid \phi') - F^*(\phi')\right). \quad (42)$$

Then, we define $R_{M,\lambda}(\phi')$ as

$$R_{M,\lambda}(\phi') = \left\{ \phi : \phi = \text{VRSO-PE}^*(I, \lambda, \phi') \text{ for some } I \in \{1, \dots, T\}^M \text{ and } F(\phi \mid \phi') \leq K(\phi') \right\}. \quad (43)$$

Iteration k of Algorithm (6) corresponds to randomly sampling ϕ_{k+1} from $R_{M,\lambda}(\phi_k)$. If the conditions of Theorem 1 are met, then $\mathbb{P}(\ell^*(k) < \infty) = 1$, so $R_{M,\lambda}(\phi_k)$ must not be the empty set. We prove two more useful Lemmas regarding $R_{M,\lambda}$ before finally proving Theorem 1.

Lemma 4. *Suppose all conditions from Theorem 1 hold. Then $R_{M,\lambda}$ is a closed point-to-set map for all non-stationary points $\phi' \in \Phi$. Note that $R_{M,\lambda}$ is closed at a point ϕ' if $\phi'_n \rightarrow \phi'$ and $\phi_n \rightarrow \phi$ with $\phi_n \in R_{M,\lambda}(\phi'_n)$, implies that $\phi \in R_{M,\lambda}(\phi')$.*

Proof. Given a point ϕ' , suppose there exists some sequence $\phi'_n \rightarrow \phi'$ as well as another sequence $\phi_n \rightarrow \phi$ with $\phi_n \in R_{M,\lambda}(\phi'_n)$. To prove the Lemma, we prove that $\phi \in R_{M,\lambda}(\phi')$, i.e. that

1. $F(\phi \mid \phi') \leq K(\phi')$, and
2. $\phi = \text{VRSO-PE}^*(I, \lambda, \phi')$ for some $I \in \{1, \dots, T\}^M$

We start with the first condition. Since $\phi_n \rightarrow \phi$ and $\phi'_n \rightarrow \phi'$, we have that $F(\phi_n \mid \phi'_n) \rightarrow F(\phi \mid \phi')$ because F is continuous by condition (7) of Theorem 1. Likewise, $K(\phi'_n) \rightarrow K(\phi')$ because K is continuous by Lemma 1. Finally, $F(\phi \mid \phi') \leq K(\phi')$ because for all n , $\phi_n \in R_{M,\lambda}(\phi'_n)$, which implies that $F(\phi_n \mid \phi'_n) \leq K(\phi'_n)$. Taking the limit of both sides as $n \rightarrow \infty$ gives the result.

We now show the second condition. By way of contradiction, assume that $\phi \neq \text{VRSO-PE}^*(I, \lambda, \phi')$ for any value of $I \in \{1, \dots, T\}^M$. Then, $\min_I \|\phi - \text{VRSO-PE}^*(I, \lambda, \phi')\|$ is strictly positive. By assumption, for each n , $\phi_n \in R_{M,\lambda}(\phi'_n)$, so $\phi_n = \text{VRSO-PE}^*(I_n, \lambda, \phi'_n)$ for some $I_n \in \{1, \dots, T\}^M$. Using the triangle inequality, we have:

$$0 < \min_I \|\phi - \text{VRSO-PE}^*(I, \lambda, \phi')\| \leq \|\phi_n - \phi\| + \|\phi_n - \text{VRSO-PE}^*(I_n, \lambda, \phi')\| \quad (44)$$

$$= \|\phi_n - \phi\| + \|\text{VRSO-PE}^*(I_n, \lambda, \phi'_n) - \text{VRSO-PE}^*(I_n, \lambda, \phi')\| \quad (45)$$

$$\leq \|\phi_n - \phi\| + \max_I \|\text{VRSO-PE}^*(I, \lambda, \phi'_n) - \text{VRSO-PE}^*(I, \lambda, \phi')\|. \quad (46)$$

However, the right-hand side of the equation converges to zero since $\phi_n \rightarrow \phi$, $\phi'_n \rightarrow \phi'$, and $\text{VRSO-PE}^*(I, \lambda, \phi')$ is continuous for fixed I and λ . This is a contradiction, so it must be that $\phi = \text{VRSO-PE}^*(I, \lambda, \phi')$ for some I . \square

Lemma 5. *Suppose all conditions from Theorem 1 hold, and consider the sequence $\{\phi_k\}_{k=0}^K$ generated from running Algorithm (6) with $\phi_0 \in \Phi$. For $k = 0, \dots, K-1$, we have that $\log p(\mathbf{y}; \phi_{k+1}) \geq \log p(\mathbf{y}; \phi_k)$. Further, if ϕ_k is not a stationary point of $\log p$, then $\log p(\mathbf{y}; \phi_{k+1}) > \log p(\mathbf{y}; \phi_k)$.*

Proof. First, note that ϕ_k is a stationary point of $\log p$ if and only if it is a stationary point of $F(\cdot | \phi_k)$ because $\nabla \log p(\phi_k) = \nabla Q(\phi_k | \phi_k) = -\nabla F(\phi_k | \phi_k)/T$.

We begin with the case where ϕ_k is a stationary point of $\log p$ (and therefore a stationary point of $F(\cdot | \phi_k)$). Then, $F(\phi_k | \phi_k) = F^*(\phi_k)$ since F is continuously differentiable and strongly convex. To exit the while loop at iteration k of Algorithm (6), it must be the case that $F(\phi_{k+1} | \phi_k) \leq F^*(\phi_k) + [(1+\zeta)/2] \left(F(\phi_k | \phi_k) - F^*(\phi_k) \right) = F(\phi_k | \phi_k)$. Finally, we know from Dempster et al. (1977) that $\log p(\mathbf{y}; \phi_{k+1}) - \log p(\mathbf{y}; \phi_k) \geq F(\phi_k | \phi_k) - F(\phi_{k+1} | \phi_k) \geq 0$. This implies that $\log p(\mathbf{y}; \phi_{k+1}) \geq \log p(\mathbf{y}; \phi_k)$.

Next, we focus on the case where ϕ_k is *not* a stationary point of $\log p$ (and therefore not a stationary point $F(\cdot | \phi_k)$). Then, $F(\phi_k | \phi_k) > F^*(\phi_k)$ since F is continuously differentiable and strongly convex. To exit the while loop at iteration k of Algorithm (6), it must be the case that $F(\phi_{k+1} | \phi_k) \leq F^*(\phi_k) + [(1+\zeta)/2] \left(F(\phi_k | \phi_k) - F^*(\phi_k) \right) < F(\phi_k | \phi_k)$. Finally, we know from Dempster et al. (1977) that $\log p(\mathbf{y}; \phi_{k+1}) - \log p(\mathbf{y}; \phi_k) \geq F(\phi_k | \phi_k) - F(\phi_{k+1} | \phi_k) > 0$. This implies that $\log p(\mathbf{y}; \phi_{k+1}) > \log p(\mathbf{y}; \phi_k)$. \square

We now prove Theorem 1.

Proof. The first part of Theorem 1 states that $\mathbb{P}(\ell^*(k) < \infty) = 1$ for all $k \geq 0$ and is true by Lemma 3. The second part of Theorem 1 states that all limit points of $\{\phi_k\}_{k=0}^\infty$ are stationary points of $\log p(\mathbf{y}; \phi)$, and $\log p(\mathbf{y}; \phi_k)$ converges monotonically to $\log p^* = \log p(\mathbf{y}; \phi^*)$ for some stationary point of $\log p$, ϕ^* . This is a direct application of Theorem 1 of Wu (1983), which requires the following conditions:

- (a) $R_{M,\lambda}$ is a closed point-to-set map for all non-stationary points.
- (b) $\log p(\mathbf{y}; \phi_{k+1}) \geq \log p(\mathbf{y}; \phi_k)$ for all $\phi_k \in \Phi$, and $\log p(\mathbf{y}; \phi_{k+1}) > \log p(\mathbf{y}; \phi_k)$ for all $\phi_k \in \Phi$ that are not stationary points of $\log p$.

Condition (a) is satisfied by Lemma 4, and condition (b) is satisfied by Lemma 5. \square

C Proof of Theorem 2

We first give a basic outline of the proof. If $\nabla \log p(\mathbf{y}; \phi_0) = 0$, then $\nabla F(\phi_0 | \phi_0) = 0$, so every gradient step of SAGA and SVRG within VRSO-PE does not change $\phi_0^{(m)}$, so $\phi_0^{(m+1)} = \phi_0^{(m)} = \phi_0^{(0)}$ (we prove this in step 3 of lemma 6). As a result, the gradient approximations do not change either, so $\widehat{\nabla} F^{(m+1)} = \widehat{\nabla} F^{(m)}$ and $\widehat{\nabla} F_t^{(m+1)} = \widehat{\nabla} F_t^{(m)}$ for $t = 1, \dots, T$ (we prove this in steps 4 and 5 of lemma 6). Finally, since the parameter estimates do not change, the conditional probability approximations $\left\{ \widehat{\alpha}_t^{(m)}, \widehat{\beta}_t^{(m)}, \widehat{\gamma}_t^{(m)}, \widehat{\xi}_t^{(m)} \right\}_{t=1}^T = \left\{ \alpha_t(\phi_0), \beta_t(\phi_0), \gamma_t(\phi_0), \xi_t(\phi_0) \right\}_{t=1}^T$ do not change either (we prove this in steps 1 and 2 of lemma 6). Because VRSO-PE returns $\phi_1 = \phi_0^{(M)} = \phi_0$, each iteration of EM-VRSO must do the same, so by induction $\phi_K = \phi_0$ (we prove this in the body of theorem 2).

To prove the Theorem more formally, we first prove a useful Lemma.

Lemma 6. *If:*

1. $\widehat{\alpha}_t^{(0)} = \alpha_t(\phi^{(0)})$ and $\widehat{\beta}_t^{(0)} = \beta_t(\phi^{(0)})$ for all $t = 1, \dots, T$,
2. $\widehat{\gamma}_t^{(0)} = \gamma_t(\phi^{(0)})$ and $\widehat{\xi}_t^{(0)} = \xi_t(\phi^{(0)})$ for all $t = 1, \dots, T$,

$$3. \nabla F(\phi^{(0)} \mid \gamma(\phi^{(0)}), \xi(\phi^{(0)})) = 0,$$

then with probability 1,

$$VRSO-PE\left(\left\{\hat{\alpha}_t^{(0)}, \hat{\beta}_t^{(0)}, \hat{\gamma}_t^{(0)}, \hat{\xi}_t^{(0)}\right\}_{t=1}^T, \phi^{(0)}, \lambda, A, P, M\right) = \phi^{(0)} \quad (47)$$

for all $\lambda \in \mathbb{R}$, $A \in \{SAGA, SVRG\}$, $P \in \{\text{True}, \text{False}\}$, and $M \in \mathbb{N}$.

Proof. We use induction to show that the following conditions hold for all $m \geq 0$:

1. $\hat{\alpha}_t^{(m)} = \alpha_t(\phi^{(0)})$ and $\hat{\beta}_t^{(m)} = \beta_t(\phi^{(0)})$ for all $t = 1, \dots, T$.
2. $\hat{\gamma}_t^{(m)} = \gamma_t(\phi^{(0)})$ and $\hat{\xi}_t^{(m)} = \xi_t(\phi^{(0)})$ for all $t = 1, \dots, T$.
3. $\phi^{(m)} = \phi^{(0)}$
4. $\hat{\nabla} F_t^{(m)} = \nabla F_t(\phi^{(0)} \mid \gamma_t(\phi^{(0)}), \xi_t(\phi^{(0)}))$ for all $t = 1, \dots, T$.
5. $\hat{\nabla} F^{(m)} = 0$

Algorithm (4) returns $\phi^{(M)}$ for some finite M , so proving the above will prove the lemma. The base case ($m = 0$) is trivial either from the assumptions of the Lemma or from lines 1–4 of Algorithm (4), so we focus on the inductive case. We assume that all of the conditions (1–5) above hold for m , and prove each condition holds for $m + 1$ sequentially.

1. First note that if $P = \text{False}$ or if $P = \text{True}$ and $t \neq t_m$, then the conditional probability approximations are not updated, so $\hat{\alpha}_t^{(m+1)} = \hat{\alpha}_t^{(m)} = \alpha_t(\phi^{(0)})$ and $\hat{\beta}_t^{(m+1)} = \hat{\beta}_t^{(m)} = \beta_t(\phi^{(0)})$ by inductive hypothesis (1).

