# Scalable Learning in Latent State Sequence Models

Christopher Aicher

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Emily B. Fox, Chair

Marina Meila

Zaid Harchaoui

Program Authorized to Offer Degree:
Statistics

University of Washington

**Abstract**

Scalable Learning in Latent State Sequence Models

Christopher Aicher

Chair of the Supervisory Committee:
Associate Professor Emily B. Fox
Statistics

In this dissertation, we develop *scalable* learning methods for sequential data models with latent (hidden) states. State space models (SSMs) and recurrent neural networks (RNNs) are popular models for sequential data using latent states. By augmenting an observed sequence with a latent state sequence, SSMs and RNNs can capture complex temporal dynamics with a simpler, smaller parametrization. Unfortunately, learning the parameters of these latent state sequence models requires processing the latent states along the entire sequence, which scales poorly for both long and high dimensional sequential data.

For long sequential data, we develop scalable training methods that use stochastic gradients based on processing subsequences. Unlike independent data models, stochastic gradients for sequential data break temporal dependencies and as a result are *biased*. We develop theory to analyze the effect of this bias on learning and develop efficient estimators to control this bias. For SSMs, we use *buffered* stochastic gradient estimates, which reduces the bias by passing additional messages in a buffer around each subsequence. For RNNs, we *adaptively* truncate backpropagation to save computation and memory when possible. We find these methods provides significant speed-ups in both synthetic and real data sets with millions of time points (i.e. ion-channel recordings, canine electroencephalogram recordings, historical weather data, financial exchange rate data, and text corpus data), while maintaining accuracy similar to the computationally prohibitive batch approaches.

For high dimensional sequential data, we focus on the computational challenge of marginalizing latent variables of many time series when clustering. Existing Bayesian methods for learning clusters of time series either mix slowly (naive Gibbs) or scale cubically in the number of dimensions (collapsed Gibbs). We propose an approximate collapsed Gibbs sampling scheme that improves mixing by approximately collapsing out parameters using expectation propagation, while scaling linearly instead of cubically. We show empirically on synthetic data, robust clustering of MNIST, and housing price prediction that our approximate sampler has similar performance to a collapsed Gibbs sampler at a fraction of the runtime.

# TABLE OF CONTENTS

# ACKNOWLEDGMENTS

This dissertation was made possible thanks to the guidance and support of many people.

I thank my advisor, Emily Fox. Her support and guidance throughout my PhD was invaluable to my growth and development as a researcher. I am very grateful for her constant commitment and flexibility to meeting and guiding me, her insightful advice and feedback on a variety of topics, and her useful suggestions on presenting the 'story' of our research.

I thank my committee members, Marina Meila, Zaid Harchaoui, and Hong Qian, for their suggestions and support throughout my PhD candidacy.

I thank all my teachers at both UW and CU for all their lessons and guidance in statistics, machine learning, and computer science. I also thank the department staff, such as Ellen Reynolds, Tracy Pham, and Kristine Chan, who helped with scheduling, printing and paperwork.

I thank Emily's lab group "DYNAMODE" members for their feedback and support during various meetings, hackathons, and paper editing sessions: Alex Tank, Chris Glenn, Chris Xie, Drausin Wulsin, Ian Covert, Jack Baker, Nicholas Foti, Rahul Nadkarni, Sam Ainsworth, Shirley Ren, and Yi-An Ma. I similarly thank the larger "MODE" lab group members: Tyler Johnson, Marco Tulio Ribeiro, Tianqi Chen, Brian Dolhansky, and Jay Garlapati.

I thank my collaborators outside of Emily's lab group: Srshti Putcha, Christopher Nemeth, and Paul Fearnhead at Lancaster and Jacob Sunshine at UW Medicine.

I thank my many mentors from my summer internships that taught me how to organize meetings and projects: Krishna Sridhar at Dato, Meltem Ozmadenci and Yi Xu at Amazon, and Shirley Ren and Juan Lavista at Microsoft.

I thank the statistics students that shared many classes, study sessions, and offices during

my PhD with: Amrit Dhar, Alec Greaves-Tunnel, Alec Zimmer, Anne Wagner, Amy Marsha, Corinne Jones, David Gerard, Eric Kernfield, Hannah Director, Kyle Lo, Lang Liu, Luca Weiss, Richard Guo, Sam Wang, Sean Jewell, Wesley Lee, and too many more friends to list.

I thank my family for all their love and support, especially, my parents, Thomas and Orapunn Aicher; my father- and mother-in-law, Sheung Tat Ng and Kwai Hung Hung; my brother and sister-in-law, Joseph Aicher and Bernadette Bucher; and the love of my life, Hoiyi Ng.

# DEDICATION

*To my friends and family*

# Chapter 1

# INTRODUCTION

Sequential data arises in many fields where individual observations are ordered in time. Unlike independent unordered data, researchers modeling sequential data must account for the temporal dynamics across sequential observations to accurately uncover patterns and make predictions.

A popular recipe for modeling *complex* temporal dynamics is to augment observed sequence with a *latent state sequence*. Latent (or hidden) state sequence models represent complex structure by treating individual observations independently given a latent state sequence and assuming the temporal dynamics arise from connections among these latent states. This separation of observations and latent states allows for complex sequential data modeling with a small parameterization (compared to modeling without latent states). We focus on two latent state sequence models: state space models and recurrent neural networks.

State space models (SSMs) are a probabilistic model of sequential data using a latent state sequence. Specifically, SSMs treat the latent state sequence as latent variables of a Markov chain. By decoupling the dynamics into a transition process between hidden states and an emission process for individual observations, SSMs can capture a wide variety of complex phenomena. Example of SSMs include hidden Markov models (HMMs) (Cappé et al., 2005), linear Gaussian state space models (LGSSMs) (Lütkepohl, 2005), stochastic volatility models (SVMs) (Shephard, 2005), and switching linear dynamical systems (SLDSs) (Fox et al., 2011). This allows for flexible modeling of a variety of complex phenomena and are widely used in many scientific applications such as target tracking (Gordon et al., 1993), financial time series (Goodhart and O'Hara, 1997; Shephard, 2005), genome sequences (Eddy, 1998), or

neural impulse recordings (Davis et al., 2016).

Recurrent neural networks (RNNs) are a deterministic neural network-based model for sequential data using a hidden state sequence. Specifically, RNNs model the evolution of the hidden state sequence by recursively applying a neural network. Examples of RNNs include long short-term memory models (LSTMs) (Hochreiter and Schmidhuber, 1997), gated recurrent units (GRUs) (Cho et al., 2014), and sequence-to-sequence models (Sutskever et al., 2014). These models have found state-of-the-art success in wide range of tasks such as language modeling (Mikolov et al., 2010), machine translation (Sutskever et al., 2014), and reinforcement learning (Van Hasselt et al., 2016).

Learning the parameters of SSMs and RNNs from observed data is a key task for researchers interested in uncovering patterns and making predictions with these latent state sequence models. Given observed training sequences, standard approaches to learning the parameters involve calculating, sampling, or marginalizing the latent state sequence. As the latent state sequence is temporally connected, the runtime and memory required in processing scales with the length and dimensionality of the latent state sequence. This is prohibitive for long and high dimensional sequences. In practice, given a long time series, one could *segment* or *downsample* to reduce length; however, this preprocessing can destroy or change important signals and computational considerations should ideally not limit scientific modeling.