If $P = \text{True}$ and $t = t_m$, then both $\hat{\alpha}_{t_m}^{(m+1)}$ and $\hat{\beta}_{t_m}^{(m+1)}$ are updated in line 9 of Algorithm (5). The approximation $\hat{\alpha}_{t_m}^{(m+1)}$ is updated using the mapping

$$\begin{aligned} \hat{\alpha}_{t_m}^{(m+1)} &= \tilde{\alpha}_{t_m}(\hat{\alpha}_{t_m-1}^{(m)}, \phi^{(m)}) \\ &= \tilde{\alpha}_{t_m}(\alpha_{t_m-1}(\phi^{(0)}), \phi^{(0)}) \\ &= \begin{cases} \delta(\eta^{(0)}) P(y_1; \theta^{(0)}), & \text{for } t_m = 1 \\ \alpha_{t_m-1}(\phi^{(0)}) \Gamma(\eta^{(0)}) P(y_{t_m}; \theta^{(0)}), & \text{for } t_m = 2, \dots, T \end{cases} \\ &= \alpha_{t_m}(\phi^{(0)}) \end{aligned} \quad (48)$$

The second line is true by inductive hypotheses (1) and (3), the third line is the definition of $\tilde{\alpha}_{t_m}$, and the final line is the definition of $\alpha_{t_m}(\phi^{(0)})$. Similar logic can be used to show that

$$\hat{\beta}_{t_m}^{(m+1)} = \beta_{t_m}(\phi^{(0)}). \quad (49)$$

2. First note that if $P = \text{False}$ or if $P = \text{True}$ and $t \neq t_m$, then the conditional probability approximations are not updated, so $\hat{\gamma}_t^{(m+1)} = \hat{\gamma}_t^{(m)} = \gamma_t(\phi^{(0)})$ and $\hat{\xi}_t^{(m+1)} = \hat{\xi}_t^{(m)} = \xi_t(\phi^{(0)})$ by inductive hypothesis (2).

If $P = \text{True}$ and $t = t_m$, both $\hat{\gamma}_{t_m}^{(m+1)}$ and $\hat{\xi}_{t_m}^{(m+1)}$ are calculated in line 10 of Algorithm (5). $\hat{\xi}_{t_m}^{(m+1)}$ is updated using the mapping

$$\begin{aligned} \hat{\xi}_{t_m}^{(m+1)} &= \tilde{\xi}_{t_m}(\hat{\alpha}_{t_m-1}^{(m+1)}, \hat{\beta}_{t_m}^{(m+1)}, \phi^{(m)}) \\ &= \tilde{\xi}_{t_m}(\alpha_{t_m-1}(\phi^{(0)}), \beta_{t_m}(\phi^{(0)}), \phi^{(0)}) \\ &= \frac{\text{diag}(\alpha_{t_m-1}(\phi^{(0)})) \Gamma(\eta^{(0)}) P(y_{t_m}; \theta^{(0)}) \text{diag}(\beta_{t_m}(\phi^{(0)}))}{\alpha_{t_m-1}(\phi^{(0)}) \Gamma(\eta^{(0)}) P(y_{t_m}; \theta^{(0)}) \beta_{t_m}(\phi^{(0)})^\top} \\ &= \xi_{t_m}(\phi^{(0)}). \end{aligned} \quad (50)$$

The second line is true by Equations (48 – 49) and inductive hypothesis (3), the third line is the definition of $\tilde{\xi}_{t_m}$, and the final line is the definition of $\xi_{t_m}(\phi^{(0)})$. Similar logic can be used to show that

$$\hat{\gamma}_{t_m}^{(m+1)} = \gamma_{t_m}(\phi^{(0)}). \quad (51)$$

3. The parameter $\phi^{(m+1)}$ is calculated in line 12 of VRS0-PE as follows:

$$\begin{aligned} \phi^{(m+1)} &= \phi^{(m)} - \lambda \left[\nabla F_{t_m} \left(\phi^{(m)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) - \hat{\nabla} F_{t_m}^{(m)} + \hat{\nabla} F^{(m)} \right] \\ &= \phi^{(0)} - \lambda \left[\nabla F_{t_m} \left(\phi^{(0)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) - \nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right) + 0 \right] \\ &= \phi^{(0)} - \lambda \left[\nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right) - \nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right) \right] \\ &= \phi^{(0)}. \end{aligned}$$

The second line is true by the inductive hypotheses (3), (4), and (5), and the third line is true by Equations (50 – 51).

4. Note that if $A = \text{SVRG}$ or if $A = \text{SAGA}$ and $t \neq t_m$, then the gradient approximation at index t is not updated, so $\hat{\nabla} F_t^{(m+1)} = \hat{\nabla} F_t^{(m)} = \nabla F_t \left(\phi^{(0)} \mid \gamma_t(\phi^{(0)}), \xi_t(\phi^{(0)}) \right)$ by inductive hypothesis (4).

If $A = \text{SAGA}$ and $t = t_m$, then $\hat{\nabla} F_{t_m}^{(m+1)}$ is calculated in line 14 of VRS0-PE as follows:

$$\begin{aligned} \hat{\nabla} F_{t_m}^{(m+1)} &= \nabla F_{t_m} \left(\phi^{(m)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) \\ &= \nabla F_{t_m} \left(\phi^{(0)} \mid \hat{\gamma}_{t_m}^{(m+1)}, \hat{\xi}_{t_m}^{(m+1)} \right) \\ &= \nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right). \end{aligned} \quad (52)$$

The second line is true by inductive hypothesis (3), and the third line is true by Equations (50 – 51).

5. If $A = \text{SVRG}$, then the full gradient approximation is not updated, so $\hat{\nabla} F^{(m+1)} = \hat{\nabla} F^{(m)} = 0$ by inductive hypothesis (5).

If $A = \text{SAGA}$, then $\hat{\nabla} F^{(m+1)}$ is calculated in line 14 of VRS0-PE as follows:

$$\begin{aligned} \hat{\nabla} F^{(m+1)} &= \hat{\nabla} F^{(m)} + \frac{1}{T} \left(\hat{\nabla} F_{t_m}^{(m+1)} - \hat{\nabla} F_{t_m}^{(m)} \right) \\ &= 0 + \frac{1}{T} \left[\hat{\nabla} F_{t_m}^{(m+1)} - \nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right) \right] \\ &= \frac{1}{T} \left[\nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right) - \nabla F_{t_m} \left(\phi^{(0)} \mid \gamma_{t_m}(\phi^{(0)}), \xi_{t_m}(\phi^{(0)}) \right) \right] \\ &= 0. \end{aligned}$$

The second line is true by inductive hypotheses (4) and (5), and the third line is true by Equation (52).

This completes the inductive step, and proves the lemma. \square

We now prove Theorem 2.

Proof. We use induction and show that if $\nabla \log p(\mathbf{y}; \phi_0) = 0$, then $\phi_k = \phi_0$ for all iterations k of Algorithm (5). The base case, $k = 0$, is trivial. For the inductive case, we assume that $\phi_k = \phi_0$ and prove that $\phi_{k+1} = \phi_k$. First, note that

$$\begin{aligned}
\nabla F(\phi_0 \mid \gamma(\phi_0), \xi(\phi_0)) &= \frac{1}{T} \sum_{t=1}^T \nabla F_t(\phi_0 \mid \gamma_t(\phi_0), \xi_t(\phi_0)) \\
&= -\frac{1}{T} \nabla Q(\phi_0 \mid \phi_0) \\
&= -\frac{1}{T} \nabla \log p(\mathbf{y}; \phi_0) \\
&= 0.
\end{aligned} \tag{53}$$

Equation (53) shows that the condition for Lemma 6 are satisfied with $\phi^{(0)} = \phi_0$. Further, step 2 of Algorithm (5) performs the E step of the EM algorithm so that $\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T = \{\alpha_t(\phi_k), \beta_t(\phi_k), \gamma_t(\phi_k), \xi_t(\phi_k)\}_{t=1}^T$. So:

$$\begin{aligned}
\phi_{k,1} &= \text{VRSO-PE}\left(\{\alpha_{k,t}, \beta_{k,t}, \gamma_{k,t}, \xi_{k,t}\}_{t=1}^T, \phi_k, \lambda, A, P, M\right) \\
&= \text{VRSO-PE}\left(\{\alpha_t(\phi_k), \beta_t(\phi_k), \gamma_t(\phi_k), \xi_t(\phi_k)\}_{t=1}^T, \phi_k, \lambda, A, P, M\right) \\
&= \text{VRSO-PE}\left(\{\alpha_t(\phi_0), \beta_t(\phi_0), \gamma_t(\phi_0), \xi_t(\phi_0)\}_{t=1}^T, \phi_0, \lambda, A, P, M\right) \\
&= \phi_0.
\end{aligned}$$

The second line is true by the definition of the E step in the EM algorithm, the third line is true by the inductive hypothesis, and the final line is true by Lemma 6. Since $\log p(\mathbf{y}; \phi_{k,1}) = \log p(\mathbf{y}; \phi_k) = \log p(\mathbf{y}; \phi_0)$, $\phi_{k,1}$ satisfies the condition to exit the while loop of Algorithm (5), so $\phi_{k+1} = \phi_{k,1} = \phi_0$.

This completes the inductive step, so $\phi_k = \phi_0$ for all $k \geq 0$. To complete the proof, note that Algorithm (5) returns ϕ_K , so $\text{EM-VRSO}(\phi_0, \lambda, A, M, K) = \phi_K = \phi_0$. \square

VARIANCE-REDUCED STOCHASTIC OPTIMIZATION FOR EFFICIENT INFERENCE OF HIDDEN MARKOV MODELS: SUPPLEMENT A: SIMULATION STUDY

Evan Sidrow
Department of Statistics
University of British Columbia
Vancouver, Canada
evan.sidrow@stat.ubc.ca

Nancy Heckman
Department of Statistics
University of British Columbia
Vancouver, Canada

Alexandre Bouchard-Côté
Department of Statistics
University of British Columbia
Vancouver, Canada

Sarah M. E. Fortune
Department of Oceanography
Dalhousie University
Halifax, Canada

Andrew W. Trites
Department of Zoology
Institute for the Oceans and Fisheries
University of British Columbia
Vancouver, Canada

Marie Auger-Méthé
Department of Statistics
Institute for the Oceans and Fisheries
University of British Columbia
Vancouver, Canada

1 Simulation study results, $T = 10^3$

Below are results for the experiments with $T = 10^3$. They are analogous to Figure (3) and Figure (4) in the main text.

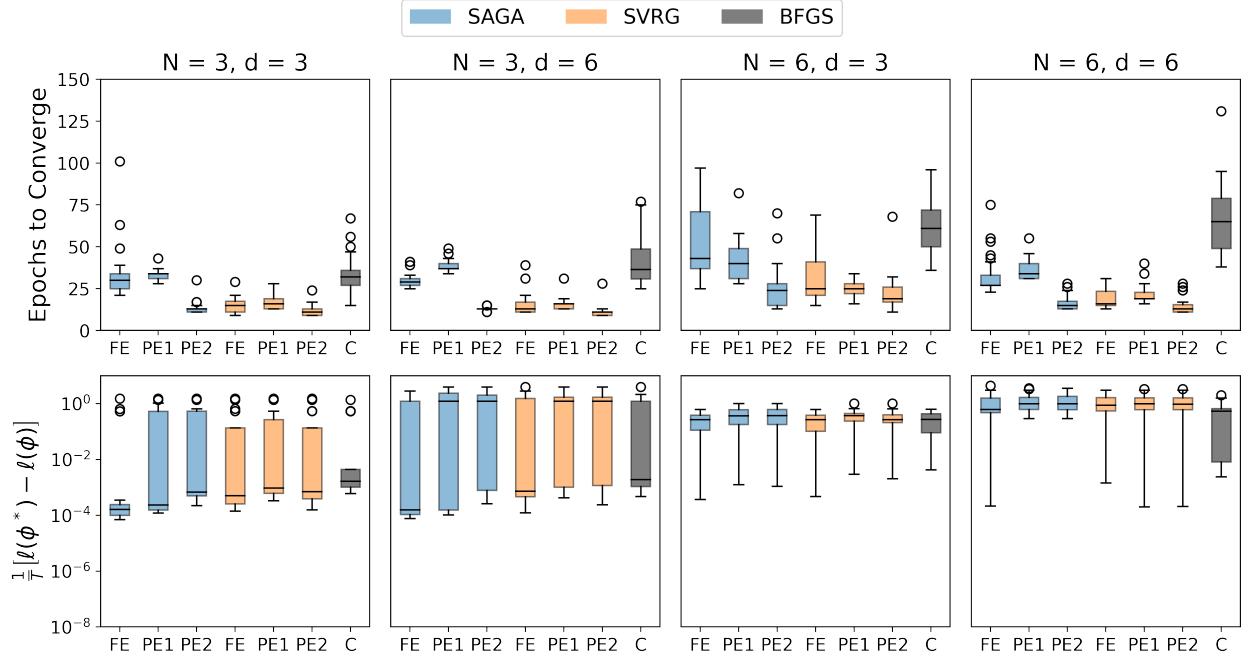


Figure 1: Box plots showing epochs to converge (top) and the maximum log-likelihood minus the log-likelihood at convergence (all divided by T , bottom) for each optimization algorithm. FE corresponds to $\{P = \text{False}, M = T\}$, PE1 corresponds to $\{P = \text{True}, M = T\}$, and PE2 corresponds to $\{P = \text{False}, M = 10T\}$. Blue corresponds to SAGA and orange corresponds to SVRG. Results are shown for all simulation studies with $T = 10^3$, $N = 3$ and $d = 3$ (left), $N = 3$ and $d = 6$ (left-middle), $N = 6$ and $d = 3$ (right-middle), and $N = 6$ and $d = 6$ (right). One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The y-axis of the bottom row is on a log-scale.

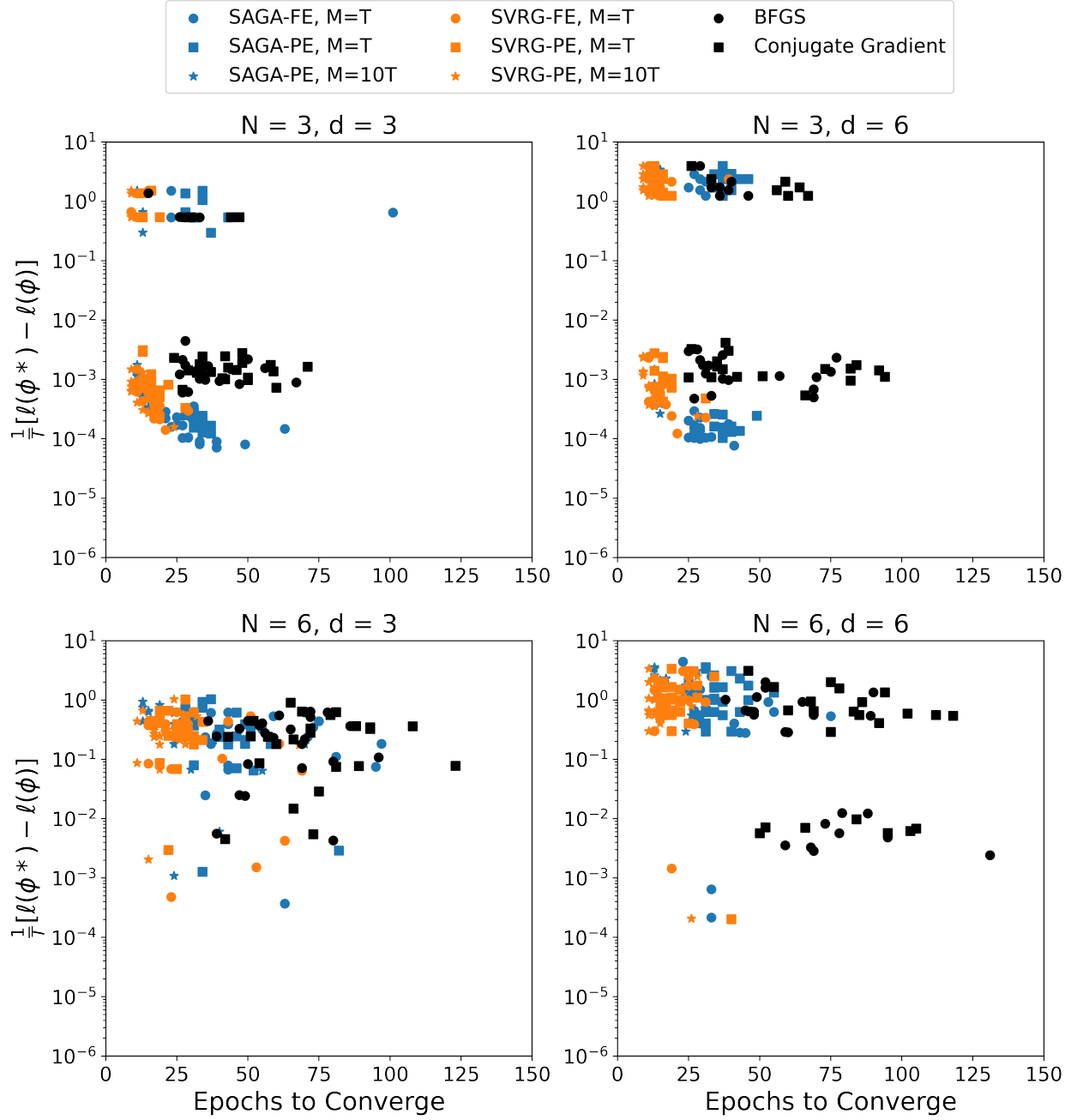


Figure 2: Maximum log-likelihood minus the log-likelihood at convergence (all over T) versus epochs to converge for all simulation studies with $T = 10^3$, $N = 3$ and $d = 3$ (top left), $N = 3$ and $d = 6$ (top right), $N = 6$ and $d = 3$ (bottom left), and $N = 6$ and $d = 6$ (bottom left). One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

The figures below show the optimality gap vs epoch for each data set of every experiment with $T = 10^3$ for all nine tested algorithms. They are similar to Figure (2) from the main text. EM-VRSO tended to converge more quickly than BFGS for experiments with $N = 3$. BFGS often out-performed EM-VRSO in the long-run (after approximately 100 epochs), but BFGS is a second-order method and this behaviour is expected in the long run on small data sets. Our method is particularly well-suited to fitting HMMs on large and complex data sets.

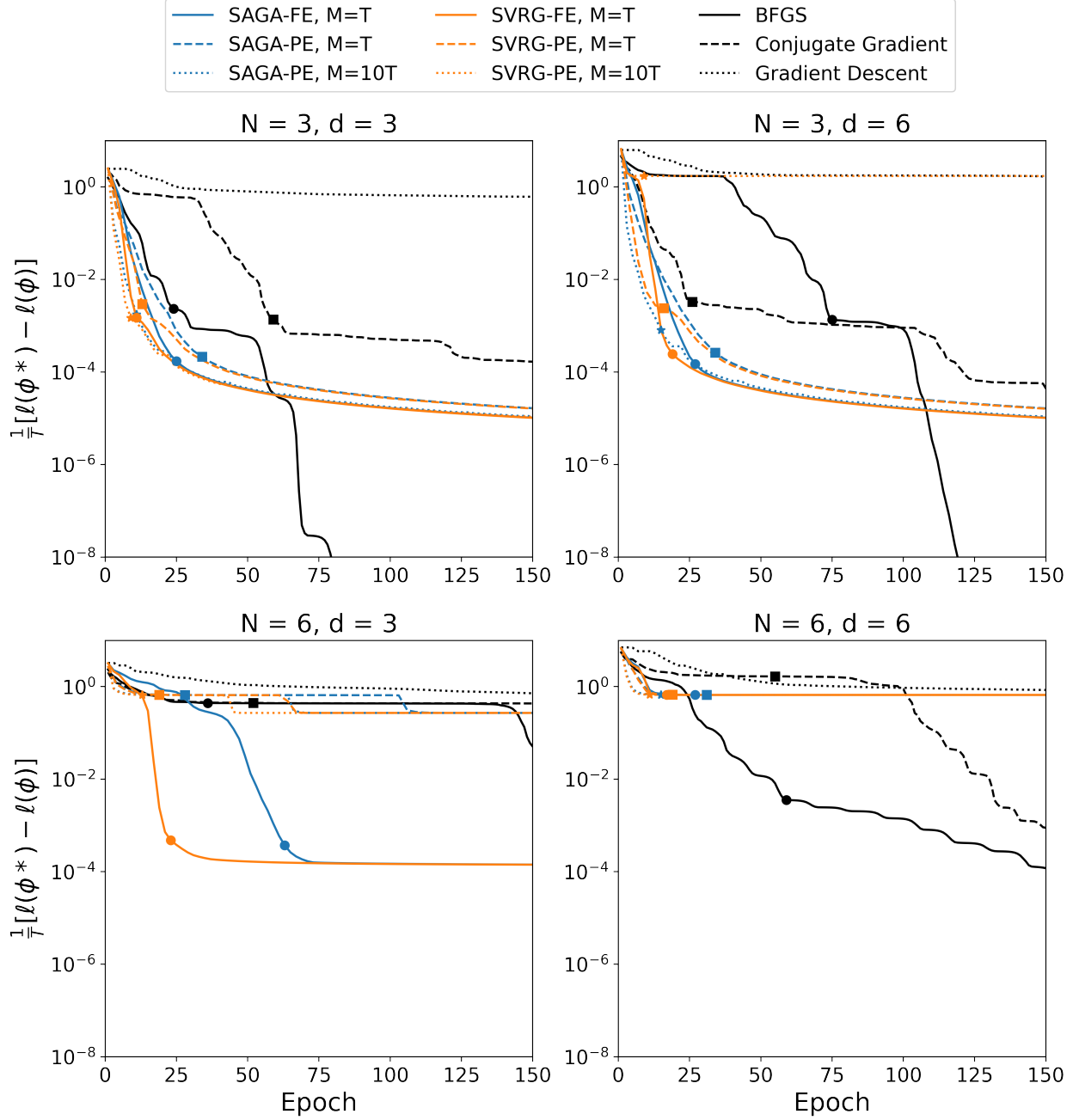


Figure 3: Optimality gap between the current log-likelihood and optimal log-likelihood for the first simulated data sets associated with experiments $T = 10^3$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

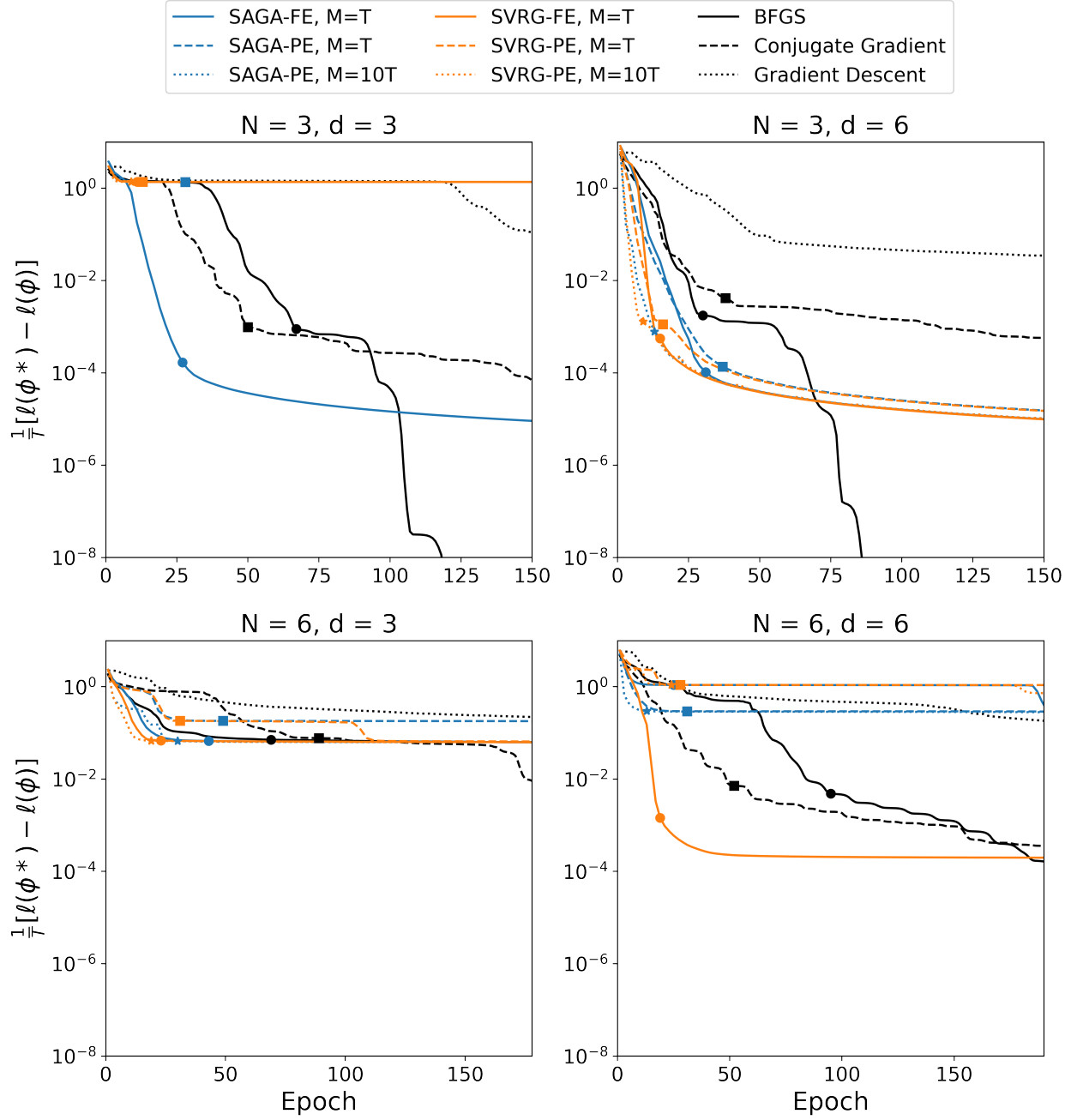


Figure 4: Optimally gap between the current log-likelihood and optimal log-likelihood for the second simulated data sets associated with experiments $T = 10^3$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