In this dissertation, we develop *scalable* learning methods for training SSMs and RNNs.

For long sequential data, where length is the bottleneck, our approach for scalable learning is to use stochastic gradient methods, such as stochastic gradient descent (SGD) for optimization (Robbins and Monro, 1951; Duchi et al., 2011; Kingma and Ba, 2014) and stochastic gradient Markov chain Monte Carlo (SGMCMC) for Bayesian inference (Welling and Teh, 2011; Ma et al., 2015; Nemeth and Fearnhead, 2019), by restricting computation to contiguous subsequences, which allows us to avoid processing the entire full sequence at each iteration. However, unlike independent data models, the stochastic gradients of sequential data models are *biased* as the minibatches considered break temporal dependencies. This

bias breaks the theoretical guarantees of SGD and SGMCMC, which assume unbiased gradients, and can negatively affect their performance in practice. Therefore, we develop novel theory to quantify the effect of this stochastic gradient bias in SGD and SGMCMC and develop methods for reducing the bias at the cost of additional computation.

For SSMs, we propose stochastic gradient estimators that control bias by performing additional computation in a *buffer* around a subsequence reducing the number of broken temporal dependencies. The idea is to capture a sufficient amount of the memory of the process in this buffer to reduce the bias of the local analysis. We prove that the bias for these *buffered stochastic gradients* decay geometrically in the buffer size under mild conditions, which allows the buffer to be small in practice. Using these buffered stochastic gradients, we develop novel SGMCMC samplers for discrete and continuous, linear and nonlinear SSMs. Our experiments on real and synthetic data demonstrate the effectiveness of our SGMCMC algorithms compared to full sequence gradient MCMC, allowing us to scale inference to long time series with millions of time points.

For RNNs, we consider stochastic gradient estimators based on truncated backpropagation through time (TBPTT) that save computation and memory at the cost of bias by truncating backpropagation after a fixed number of lags (Williams and Zipser, 1995; Sutskever, 2013). In practice, choosing the optimal truncation length is difficult: TBPTT will not converge if the truncation length is too small, and will converge slowly if it is too large. We show that for many realistic RNNs, TBPTT gradients decay geometrically *in expectation* for large lags and prove that under this condition, we can control the bias by varying the truncation length adaptively. Specifically, when the *relative bias* is less than one, we prove a non-asymptotic convergence rate for non-convex SGD with biased gradients. Using this theory, we propose an adaptive TBPTT scheme that converts the problem from choosing a temporal lag to one of choosing a tolerable amount of relative gradient bias. We evaluate our adaptive TBPTT method on synthetic data and language modeling tasks and find that our adaptive TBPTT ameliorates the computational pitfalls of fixed TBPTT.

We also tackle scalable learning for clustering *high-dimensional* latent variable SSMs.

Existing Bayesian methods for inferring clusters of time series either: (i) require conditioning on latent variables to decouple time series, but results in slow mixing or (ii) require calculating a collapsed likelihood, but with computation scaling cubically with the number of time series per cluster. To infer the latent cluster assignments efficiently, we consider approximate methods that trade exactness for scalability. We develop an expectation propagation (EP) based approximation for the collapsed likelihood approach. Our experiments on real and synthetic data show that our approximate sampler enables a runtime-accuracy tradeoff in sampling these types of models, providing results with competitive accuracy much more rapidly, scaling linearly instead of cubically.

### *1.1 Contributions*

We summarize our main contributions below.

**Contributions to theory:**

- We analyze the effect of *biased* stochastic gradients on learning with SGD and on inference with SGMCMC. We prove that for smooth (possibly non-convex) losses, the relative bias of the stochastic gradients control the convergence and convergence rate of SGD. We also prove asymptotic bounds for the bias and mean squared error of finite sample averages and the posterior mean for SGMCMC with biased stochastic gradients.

- We quantify bounds on the bias of stochastic gradients based on subsequences for both SSMs (with buffered stochastic gradients) and RNNs (with TBPTT). For SSMs, we show that the bias in our buffered stochastic gradient estimates decays geometrically; therefore the bias can be easily controlled with small additional overhead. For RNNs, we quantify how to select the truncation length in TBPTT to control the relative bias.

These theoretical results allow us to understand the trade-off between computation and accuracy for our biased stochastic gradient estimates.

**Contributions to methodology:**

- We develop a generic SGMCMC scheme using buffered stochastic gradients for Bayesian inference of the parameters of general SSMs, including extensions for nonlinear SSMs and mixed-type latent variables. By processing subsequences instead of full sequences and controlling the bias using a buffer, these methods leverage stochastic gradients for speedups, while converging to a provably good approximation to the posterior.

- We develop an adaptive TBPTT scheme for training RNNs that allows for computational speed-ups by reducing the truncation length when possible to control the relative bias.

- We develop an EP-based approximate collapsed Gibbs sampler for latent variable clustering models that perform similarly to collapsed Gibbs samplers, without the computational bottlenecks of computing the exact collapsed likelihood.

Our experiments on real and synthetic data show the effectiveness of these scalable methods for long and complex hidden state sequential data models.

### *1.2 Dissertation Outline*

We summarize the organization of the remainder of this dissertation as follows. In Chapter 2, we provide the background for SSMs, RNNs, and parameter learning methods. In Chapter 3, we present our framework and error analysis of buffered stochastic gradients for SSM. In Chapter 4, we present our error analysis of TBPTT and our adaptive TBPTT scheme for training RNNs. In Chapter 5, we present our approximate collapsed Gibbs sampler for latent variable clustering.

# Chapter 2
# BACKGROUND

In this chapter, we review the necessary background on Bayesian inference, stochastic gradient methods, and sequential data models (e.g. SSMs, RNNs) for the later chapters of this dissertation.

## *2.1 Bayesian Inference*

We consider jointly inferring the parameters $\theta$ and latent states $u = u_{1:T}$ of a latent variable model for observations $y = y_{1:T}$

$$p(y, u \,|\, \theta) = \prod_t p(y_t \,|\, u_t, \theta) \cdot p(u \,|\, \theta) \ , \tag{2.1}$$

where $p(y, u \,|\, \theta)$ is the *complete data likelihood*.

The key quantity in Bayesian inference is the posterior distribution. Given the observations and a prior for the parameters, $p(\theta)$, the *posterior* is calculated using Bayes theorem

$$p(u, \theta \,|\, y) \propto p(y, u \,|\, \theta) p(\theta) \ . \tag{2.2}$$

The challenge for Bayesian inference is calculating the normalizing constant of the posterior $\int p(y, u, \theta) du d\theta$.

If the likelihood is conjugate to the prior, then the posterior (and normalizing constant) can be calculated analytically in closed-form. When the likelihood is non-conjugate, a variety of approximate methods have been proposed including sampling based approaches such as Markov chain Monte Carlo (MCMC) (Bishop, 2006) and deterministic approximation approaches such as variational inference (Blei et al., 2017) and expectation propagation (Minka, 2001).