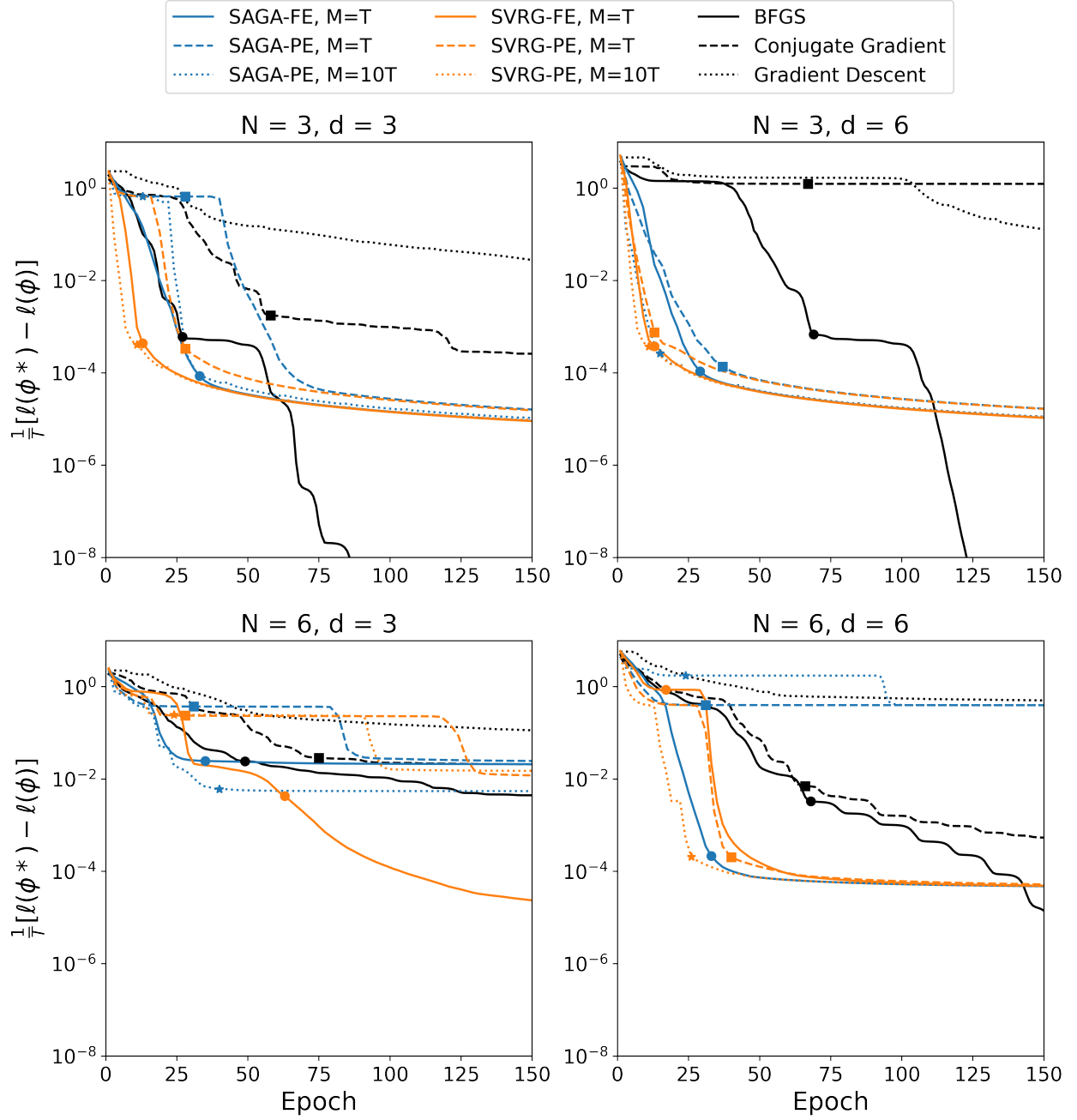


Figure 5: Optimality gap between the current log-likelihood and optimal log-likelihood for the third simulated data sets associated with experiments $T = 10^3$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

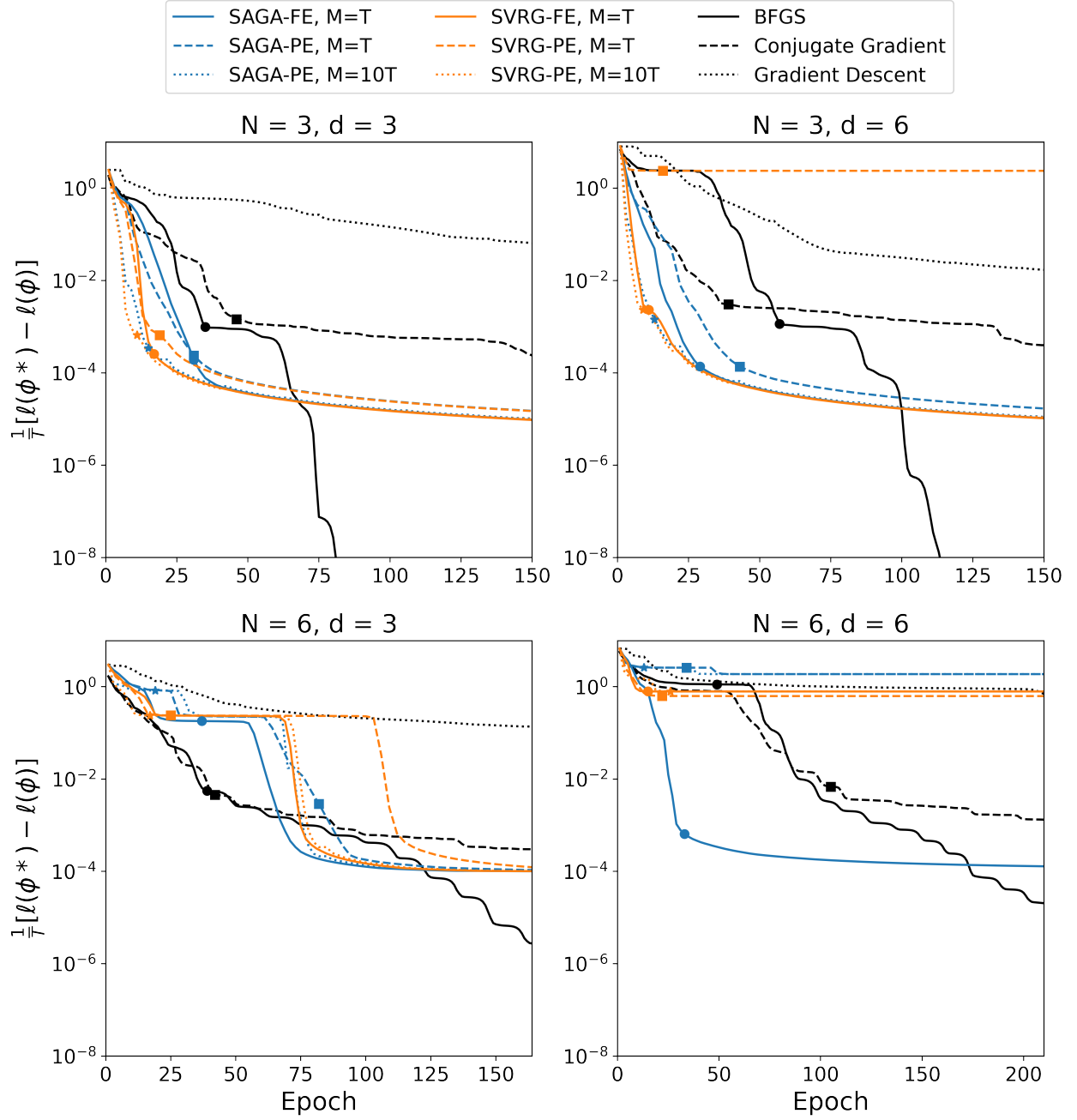


Figure 6: Optimality gap between the current log-likelihood and optimal log-likelihood for the fourth simulated data sets associated with experiments $T = 10^3$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

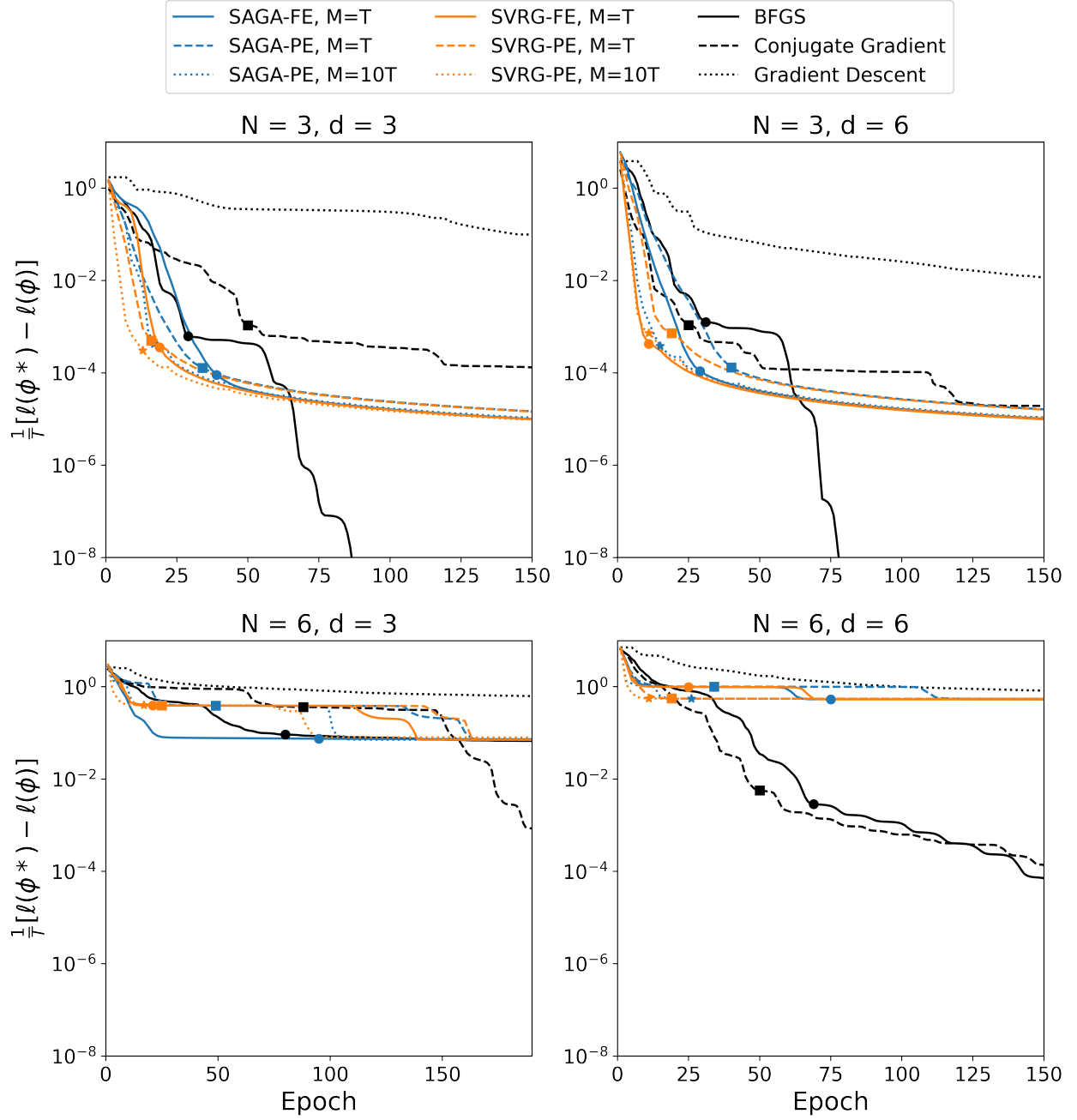


Figure 7: Optimally gap between the current log-likelihood and optimal log-likelihood for the fifth simulated data sets associated with experiments $T = 10^3$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

2 Simulation study results, $T = 10^5$

The figures below are similar to Figure (2) from the simulation study in the main text. They show the optimality gap vs epoch for each experiment and each of the nine tested algorithms. However, we include all simulated data sets here.

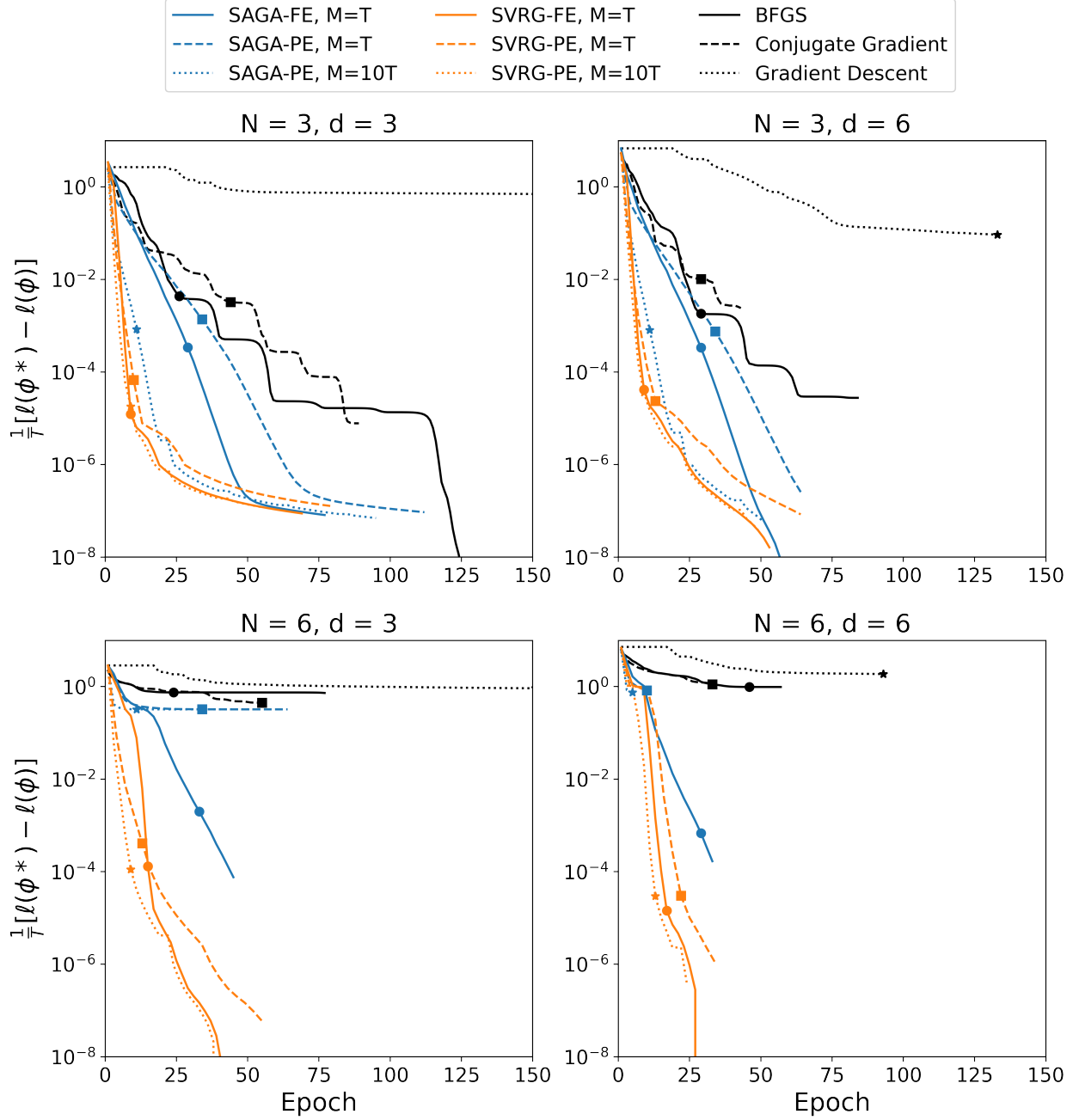


Figure 8: Optimality gap between the current log-likelihood and optimal log-likelihood for the first simulated data sets associated with experiments $T = 10^5$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

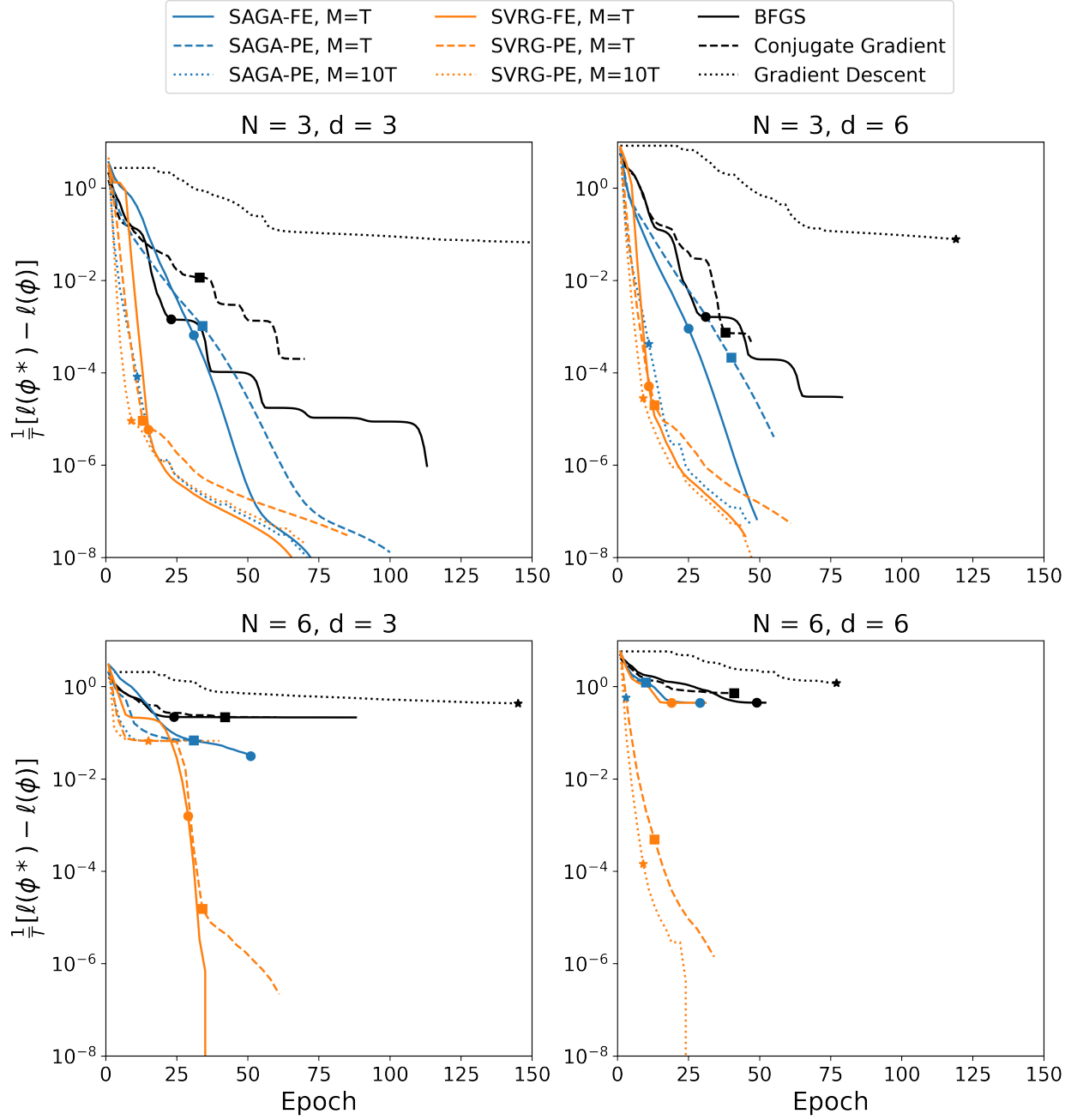


Figure 9: Optimality gap between the current log-likelihood and optimal log-likelihood for the second simulated data sets associated with experiments $T = 10^5$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

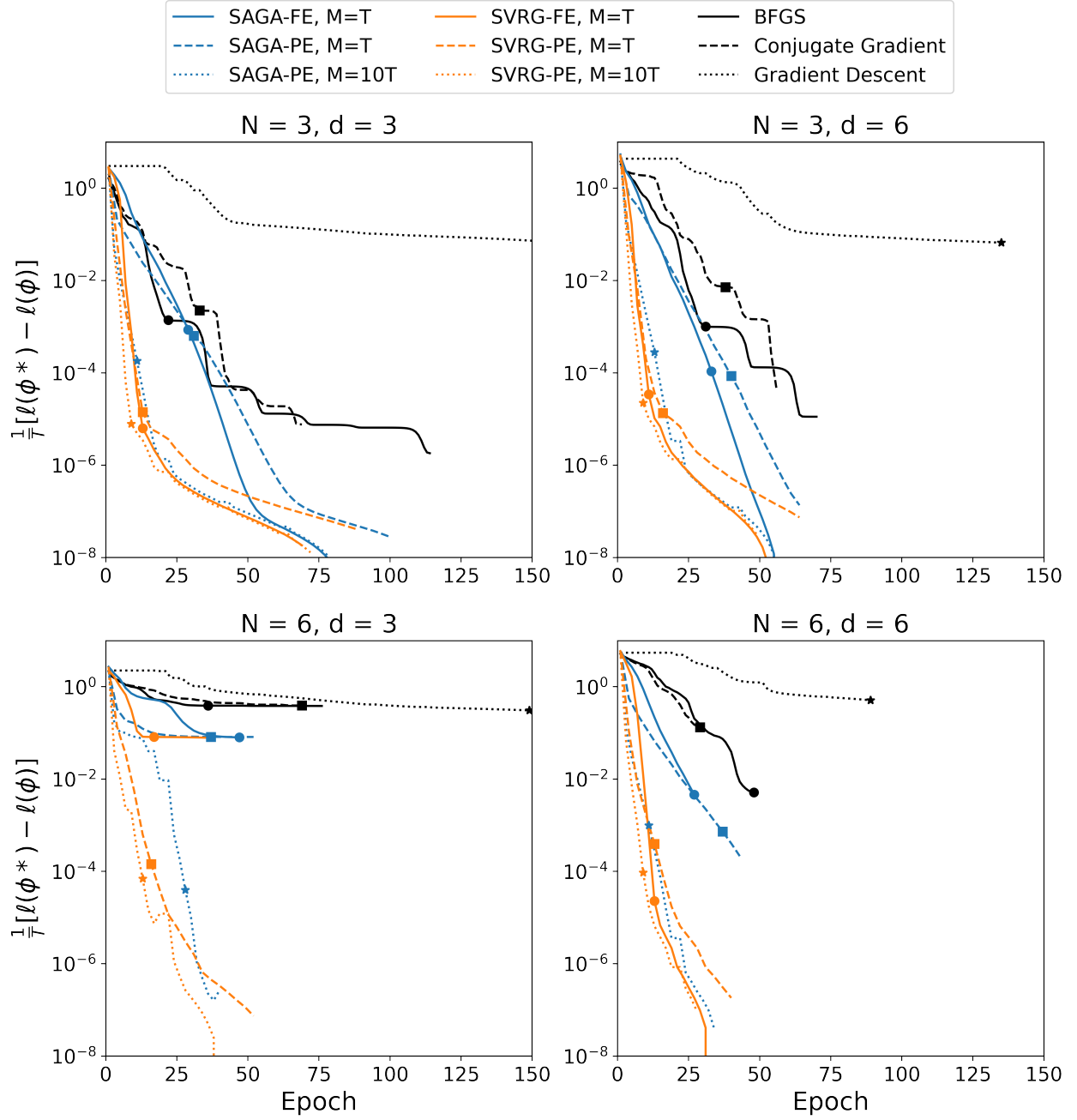


Figure 10: Optimality gap between the current log-likelihood and optimal log-likelihood for the third simulated data sets associated with experiments $T = 10^5$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

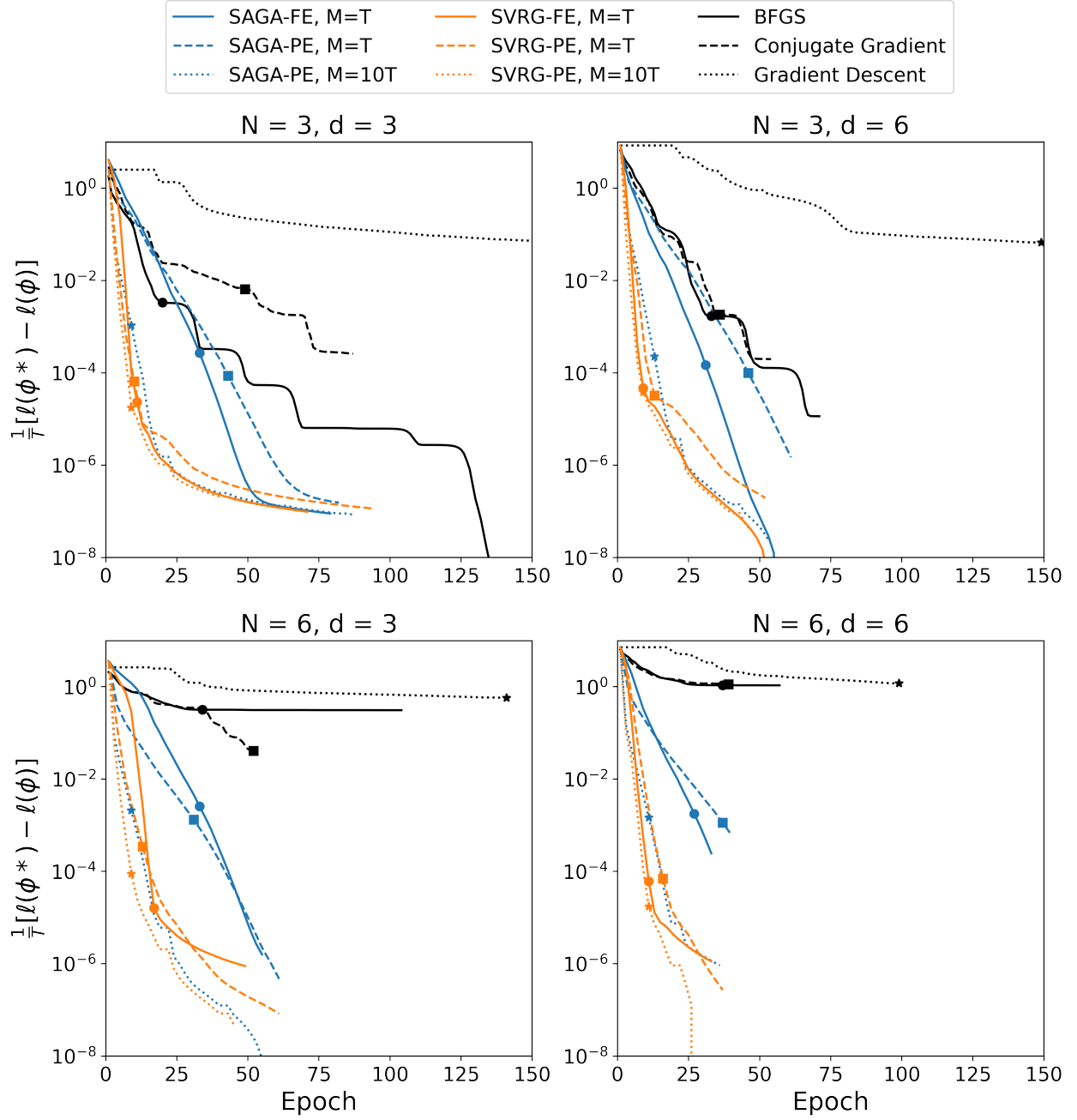


Figure 11: Optimality gap between the current log-likelihood and optimal log-likelihood for the fourth simulated data sets associated with experiments $T = 10^5$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