### *2.1.1 Gibbs Sampling*

A classic sampling approach for Bayesian inference is Gibbs sampling, which constructs a Markov chain to draw from the posterior by iteratively sampling from full conditionals. For example, to sample both the latent variables $u$ and parameters $\theta$ from the posterior $p(u, \theta \mid \theta)$, we iterate between:

- Sample $u^{(s+1)} \sim p(u \mid y, \theta^{(s)}))$
- Sample $\theta^{(s+1)} \sim p(\theta \mid y, u^{(s+1)})$

If either $u$ or $\theta$ can be decomposed further (e.g. $\theta_{1:K}$), then we can apply the Gibbs sampling idea further. Different decompositions lead to different variations of Gibbs sampling.

To fix ideas, we use a latent variable clustering model as a running example. We assume that each observation $y_i$ has an associated latent variable $u_i = z_i \in \{1, \ldots, K\}$ denoting its cluster assignment for $i \in \{1, \ldots, N\}$. We split the parameters $\theta$ into cluster-specific parameters defining the observation distribution as $\phi = \phi_{1:K}$ and cluster-proportion parameters $\pi = \pi_{1:K}$, such that

$$p(y, u|\theta) = p(y, z \mid \phi, \pi) = \prod_{i=1}^{N} p(z_i \mid \pi) \cdot p(y_i \mid z_i, \phi) \ .$$

We now present both *naive* Gibbs sampling and *collapsed* Gibbs sampling.

**Naive Gibbs Sampling** The naive Gibbs sampler targets $p(z, \pi, \phi \mid y)$ and iteratively samples each variable from the posterior conditioned on the current value of *all* other variables:

- Sample $z_i^{(s+1)} \sim p(z_i \mid y, z_{-i}^{(s)}, \pi^{(s)}, \phi^{(s)})$ for all $i$
- Sample $\pi^{(s+1)}, \phi^{(s+1)} \sim p(\pi, \phi \mid y, z^{(s+1)})$

Here, $-i$ denotes all elements except $i$. The full conditional of $z_i$ decomposes into the product of a prior and likelihood term

$$p(z_i \mid y, z_{-i}, \pi, \phi) \propto p(z_i \mid \pi) \cdot p(y_i \mid z_i, \phi) \ . \tag{2.3}$$

Because we condition on the parameters $\phi, \pi$, the observation $y_i$ and assignment $z_i$ are conditionally independent of $y_{-i}, z_{-i}$. Therefore, in naive Gibbs we sample $z_i^{(s+1)}$ by simply taking the product of the prior $p_\pi$ and likelihood $p_\phi$ for each possible cluster assignment and then normalizing. One drawback of this naive Gibbs sampling scheme is that it can *mix* (i.e. move between regions of the posterior and escape poor initializations) extremely slowly.

**Collapsed Gibbs Sampling** To improve the mixing of naive Gibbs sampling, collapsed Gibbs targets $p(z \,|\, y)$ by integrating out $\pi, \phi$ and only iterating

- Sample $z_i^{(s+1)} \sim p(z_i \mid y, z_{-i}^{(s)})$ for all $i$

Similar to Eq. (2.3) for naive Gibbs, the conditional posterior can be decomposed into the product of a prior and likelihood term

$$p(z_i \mid y, z_{-i}) \propto p(z_i \mid z_{-i}) \cdot p(y_i \mid y_{-i}, z) \ . \tag{2.4}$$

In contrast to naive Gibbs, here things do not decouple across $i$ as dependencies are introduced in the marginalization of $\pi, \phi$:

$$p(z_i|z_{-i}) = \int p(z_i|\pi)p(\pi|z_{-i}) \ d\pi \tag{2.5}$$

$$p(y_i|y_{-i}, z) = \int p(y_i|z_i, \phi)p(\phi|y_{-i}, z_{-i}) \ d\phi \,. \tag{2.6}$$

When the integrals of Eqs. (2.5) and (2.6) are tractable (e.g. due to conjugacy), sampling $z$ from a collapsed Gibbs sampler can be considered. However, when either of the integrals is intractable, we cannot fully perform collapsed Gibbs sampling. In practice, we integrate (or *collapse*) out the variables that are analytically tractable and condition on those that are not (Van Dyk and Park, 2008; Liu, 2008).

### *2.1.2 Stochastic Gradient MCMC*

An increasingly popular method for *scalable* Bayesian inference is *stochastic gradient* Markov chain Monte Carlo (SGMCMC) (Chen et al., 2015a; Ma et al., 2015; Welling and Teh, 2011).

The idea behind gradient-based MCMC is to generate samples from the posterior distribution $p(\theta \mid y)$ by simulating continuous dynamics for a *potential energy* function $U(\theta) \propto -\log p(y, \theta)$. For example, the *Langevin diffusion* over $U(\theta)$ is given by the stochastic differential equation (SDE)

$$d\theta_s = g(\theta)ds + \sqrt{2}dW_s \ , \tag{2.7}$$

where $s$ indexes continuous time, $dW_s$ is Brownian motion, and $g(\theta) = -\nabla U(\theta) = \nabla_\theta \log p(y, \theta)$ is the gradient of the log-joint distribution. As $s \to \infty$, the distribution of $\theta_s$ converges to the SDE's stationary distribution, which by the Fokker-Planck equation, is the posterior $p(\theta \mid y)$ (Ma et al., 2015). Because we cannot perfectly simulate Eq. (2.7), in practice we use a discretized numerical approximation. One straightforward approximation is the Euler-Mayurma discretization

$$\theta^{(s+1)} \leftarrow \theta^{(s)} + hg(\theta^{(s)}) + \mathcal{N}(0, 2h) \ , \tag{2.8}$$

where $h$ is the stepsize and $s$ indexes discrete time steps. This recursive update defines the Langevin Monte-Carlo (LMC) algorithm. Typically, a Metropolis-Hastings correction step is added to account for the discretization error (Roberts et al., 1996; Roberts and Rosenthal, 1998).

For large datasets, computing $g(\theta)$ at every step in Eq. (2.8) is computationally prohibitive. To alleviate this, the key ideas of *stochastic gradient* Langevin dynamics (SGLD) are to replace $g(\theta)$ with a quick-to-compute unbiased estimator $\hat{g}(\theta)$ and to use a decreasing stepsize $h^{(s)}$ to avoid costly Metropolis-Hastings correction steps (Welling and Teh, 2011; Nemeth and Fearnhead, 2019)

$$\theta^{(s+1)} \leftarrow \theta^{(s)} + h^{(s)}\hat{g}(\theta^{(s)}) + \mathcal{N}(0, 2h^{(s)}) \ . \tag{2.9}$$

For i.i.d. data, an example of $\hat{g}(\theta)$ is to use a random minibatch $\mathcal{S} \subset 1:T$, $|\mathcal{S}| \ll T$

$$\hat{g}(\theta) = -\frac{1}{\Pr(\mathcal{S})} \sum_{t \in \mathcal{S}} \nabla \log p(y_t \,|\, \theta) - \nabla \log p(\theta) \ , \tag{2.10}$$

which only requires $O(|\mathcal{S}|)$ time to compute. When $\widehat{g}_\theta$ is unbiased and with an appropriate decreasing stepsize, the distribution of $\theta^{(k)}$ asymptotically converges to the posterior distribution (Teh et al., 2016). However, in practice one uses a small, finite step-size for greater efficiency, which introduces a small bias (Dalalyan and Karagulyan, 2019).

The accuracy of this *approximate* MCMC scheme depends on the variance of the stochastic gradients $\hat{g}(\theta)$, the discretization stepsize $h$, and the number of steps. Vollmer et al. (2016) and Chen et al. (2015a) explore the estimation error of SGLD for the posterior expected value of a test function $\phi : \Theta \to \mathbb{R}$. Examples of test functions include the heldout loglikelihood on a test set $\phi(\theta) = \log p(y^*|\theta)$, the L2 estimation error $\phi(\theta) = \|\theta - \theta^*\|$, or a projection $\phi(\theta) = \theta_i$. Let $\bar{\phi}$ be the posterior expected value and let $\hat{\phi}_{K,h}$ be the $K$-sample estimator for $\bar{\phi}$ after running SGLD with a fixed stepsize $h$

$$\bar{\phi} = \mathbb{E}_{\,p(\theta|y)}[\phi(\theta)] \quad \text{and} \quad \hat{\phi}_{K,h} = \frac{1}{K} \sum_{k=1}^{K} \phi(\theta^{(k)}) \ . \tag{2.11}$$

Vollmer et al. (2016) show that the bias and MSE of $\hat{\phi}_{K,h}$ is $\mathcal{O}(h + \frac{1}{Kh})$ and $\mathcal{O}(h^2 + \frac{1}{Kh})$ respectively.

A Riemannian extension of SGLD (SGRLD) simulates the Langevin diffusion over a Riemannian manifold with metric $D(\theta)^{-1}$ by preconditioning the gradient and noise of Eq. (2.9) by $D(\theta)$. By incorporating geometric information about structure of $\theta$, SGRLD aims for a diffusion which mixes more rapidly. Suggested examples of the metric $D(\theta)^{-1}$ are the Fisher information matrix $\mathcal{I}(\theta) = \mathbb{E}_{\,y}[\nabla^2 \log p(y \,|\, \theta)]$ or a noisy Hessian estimate $\widehat{\nabla^2 \log p}(y \,|\, \theta)$ (Girolami and Calderhead, 2011; Patterson and Teh, 2013). Given $D(\theta)$, each step of SGRLD is

$$\theta^{(s+1)} \leftarrow \theta^{(s)} + h \left[ D(\theta^{(s)}) \cdot \hat{g}(\theta^{(s)}) + \Gamma(\theta^{(s)}) \right] + \mathcal{N}\left(0, 2hD(\theta^{(s)})\right) \ , \tag{2.12}$$

where the vector $\Gamma(\theta)$ is a correction term $\Gamma(\theta)_i = \sum_j \frac{\partial D(\theta)_{ij}}{\partial \theta_j}$ to ensure the dynamics converge to the target posterior (Ma et al., 2015; Xifara et al., 2014).

Algorithm 1 is the pseudocode for SGLD and SGRLD. An example of `NoisyGradient` is to calculate $\hat{g}$ using a random minibatch, Eq. (2.10). For SGLD, `GetPreconditioner` would return the identity matrix for $D$ and a zero vector for $\Gamma$.

**Algorithm 1** SGLD and SGRLD

Input: data $y$, parameters $\theta^{(0)}$, stepsize $h$

**for** $s = 0, 1, 2, \ldots, K_{\text{steps}} - 1$ **do**

$\hat{g}(\theta^{(s)}) = \texttt{NoisyGradient}(y, \theta^{(s)})$

$D^{(s)}, \Gamma^{(s)} = \texttt{GetPreconditioner}(\theta^{(s)})$

$\theta^{(s+1)} \leftarrow \theta^{(s)} + h^{(s)} \left[D^{(s)} \tilde{g}(\theta^{(s)}) + \Gamma^{(s)}\right] + \mathcal{N}\left(0, 2h^{(s)} D^{(s)}\right)$

**end for**

Return $\theta^{(1:K_{\text{steps}})}$

Many extensions of SGLD exist in the literature, including using control variates to reduce the variance of $\widehat{g}_\theta$ (Baker et al., 2017; Nagapetyan et al., 2017; Chatterji et al., 2018) and using augmented dynamics to improve mixing (Ma et al., 2015), such as stochastic gradient Hamiltonian Monte Carlo (Chen et al., 2014) and stochastic gradient Nosé-Hoover thermostat (Ding et al., 2014).

### *2.1.3 Expectation Propagation*

Unlike MCMC methods which approximate the posterior by sampling, *expectation propagation* (EP) approximates the posterior by solving a variational optimization problem (Minka, 2001). Specifically, EP looks for a distribution $q(\theta)$ in a (tractable) family of distributions $\mathcal{Q}$ such that $q(\theta)$ is close to the posterior $p(\theta)$ in Kullback-Leibler divergence

$$\mathrm{KL}(p \,||\, q) = \mathbb{E}_{\,\theta\sim p}[\log q(\theta) - \log p(\theta)] = \int p(\theta) \log \frac{q(\theta)}{p(\theta)} \, d\theta \ . \tag{2.13}$$

Unfortunately, directly minimizing $\mathrm{KL}(p||q)$ is intractable as it requires expectations with respect to the posterior $p$. Instead of directly minimizing $\mathrm{KL}(p||q)$, the idea of EP is to iteratively minimize individual likelihood terms when the posterior factorizes.

If the posterior $p$ for $\theta$ given all observations $y$ is

$$p(\theta|y) \propto p_0(\theta) \cdot \prod_{j=1}^{N} f_j(y_j, \theta) \ , \tag{2.14}$$

where $p_0(\theta)$ is the prior for $\theta$ belonging to exponential family $\mathcal{Q}$, then the EP idea is to approximate the likelihood terms $f_j(y_j, \theta)$ with *site approximations* $\tilde{f}_j(\theta)$ that are conjugate to the prior. For example, if the prior $p_0(\theta)$ is Gaussian, then

$$\tilde{f}_j \in \tilde{\mathcal{F}} = \{C_j \mathcal{N}(\theta|\mu_j, \Sigma_j) \ : \ C_j > 0, \mu_j \in \mathbb{R}^n, \Sigma_j \in \mathbb{S}^n\} \tag{2.15}$$

where $\mathcal{N}(\cdot|\mu, \Sigma)$ denotes a multivariate Gaussian density with mean $\mu$ and variance $\Sigma$ and $C$ is a scaling constant. Note that $\tilde{f}_j$ is a *likelihood approximation*, not a probability density, thus its parameterization does not necessarily integrate to one (Seeger, 2005).