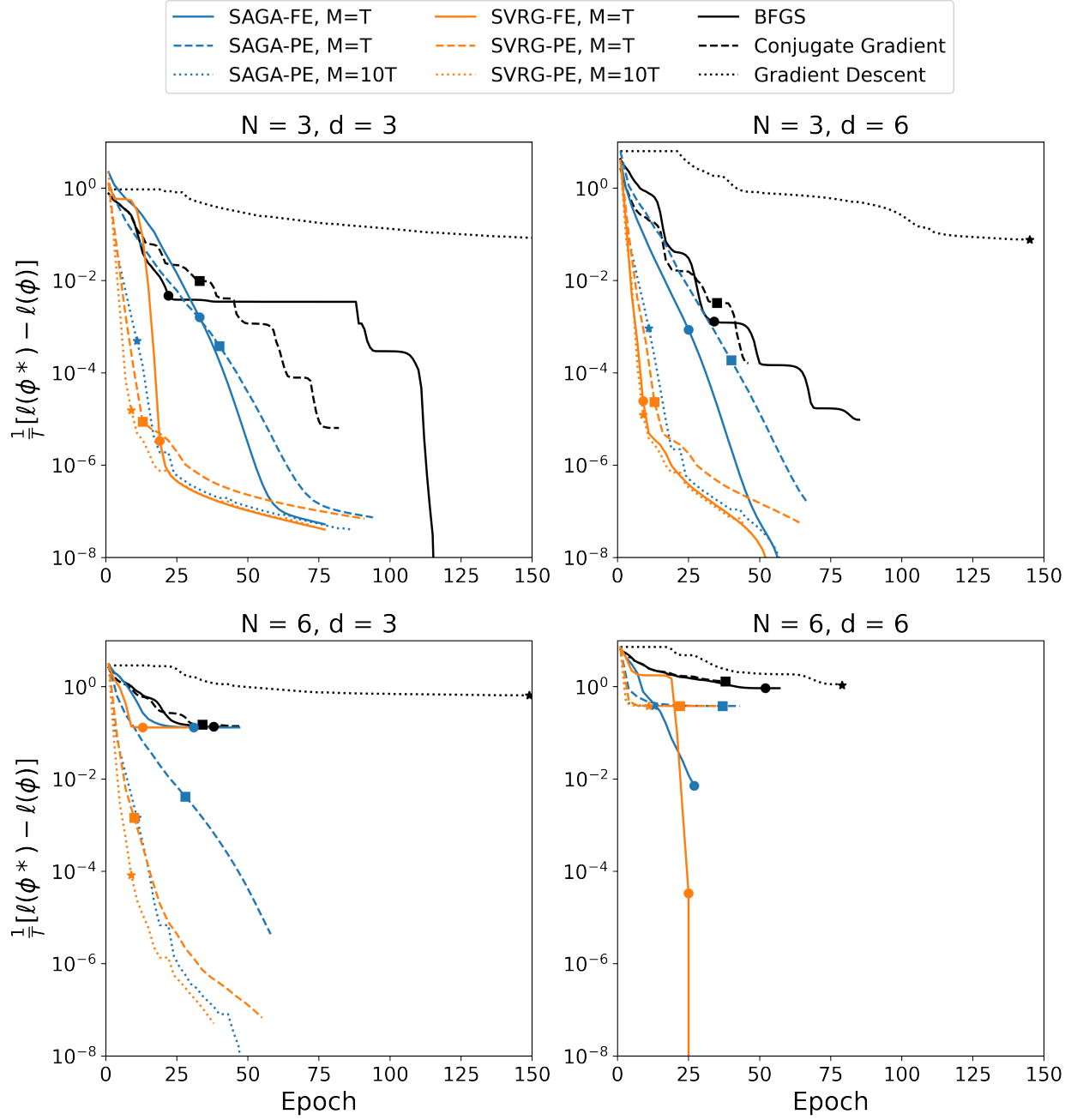


Figure 12: Optimality gap between the current log-likelihood and optimal log-likelihood for the fifth simulated data sets associated with experiments $T = 10^5$, $N \in \{3, 6\}$ and $d \in \{3, 6\}$. One epoch represents either one full E step, T iterations with the M step, or one gradient step for full-gradient algorithms. The log-likelihood of ϕ is denoted as $\ell(\phi)$ on the y-axis, which is on a log-scale.

VARIANCE-REDUCED STOCHASTIC OPTIMIZATION FOR EFFICIENT INFERENCE OF HIDDEN MARKOV MODELS: SUPPLEMENT B: CASE STUDY

Evan Sidrow
Department of Statistics
University of British Columbia
Vancouver, Canada
evan.sidrow@stat.ubc.ca

Nancy Heckman
Department of Statistics
University of British Columbia
Vancouver, Canada

Alexandre Bouchard-Côté
Department of Statistics
University of British Columbia
Vancouver, Canada

Sarah M. E. Fortune
Department of Oceanography
Dalhousie University
Halifax, Canada

Andrew W. Trites
Department of Zoology
Institute for the Oceans and Fisheries
University of British Columbia
Vancouver, Canada

Marie Auger-Méthé
Department of Statistics
Institute for the Oceans and Fisheries
University of British Columbia
Vancouver, Canada

1 Case study parameter results

The parameters presented here come from a run of EM-VRSO with $A = \text{SAGA}$, $P = \text{True}$, and $M = 10T$. They represent the parameters with the highest log-likelihood among all random initializations and optimization algorithms in the case study from the main text.

Dive Type	Parameter Estimate	Descent	Bottom	Ascent
1	$\hat{\mu}$	0.70	0.02	-0.67
	$\hat{\sigma}$	0.45	0.23	0.40
	\hat{p}	0	0	0.12
2	$\hat{\mu}$	0.33	0.00	-0.44
	$\hat{\sigma}$	0.15	0.15	0.21
	\hat{p}	0	0	0.47
3	$\hat{\mu}$	2.71	0.01	-2.50
	$\hat{\sigma}$	1.85	0.64	1.58
	\hat{p}	0	0	0.04

Table 1: Maximum likelihood parameter estimates from the Killer Whale case study described in the main text. Recall that D_t represents the change in depth in meters at time index t and $E_t \in \{0, 1\}$ encodes whether a dive ends at time index t . The parameter μ corresponds to the state-dependent mean of D_t , σ corresponds to the state-dependent standard deviation of D_t , and p corresponds to the state-dependent probability that $E_t = 1$.

Figure (1) below shows the estimated densities of change in depth D_t for all dive types and dive durations.

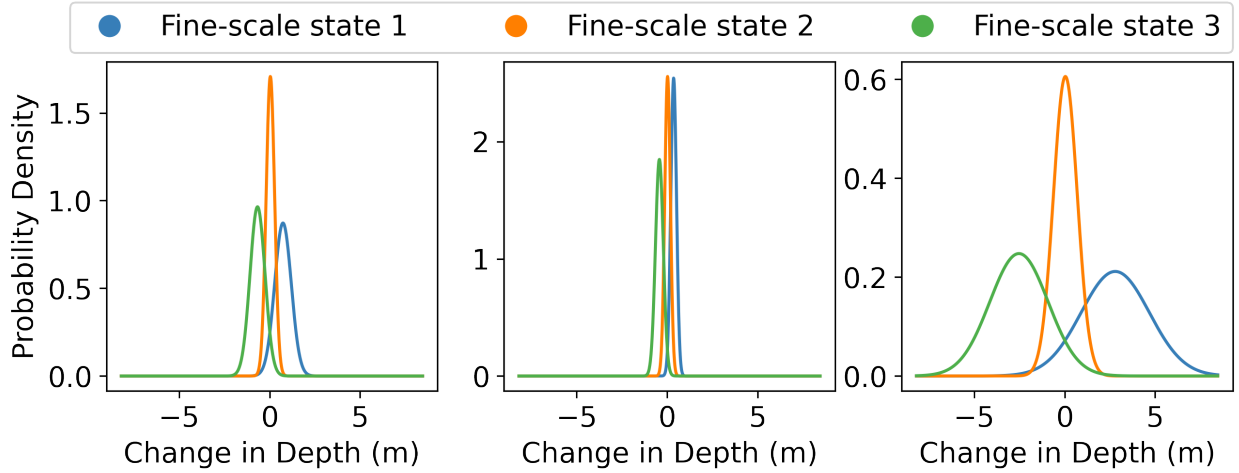


Figure 1: State-dependent density of change in depth for dive types 1, 2, and 3 (left to right). Fine-scale state 1 corresponds to descent, fine-scale state 2 corresponds to bottom, and fine-scale state 3 corresponds to ascent. Dive type 1 corresponds to medium-depth dives, dive type 2 corresponds to shallow dives, and dive type 3 corresponds to deep dives.

The coarse-scale, inter-dive transition probability matrix and initial distribution estimates were

$$\hat{\mathbf{T}}^{(c)} = \begin{pmatrix} 0.223 & 0.758 & 0.017 \\ 0.123 & 0.853 & 0.027 \\ 0.074 & 0.857 & 0.068 \end{pmatrix}, \quad (1)$$

$$\hat{\boldsymbol{\delta}}^{(c)} = (0.258 \quad 0.002 \quad 0.740). \quad (2)$$

The long-run proportion of each dive type is found by solving for the stationary distribution of $\hat{\mathbf{\Gamma}}^{(c)}$, i.e. solving the equation $\hat{\boldsymbol{\pi}}^{(c)} \hat{\mathbf{\Gamma}}^{(c)} = \hat{\boldsymbol{\pi}}^{(c)}$. Doing so gives the following:

$$\hat{\boldsymbol{\pi}}^{(c)} = (0.136 \quad 0.840 \quad 0.024). \quad (3)$$

In words, approximately 13.6% of all dives are of type 1 (medium-depth), 84.0% of all dives are of type 2 (shallow), and 2.4% of all dives are of type 3 (deep).

The fine-scale, intra-dive transition probability matrices and initial distributions were estimated as

$$\hat{\mathbf{\Gamma}}^{(f,1)} = \begin{pmatrix} 0.871 & 0.123 & 0.006 \\ \mathbf{0} & 0.962 & 0.038 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}, \quad \hat{\mathbf{\Gamma}}^{(f,2)} = \begin{pmatrix} 0.668 & 0.304 & 0.028 \\ \mathbf{0} & 0.788 & 0.212 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}, \quad (4)$$

$$\hat{\mathbf{\Gamma}}^{(f,3)} = \begin{pmatrix} 0.958 & 0.040 & 0.002 \\ \mathbf{0} & 0.982 & 0.018 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}, \quad (5)$$

$$\hat{\boldsymbol{\delta}}^{(f,1)} = (\mathbf{1} \quad \mathbf{0} \quad \mathbf{0}), \quad \hat{\boldsymbol{\delta}}^{(f,2)} = (\mathbf{1} \quad \mathbf{0} \quad \mathbf{0}), \quad \hat{\boldsymbol{\delta}}^{(f,3)} = (\mathbf{1} \quad \mathbf{0} \quad \mathbf{0}) \quad (6)$$

Recall that each dive must begin with descent, so all initial distributions are identical. In addition, note that the Markov chains defined by $\hat{\mathbf{\Gamma}}^{(f,i)}$ above are not ergodic, so stationary distributions do not exist for the fine-scale, intra-dive Markov chains.

2 Case study decoded dive profiles

The figures below show approximately one hour of dive profiles associated with each of the eight whales used to train the hierarchical HMM in the case study.