The resulting approximation $q(\theta|y)$ is then

$$q(\theta|y) \propto p_0(\theta) \cdot \prod_{j=1}^{N} \tilde{f}_j(\theta) \ , \tag{2.16}$$

which is a member of the exponential family $\mathcal{Q}$. To construct good site approximations $\tilde{f}_{1:N}$ for $f_{1:N}$, EP iteratively selects each $\tilde{f}_i$ to minimize the 'local' KL divergence of $q$ to the *tilted* distribution

$$\tilde{p}_i \propto q(\theta) \cdot \frac{f_i(\theta)}{\tilde{f}_i(\theta)} = p_0(\theta) \cdot \prod_{j \neq i} \tilde{f}_j(\theta) \cdot f_i(\theta) \tag{2.17}$$

The tilted distribution $\tilde{p}_i$ is the approximation $q$ with the $i$-th site approximation $\tilde{f}_i$ replaced with the true likelihood term $f_i$. EP selects $\tilde{f}_i$ to minimize the KL divergence $\mathrm{KL}(\tilde{p}_i||q)$ (holding $\tilde{f}_j$, $j \neq i$ fixed)

$$\tilde{f}_i(\theta) = \underset{\tilde{f} \in \tilde{\mathcal{F}}}{\operatorname{argmin}} \, \mathrm{KL}\left(\tilde{p}_i(\theta) \ || \ q(\theta)\right) = \frac{\operatorname{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(\tilde{p}_i(\theta) \, || \, q(\theta))}{p_0(\theta) \cdot \prod_{j \neq i} \tilde{f}_j(\theta)} \ . \tag{2.18}$$

Minimizing each local KL divergence, Eq. (2.18), is equivalent to matching $\theta$'s sufficient statistics' moments. This can be done analytically for a wide class of distributions (Seeger, 2005).

EP iteratively applies Eq. (2.18) until convergence. However, standard EP is not guaranteed to converge and may have multiple fixed points. By minimizing the local KL-divergence

to the tilted-distribtions, the fixed points of EP are equivalent to maximizing Bethe entropy approximation of the log-marginal probability (Hasenclever et al., 2017; Heskes and Zoeter, 2002)

$$\underset{q\in\mathcal{Q}}{\operatorname{argmax}} -H(q) + \sum_{i=1}^{T}(-H(\tilde{p}_i) + H(q)) \ , \tag{2.19}$$

where $H(q) = -\mathbb{E}_q[\log q]$ is entropy.

The Bethe entropy approximation is not concave in $q$, which allows for multiple fixed points. Furthermore, because EP iteratively applies simple-to-compute "coordinate-ascent" updates Eq. (2.18), it is possible for EP to converge to limit-cycles. To overcome these problems, Heskes and Zoeter (2002) introduced an inner-outer "double"-loop algorithm at the cost of additional computation. This was further extended to allow for parallel computation using stochastic natural gradients by Teh et al. (Hasenclever et al., 2017).

EP has been shown to be consistent and exact in the large data limit for log-concave $p$ for the Gaussian approximating family $\mathcal{Q}$ by showing EP asymptotically behaves like the Laplace approximation of Newton-method's (Dehaene and Barthelmé, 2018).

### *2.2 State Space Models*

State space models (SSMs) for time series are a class of discrete-time bivariate stochastic process $\{u_t, y_t\}_{t=1}^T$, consisting of a latent state sequence $u := u_{1:T}$ generated by Markov chain and an observation sequence $y := y_{1:T}$ generated independently conditioned on $u$ (Cappé et al., 2005). For a generic SSM, the joint distribution of $y$ and $u$ factorizes as

$$p(y, u \,|\, \theta) = \prod_{t=1}^{T} p(y_t \,|\, u_t, \theta) p(u_t \,|\, u_{t-1}, \theta) \cdot p_0(u_0) \ , \tag{2.20}$$

where $\theta$ are model-specific parameters, $p(y_t|u_t, \theta)$ is the *emission density*, $p(u_t|u_{t-1}, \theta)$ is the *transition density*, and $p_0(u_0)$ is a prior for the latent states. As the latent state sequence $u$ is unobserved, the likelihood of $\theta$ given only the observations $y$ (marginalizing $u$) is

$$p(y \,|\, \theta) = \int \prod_{t=1}^{T} p(y_t \,|\, u_t, \theta) p(u_t \,|\, u_{t-1}, \theta) \cdot p_0(u_0) \ du \ , \tag{2.21}$$

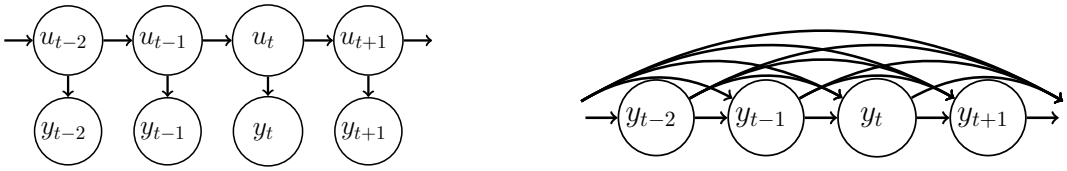


Figure 2.1: Graphical Model of a SSM: (left) the joint process $u, y$, Eq. (2.20) and (right) $y$ marginalizing out $u$, Eq. (2.21). The parameters $\theta$ are not shown, but connect to all nodes.

Unconditionally, the observations $y$ are not independent and the graphical model of this *marginal likelihood*, Eq. (2.21), has many long term dependencies, Figure 2.1 (right). In contrast, when conditioned on $u$ the observations $y$ are independent and the *complete-data likelihood*, Eq. (2.20), has a simpler chain structure, Figure 2.1 (left). The latent chain structures lets SSMs represent complex structure, while maintaining tractable inference.

### *2.2.1 Example SSMs*

We present two simple examples of SSMs: a discrete latent variable model, the Gaussian HMM, and a continuous latent variable model, the linear Gaussian SSM (LGSSM). Additional details and more advanced SSMs (such as mixed-type latent variables and nonlinear SSMs) are described in Chapter 3.4. For clarity, we will denote the discrete latent variables with $z$ instead of $u$ and continuous latent variables with $x$ instead of $u$.

#### *Gaussian HMM*

The Gaussian HMM is a discrete latent state HMM with Gaussian emissions. Let $K$ denote the number of latent states and $m$ denote the dimension of each observation. That is $u_t \equiv z_t \in \{1, \ldots, K\}$ and $y_t \in \mathbb{R}^m$.

The transition process for the latent variables is an HMM: the discrete latent variable at time $t$ is drawn from a categorical distribution with probability $\Pi_{z_{t-1}}$ dependent on the latent variable at time $t-1$.

The emission process for the observations is a Gaussian mixture of $K$ components. Conditional on the latent variable at time $t$, the observation is drawn from the Gaussian component with mean $\mu_{z_t}$ and variance $\Sigma_{z_t}$.

The generative process of the Gaussian HMM can be summarized as follows

$$\begin{aligned} z_t \,|\, z_{t-1}, \theta &\sim \text{Categorical}(z_t \,|\, \Pi_{z_{t-1}}) \\ y_t \,|\, z_t, \theta &\sim \mathcal{N}\left(y_t \,|\, \mu_{z_t}\,,\, \Sigma_{z_t}\right), \end{aligned} \tag{2.22}$$

where $\theta = \{\Pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are the parameters with $\Pi_k \in \Delta^K$ (simplex over $K$ states), $\mu_k \in \mathbb{R}^m$, $\Sigma_k \in \mathbb{S}_+^m$ (positive definite matrices) for $k = 1, \ldots, K$.

*Linear Gaussian SSM*

The LGSSM is a continuous latent state SSM with both linear Gaussian transitions and emissions. Let $n$ denote the dimension of latent states and $m$ denote the dimension of each observation. That is $u_t \equiv x_t \in \mathbb{R}^n$ and $y_t \in \mathbb{R}^m$. The transition process for the latent variables is a linear Gaussian autoregressive process and the emission process is linear Gaussian

$$\begin{aligned} x_t \,|\, x_{t-1}, \theta &\sim \mathcal{N}(x_t \,|\, Ax_{t-1}, Q) \\ y_t \,|\, z_t, \theta &\sim \mathcal{N}\left(y_t \,|\, Cx_t\,,\, R\right), \end{aligned} \tag{2.23}$$

where $\theta = \{A, Q, C, R\}$ are the parameters with $A \in \mathbb{R}^{n \times n}$ as the latent state transition matrix, $Q \in \mathbb{S}_+^n$ as the transition noise covariance, $C \in \mathbb{R}^{m \times n}$ as the emission matrix, and $R \in \mathbb{S}_+^m$ as the emission noise covariance.

### *2.2.2 Fisher's Identity*

To infer $\theta$ given $y$, we can maximize the marginal likelihood $p(y \,|\, \theta)$ or, given a prior $p(\theta)$, sample or maximize the posterior $p(\theta \,|\, y) \propto p(y \,|\, \theta)p(\theta)$. However, traditional inference methods for $\theta$, such as expectation maximization (EM), variational inference, or Gibbs sampling,

take advantage of the conditional independence structure in $p(y, u \mid \theta)$, Eq. (2.20), rather than working directly with $p(y \mid \theta)$, Eq. (2.21) (Beal et al., 2003; Scott, 2002). To use $p(y, u \mid \theta)$ with unobserved $u$, these methods rely on sampling or taking expectations of $u$ from the posterior $\gamma(u) := p(u \mid y, \theta)$. As an example, gradient-based methods take advantage of *Fisher's identity* (Cappé et al., 2005)

$$\nabla \log p(y \mid \theta) = \mathbb{E}_{u|y,\theta}[\nabla \log p(y, u \mid \theta)] = \mathbb{E}_{u\sim\gamma}[\sum_{t=1}^{T} \nabla \log p(y_t, u_t \mid u_{t-1}, \theta)] \ , \tag{2.24}$$

which allows gradients of Eq. (2.21) to be computed in terms of Eq. (2.20).

To compute the latent state posterior $\gamma(u) = p(u|y, \theta)$, these methods use the well-known *forward-backward algorithm* (Scott, 2002; Cappé et al., 2005). The algorithm works by recursively computing a sequence of forward messages $\alpha_t(u_t)$ and backward messages $\beta_t(u_t)$ which are used to compute the pairwise marginals of $\gamma$. More specifically,

$$\alpha_t(u_t) := p(u_t, y_{\leq t} \mid \theta) = \int p(y_t, u_t \mid u_{t-1}, \theta)\alpha_{t-1}(u_{t-1})\, du_{t-1} \tag{2.25}$$

$$\beta_t(u_t) := p(y_{>t} \mid u_t, \theta) = \int p(y_{t+1}, u_{t+1} \mid u_t, \theta)\beta_{t+1}(u_{t+1})\, du_{t+1} \tag{2.26}$$

$$\gamma_{t-1:t}(u_{t-1}, u_t) := p(u_{t-1}, u_t \mid y, \theta) \propto \alpha_{t-1}(u_{t-1})p(y_t, u_t \mid u_{t-1}, \theta)\beta_t(u_t) \ . \tag{2.27}$$

When message passing is tractable (i.e., when Eqs. (2.25)-(2.26) involve discrete or conjugate likelihoods), the forward-backward algorithm can be calculated in closed form. When message passing is intractable, the messages can be approximated using Monte-Carlo sampling methods (e.g. blocked Gibbs sampling (Carter and Kohn, 1994; Fox et al., 2011), particle methods (Andrieu et al., 2010; Briers et al., 2010; Doucet and Johansen, 2009; Sudderth et al., 2010)). In all cases, the computation time and memory of the forward-backward algorithm scales in both the length of time series $T$ and the dimension of the latent states $u$. For example, the time and memory for Gaussian HMMs scales $\mathcal{O}(TK^2)$ and the LGSSM scales $\mathcal{O}(TM^3)$.

### *2.2.3 Particle Smoothers*

When message passing to compute $p(u|y,\theta)$ is not available in closed-form, we can still approximate the gradient of the log-likelihood, Eq. (2.24), using *particle smoothing* methods. Particle smoothing algorithms (see e.g. Doucet and Johansen, 2009; Fearnhead and Künsch, 2018) approximate the expectation of a function $\phi(u_{1:T})$ with respect to the posterior density, $p(u_{1:T}|y_{1:T},\theta)$ using sequential Monte Carlo. This is done by generating a collection of $N$ random samples or *particles*, $\{u_t^{(i)}\}_{i=1}^N$ and calculating their associated importance weights, $\{w_t^{(i)}\}_{i=1}^N$, recursively over time. The particles and weights are updated using a *particle filter* with *sequential importance resampling* (SIR) (Doucet and Johansen, 2009) in the following manner.

(i) *Resample* auxiliary ancestor indices $\{a_1,\ldots,a_N\}$ with probabilities proportional to the importance weights, i.e. $a_i \sim \text{Categorical}(w_{t-1}^{(i)})$.

(ii) *Propagate* particles $u_t^{(i)} \sim q(\cdot|u_{t-1}^{(a_i)},y_t,\theta)$, using a proposal distribution $q(\cdot|\cdot)$.

(iii) *Update* and normalize the weight of each particle,

$$w_t^{(i)} \propto \frac{p(y_t|u_t^{(i)},\theta)p(u_t^{(i)}|u_{t-1}^{(a_i)},\theta)}{q(u_t^{(i)}|u_{t-1}^{(a_i)},y_t,\theta)} \;\;,\;\; \sum_i w_t^{(i)} = 1 \;\;. \tag{2.28}$$

If the proposal density $q(u_t|u_{t-1},y_t,\theta)$ is the transition density $p(u_t|u_{t-1},\theta)$, SIR is also known as the *bootstrap particle filter* (Gordon et al., 1993). By using the transition density for proposals, the importance weight recursion in Eq. (2.28) simplifies to $w_t^{(i)} \propto p(y_t|u_t^{(i)},\theta)$.