Figure 2: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale A100. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

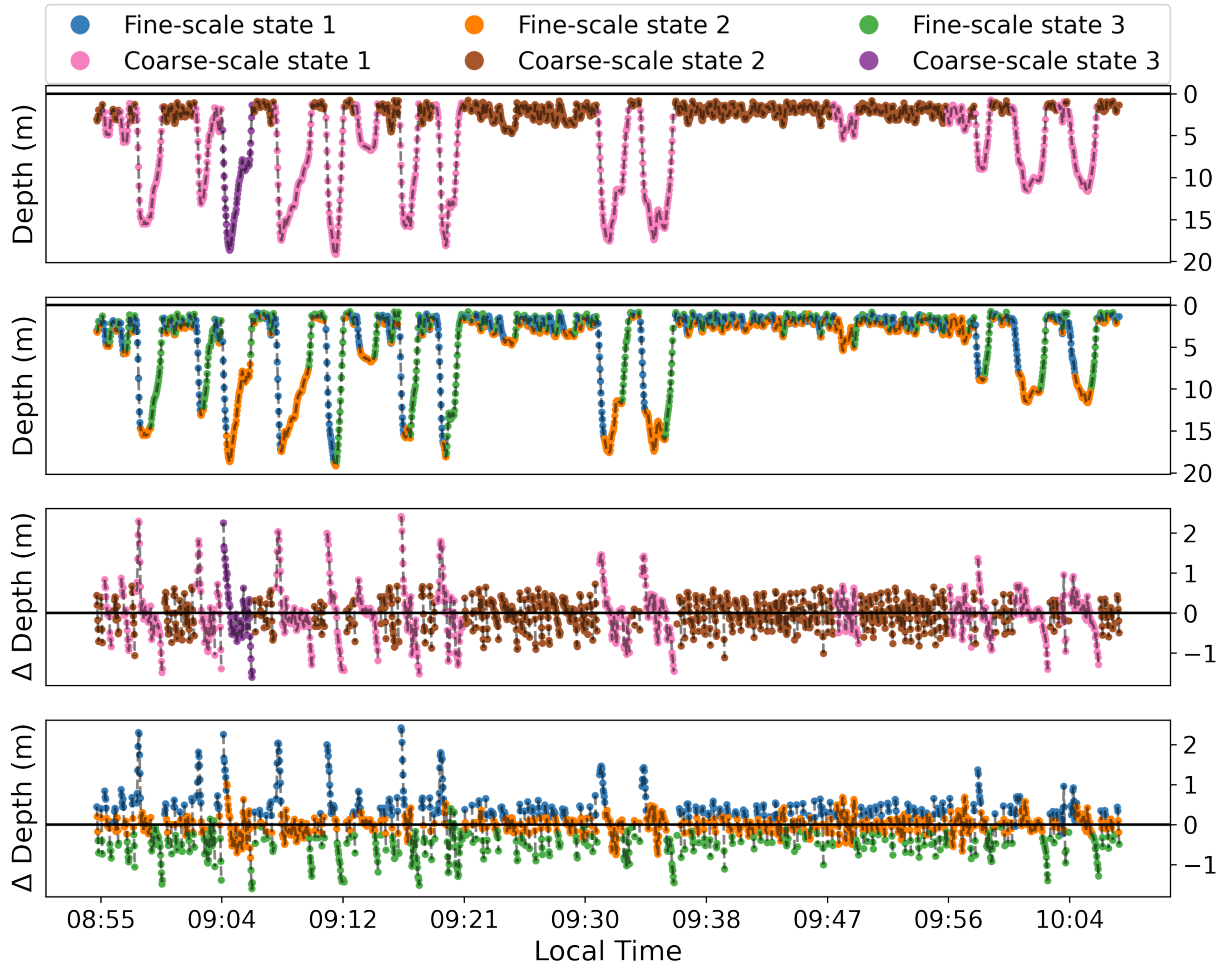


Figure 3: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale A113. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

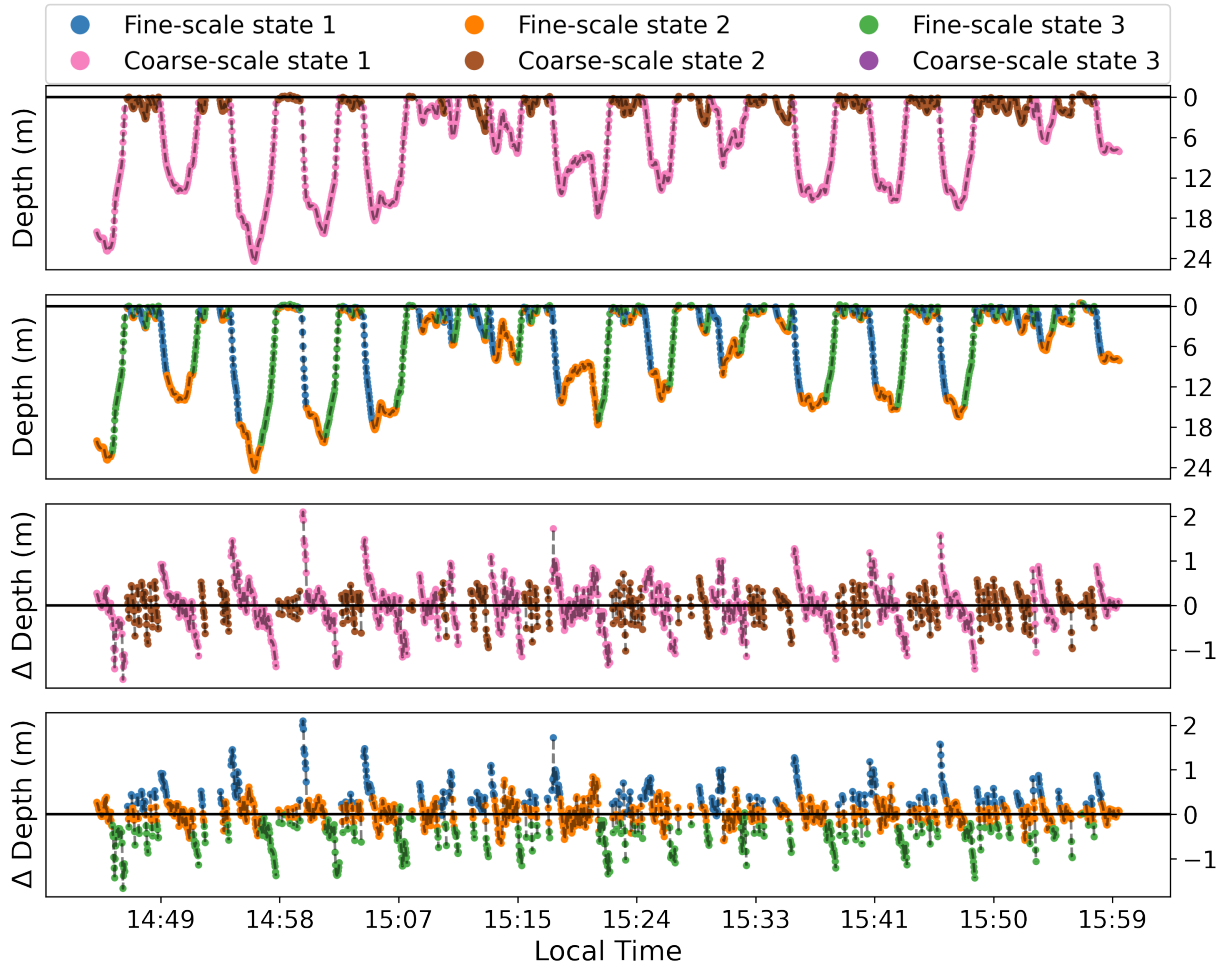


Figure 4: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale D21. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

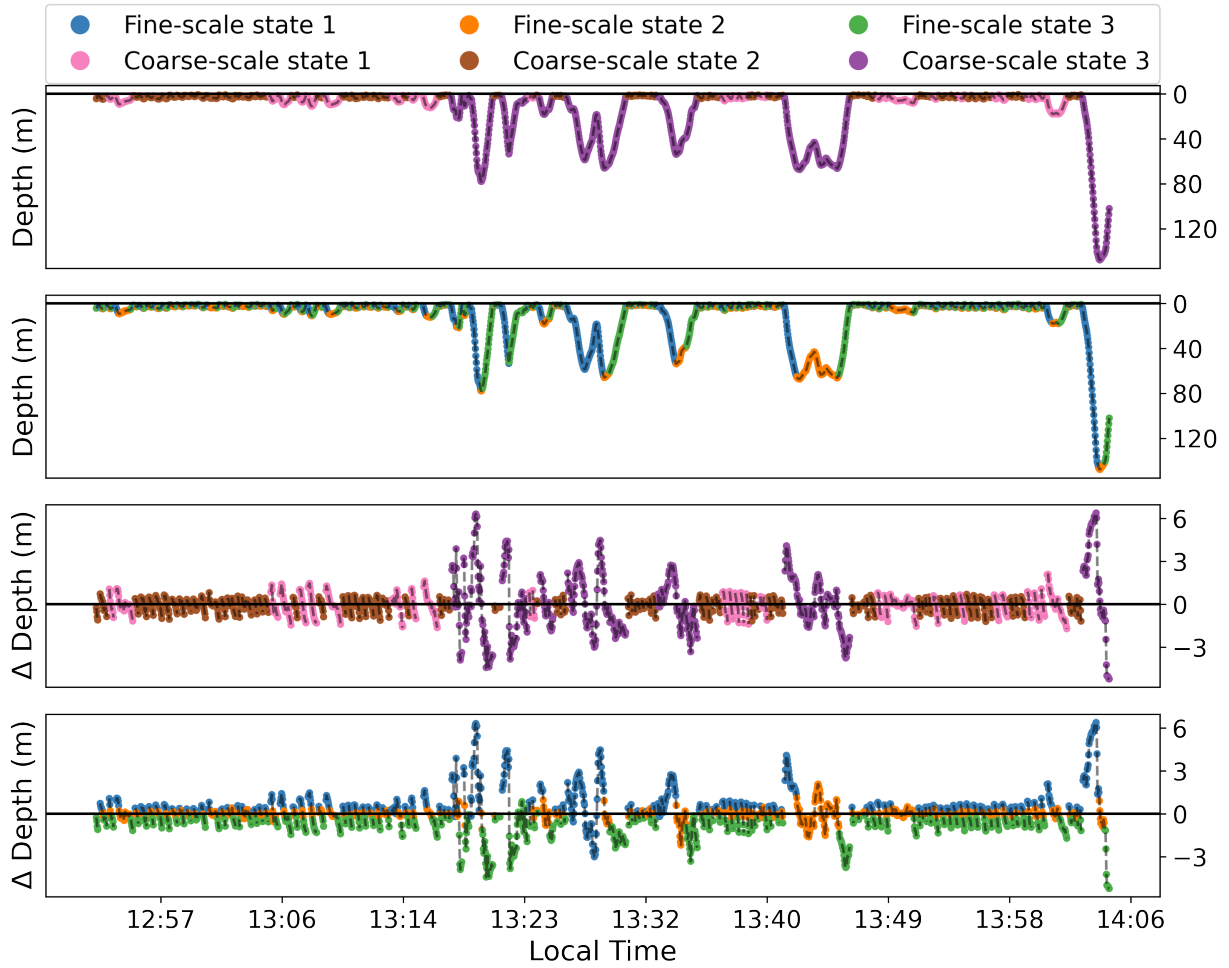


Figure 5: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale D26. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

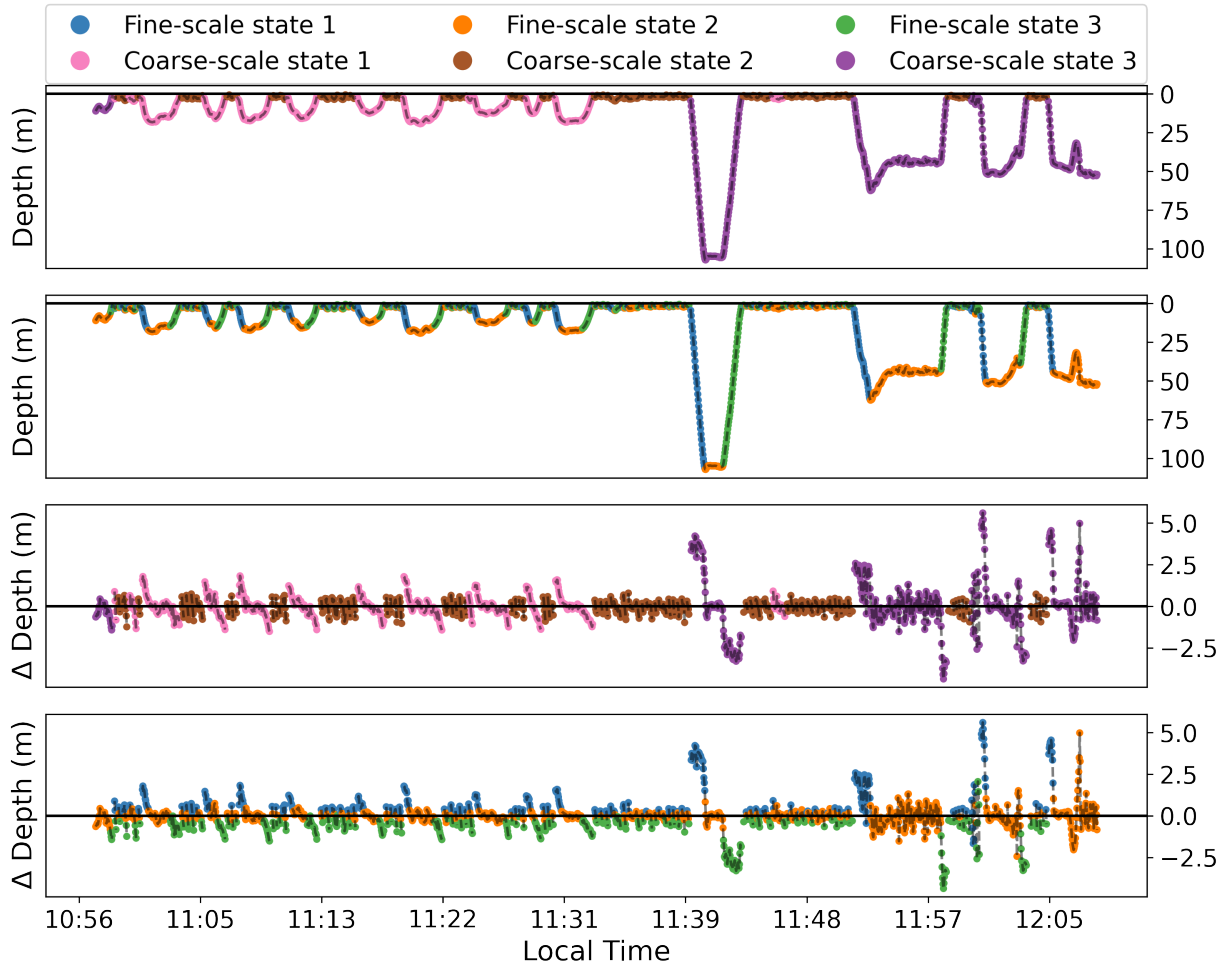


Figure 6: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale I107. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