The auxiliary ancestor indices, $\{a_i\}_{i=1}^N$, represent the *ancestors* of the particles, $\{u_t^{(i)}\}_{i=1}^N$, sampled at time $t$ and allows us to keep track of the lineage of particles over time $\xi_t^{(i)} = [\xi_{t-1}^{(a_i)}, u_t^{(i)}]$. To convert the particle filter into a particle smoother, we evaluate our function $\phi(u_{1:T})$ along these sampled lineages

$$\mathbb{E}_{p(u|y,\theta)}[\phi(u)] \approx \hat{\phi} = \sum_{i=1}^N w_T^{(i)} \cdot \phi\left(\xi_T^{(1:N)}\right) \;\;.$$

When our target function decomposes into a pairwise sum $\phi(u_{1:T}) = \sum_{t=1}^{T} \phi_t(u_t, u_{t-1})$ — such as for Fisher's identity $\phi_t(u_t, u_{t-1}) = \nabla_\theta \log p(y_t, u_t \mid u_{t-1}, \theta)$ — then we only need to keep track of the partial sum $\mathrm{PS}(\phi)_t = \sum_{s=1}^{t} \phi_s(u_s, u_{s-1})$ rather than the full lineage of $u_{1:t}$ during SIR. The complete particle smoothing scheme is detailed in Algorithm 2.

**Algorithm 2** Particle Smoother

1: **Input:** number of particles, $N$, pairwise statistics, $h_{1:T}$, observations $y_{1:T}$, proposal density $q$,
2: Draw $u_0^{(i)} \sim \nu(u_0|\theta)$, set $w_0^{(i)} = \frac{1}{N}$, and $\mathrm{PS}(\phi)_0^{(i)} = 0 \ \forall i$.
3: **for** $t = 1, \ldots, T$ **do**
4: Resample ancestor indices $\{a_1, \ldots, a_N\}$.
5: Propagate particles $u_t^{(i)} \sim q(\cdot|u_{t-1}^{(a_i)}, y_t, \theta)$.
6: Update each $w_t^{(i)}$ according to Eq. (2.28).
7: Update statistics $\mathrm{PS}(\phi)_t^{(i)} := \mathrm{PS}(\phi)_{t-1}^{(a_i)} + \phi_t(u_t^{(i)}, u_{t-1}^{(a_i)})$.
8: **end for**
9: Return $\hat{\phi} := \sum_{i=1}^{N} w_T^{(i)} \cdot \mathrm{PS}(\phi)_T^{(i)}$. // $\hat{\phi}$ is the estimate for $\mathbb{E}_{u|y,\theta}(\phi)$

A key challenge for particle smoothers is handling large sequence length $T$. Not only do long sequences require $\mathcal{O}(T)$ computation, but particle smoothers also require a large number of particles, $N$, to avoid *particle degeneracy*: the use of resampling in the particle filter causes path-dependence over time, depleting the number of distinct particle paths overall. For Algorithm 2, the asymptotic variance in the estimate scales as $\mathcal{O}(T^2/N)$ (Poyiadjis et al., 2011). Therefore to maintain a constant variance, the number of particles would need to increase quadratically with $T$, which is computationally infeasible for long sequences. Poyiadjis et al. (2011); Nemeth et al. (2016) and Olsson and Westerborn (2017) propose alternatives to Step 7 of Algorithm 2 that trade additional computation or bias to decrease the variance in $H$ to $\mathcal{O}(T/N)$. Fixed-lag particle smoothers provide another approach to avoid particle degeneracy, where sample paths are not updated after a fixed lag (Kitagawa and Sato, 2001; Dahlin et al., 2015).

### *2.2.4 Wasserstein Distance*

Our analysis of gradient bias for SSMs in Chapter 3.3 relies on Wasserstein distance, which we introduce here. Wasserstein distance is a class of probability distribution metrics. Although these metrics are increasingly popular in optimal transport (Villani, 2008), we will focus specifically on the properties necessary for our analysis of the latent state posteriors in SSMs.

We first review the definition of Wasserstein distance. Let $\mathcal{W}_p(\gamma, \widetilde{\gamma})$ be the $p$-Wasserstein distance,

$$\mathcal{W}_p(\gamma, \widetilde{\gamma}) := \left[\inf_{\xi} \int \|u - \widetilde{u}\|_2^p \, d\xi(u, \widetilde{u})\right]^{1/p} , \tag{2.29}$$

where $\xi$ is a joint measure or *coupling* over $(u, \widetilde{u})$ with marginals $\int_{\widetilde{u}} d\xi(u, \widetilde{u}) = d\gamma(u)$ and $\int_u d\xi(u, \widetilde{u}) = d\widetilde{\gamma}(\widetilde{u})$. Wasserstein distance satisfies all the properties of a metric.

A useful property of the 1-Wasserstein distance is the following Kantorovich-Rubinstein duality formula for the difference of expectations of Lipschitz functions (Villani, 2008)

$$\mathcal{W}_1(\gamma, \widetilde{\gamma}) = \sup_{\|\psi\|_{Lip} \leq 1} \left\{ \int \psi \, d\gamma - \int \psi \, d\widetilde{\gamma} \right\} \;\Rightarrow\; |\mathbb{E}_{\gamma}[\psi] - \mathbb{E}_{\widetilde{\gamma}}[\psi]| \leq \|\psi\|_{Lip} \cdot \mathcal{W}_1(\gamma, \widetilde{\gamma}) \;, \tag{2.30}$$

where $\|\psi\|_{Lip}$ denotes the Lipchitz constant of $\psi$.

**Wasserstein distance for sequences of random variables** We now preview how to bound Wasserstein distances between *sequences* of variables.

Let $\Psi : \mathcal{U} \to \mathcal{V}$ be a *transition kernel* between random variables $u$ and $v$, that is for any measure $\mu(u)$ over $\mathcal{U}$, we define the induced measure $(\mu\Psi)(v)$ over $\mathcal{V}$ as $(\mu\Psi)(v) = \int \Psi(u, v)\mu(du)$.

A *random mapping* $\psi$ is a random function that maps $\mathcal{U}$ to $\mathcal{V}$ such that if $u \sim \mu$ then $\psi(u) \sim \mu\Psi$. For example, if $\Psi(u, v) = \mathcal{N}(v \,|\, u, 1)$, then a random mapping for $\Psi$ is the identity function plus Gaussian noise $\psi(u) = u + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. Note that if $\psi$ is deterministic, then $(\mu\Psi)(v)$ is the *push-forward* measure of $\mu$ through the mapping $\psi$; otherwise it is the average (or marginal) over $\psi$ of push-forward measures (Villani, 2008).

We say a random mapping $\psi$ and the kernel $\Psi$ has Lipschitz constant $L$ with respect to Euclidean distance if

$$\|\psi\|_{Lip} = \|\Psi\|_{Lip} = L \Leftrightarrow \sup_{u,u'} \left\{ \frac{\mathbb{E}_{\psi}[\|\psi(u) - \psi(u')\|_2]}{\|u - u'\|_2} \right\} \leq L \ . \tag{2.31}$$

Note that $L$ is an upper-bound on the *expected value* of Lipschitz constants for random instances of $\psi$.