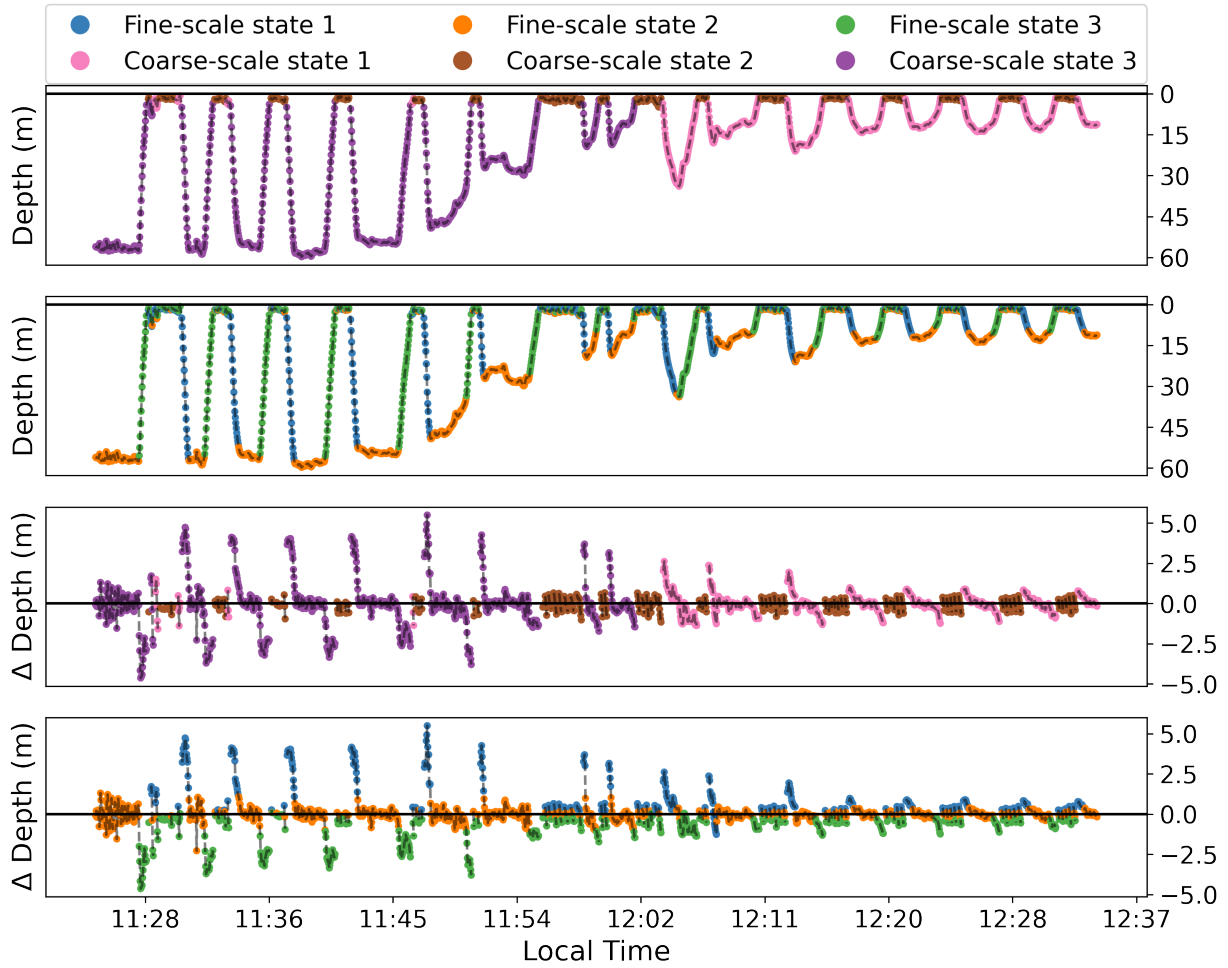


Figure 7: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale I145. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

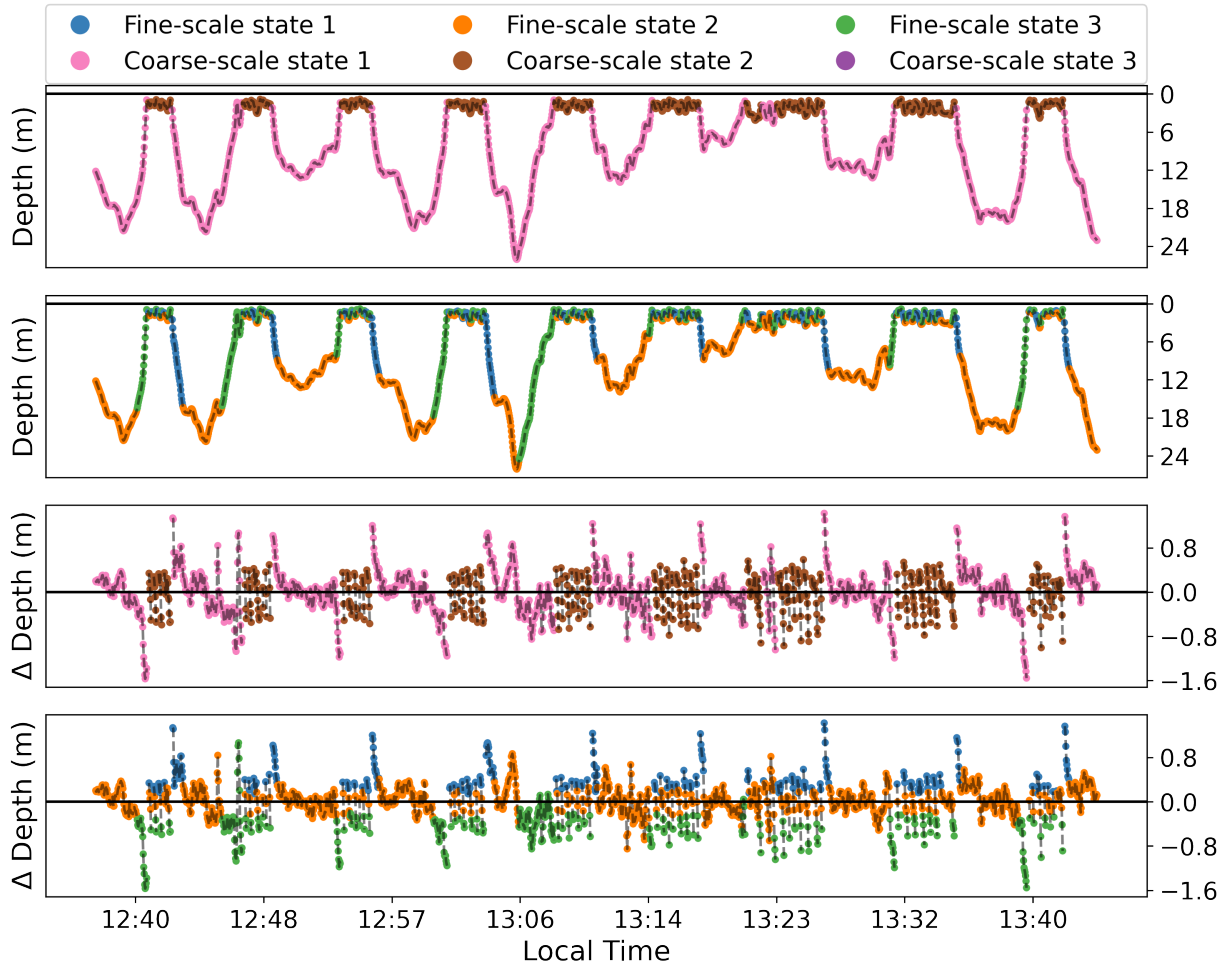


Figure 8: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale R48. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).

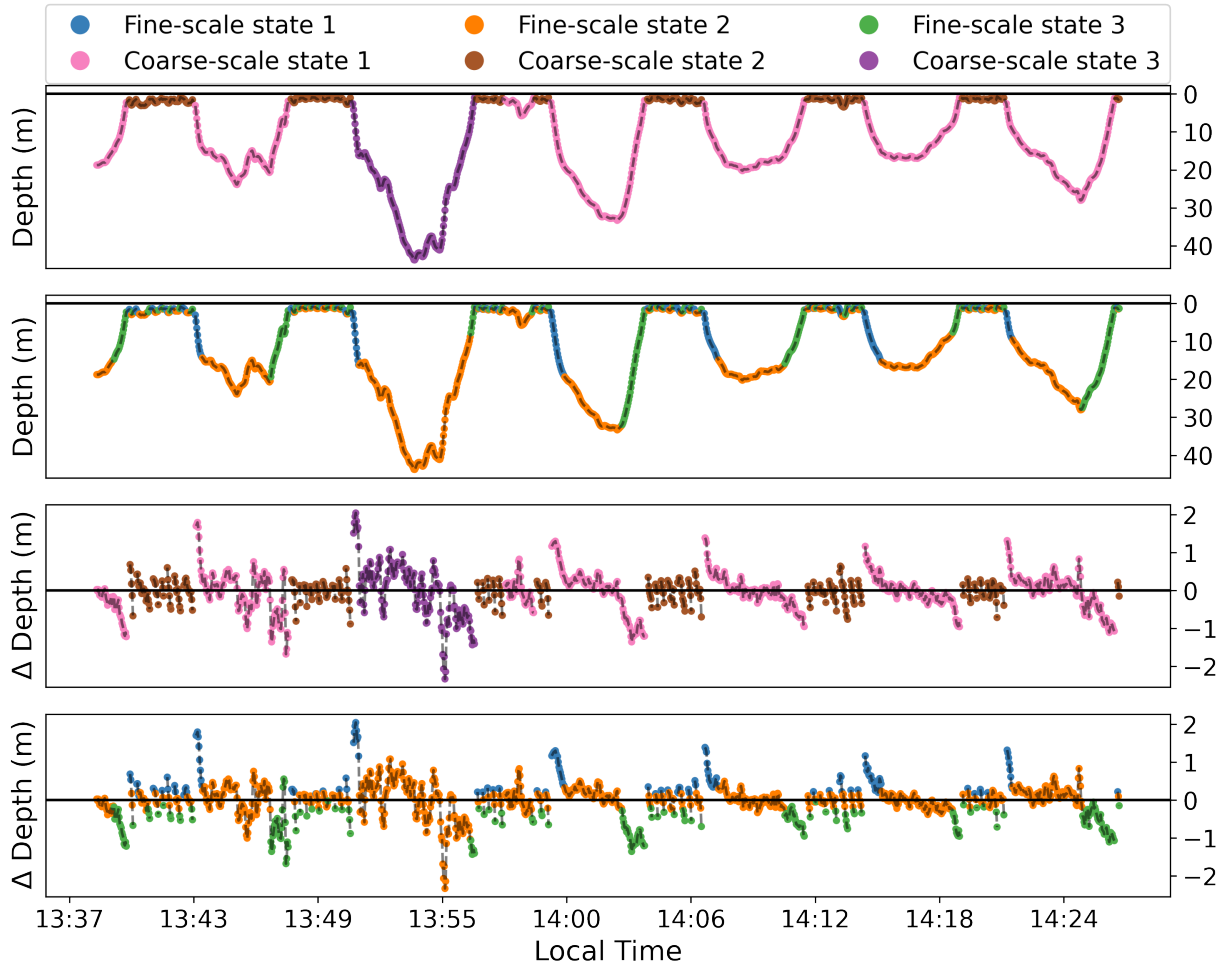


Figure 9: Dive profiles and change in depth vs time for a selected hour of dive time for killer whale R58. All data is color-coded by most likely dive type (rows one and three) as well as most likely dive phase (rows two and four).