These definitions are useful for proving bounds in Wasserstein distance for SSMs. For example, if the Lipschitz constant is less than one $\|\Psi\|_{Lip} < 1$, then the kernel $\Psi$ induces a contraction in $p$-Wasserstein distance. That is $\mathcal{W}_p(\mu\Psi, \tilde{\mu}\Psi) \leq \|\Psi\|_{Lip} \cdot \mathcal{W}_p(\mu, \tilde{\mu})$ as

$$\begin{aligned}
\mathcal{W}_p(\mu\Psi, \tilde{\mu}\Psi)^p &= \inf_{\xi(\mu\Psi, \tilde{\mu}\Psi)} \int \|v - \tilde{v}\|_2^p \, d\xi(v, \tilde{v}) \\
&\leq \inf_{\xi(\mu, \tilde{\mu})} \int \|\psi(u) - \psi(\tilde{u})\|_2^p \, d\xi(u, \tilde{u}) df_K \\
&\leq \inf_{\xi(\mu, \tilde{\mu})} \int \|\Psi\|_{Lip}^p \cdot \|u - \tilde{u}\|_2^p \, d\xi(u, \tilde{u}) \\
&= \|\Psi\|_{Lip}^p \cdot \mathcal{W}_p(\mu, \tilde{\mu})^p \ .
\end{aligned} \tag{2.32}$$

This implies that for any pair of distributions $\mu$ and $\tilde{\mu}$, after applying the transition kernel $\Psi$, the pair $\mu\Psi$ and $\tilde{\mu}\Psi$ are closer (in any $p$-Wasserstein distance).

### *2.3 Recurrent Neural Networks*

Recurrent neural networks (RNNs) are a popular method for nonlinear modeling of temporal dynamic behavior. A generic RNN is designed to take a variable length sequence of inputs $x_t \in \mathbb{R}^{d_x}$ and return a hidden state output $h_t \in \mathbb{R}^{d_h}$ for each time step $t$. Crucially, RNNs share information across time steps by including the hidden state of the previous time step $h_{t-1}$ as an input at time $t$. That is,

$$h_t = H(h_{t-1}, x_t; \theta) \tag{2.33}$$

for some function $H : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_h}$ with parameters $\theta \in \Theta$ (e.g, weights and biases) of the network. This framework encompasses most popular RNNs. We now present some examples that we use later.

### *2.3.1 Example RNNs*

**Simple RNN:** A simple RNN is a linear map composed with a non-linear activation function $\rho$ (e.g., $\rho = \tanh$)

$$h_t = H(h_{t-1}, x_t, \theta) = \rho(W h_{t-1} + U x_t) \ ,$$

where the weight matrices $W, U$ are the parameters.

**Long Short-Term Memory (LSTM):** A popular class of sequence models are LSTMs (Hochreiter and Schmidhuber, 1997). The hidden state consists of a pair of vectors $h_t = (c_t, \tilde{h}_t)$ and

$$\begin{aligned}
f_t &= \sigma(W_f \tilde{h}_{t-1} + U_f x_t) \\
i_t &= \sigma(W_i \tilde{h}_{t-1} + U_i x_t) \\
o_t &= \sigma(W_o \tilde{h}_{t-1} + U_o x_t) \\
z_t &= \tanh(W_z \tilde{h}_{t-1} + U_z x_t) \\
c_t &= i_t \circ z_t + f_t \circ c_{t-1} \\
\tilde{h}_t &= o_t \cdot \tanh(c_t) \ ,
\end{aligned}$$

where the eight matrices $W_*, U_*$, for $* \in \{f, i, o, z\}$ are the parameters, $\sigma$ is the logistic function, and $\circ$ denotes elementwise multiplication. LSTMs are examples of *gated-RNNs* which capture more complex time dependence through the use of gate variables $f_t, i_t, o_t$. Other examples of gated-RNNs includes gated recurrent units (GRUs) (Cho et al., 2014) and gated orthogonal recurrent units (GORUs) (Jing et al., 2019).

**Stacked RNNs:** RNNs can be composed by *stacking* the hidden layers. This allows different layers to learn structure at varying resolutions. Each RNN-layer treats the output of the previous layer as its input. Specifically, each layer $l = 1, \ldots, N_l$ is described as

$$h_t^{(l)} = H_{(l)}(h_{t-1}^{(l)}, h_t^{(l-1)}, \theta_{(l)}) \ ,$$

where $h_t^{(0)} = x_t$.

### *2.3.2 Training RNNs*

As with most deep neural networks, RNNs are trained with gradient descent. Given an observed input sequence $x_{1:T}$ and a loss function for each time step $\mathcal{L}_t$, we train $\theta$ to minimize the average loss over the entire sequence

$$\mathcal{L}(\theta) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_t(h_t(\theta, x_{1:t})) \ . \tag{2.34}$$

For example, an individual loss at time step $t$ measures the accuracy of $h_t$ at predicting a target output $y_t$.

To scale gradient descent for large $T$, we use stochastic gradient descent (SGD) (Robbins and Monro, 1951), which uses a random estimator $\hat{g}$ based on a minibatch for the full gradient $g = \nabla_\theta \mathcal{L}$. We first consider estimating $g$ using the gradient of the loss at a random individual time step, $\mathcal{L}_s$, where $s$ is a *random* index drawn uniformly from $\{1, \ldots, T\}$. In Chapter 4.3.1, we discuss efficient ways of computing Eq. (2.36) for multiple consecutive losses $\mathcal{L}_{s:s+t}$ simultaneously.

Unrolling the RNN and applying the chain rule, the gradient of $\mathcal{L}_s$ is

$$\nabla_\theta \mathcal{L}_s = \frac{\partial \mathcal{L}_s}{\partial \theta} + \sum_{k=0}^{s} \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \cdot \frac{\partial h_{s-k}}{\partial \theta}, \tag{2.35}$$

which can be efficiently computed using *backpropagation through time* (BPTT) (Werbos et al., 1990; Williams and Zipser, 1995). This consists of a forward pass to compute all the hidden states $h_t$ and a backward pass where the gradients are *backpropagated* through the network via reverse-mode differentiation (LeCun et al., 1988; Werbos, 1994; Rumelhart et al., 1985; Goodfellow et al., 2016) using auto-differentiation tools such as Tensorflow (Abadi et al., 2016), PyTorch (Paszke et al., 2017) or MXNet (Chen et al., 2015b).

### *2.3.3 Truncated Backpropagation Through Time*

For long sequences $T$, unrolling the RNN is both computationally and memory prohibitive; therefore in practice, the gradients terms, Eq. (2.35), are truncated after $K$ steps (Williams

and Zipser, 1995; Sutskever, 2013)

$$\widehat{\nabla}_\theta^K \mathcal{L}_s = \frac{\partial \mathcal{L}_s}{\partial \theta} + \sum_{k=0}^{K} \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \frac{\partial h_{s-k}}{\partial \theta}, \tag{2.36}$$

for $K \ll T$. This reduces the memory and computation required as only the last $K$ hidden states $h_{s-K:s}$ instead of all hidden states $h_{1:s}$ are processed.

Let $\hat{g}_K = \widehat{\nabla}_\theta^K \mathcal{L}_s$ be our stochastic gradient estimator for $g$ truncated at $K$ steps. When $K = T$, $\hat{g}_T$ is an unbiased estimate for $g$; however for general $K < T$, the gradient estimator $\hat{g}_K$ is *biased*, $\mathbb{E}\left[\hat{g}_K(\theta)\right] = \mathbb{E}_s\left[\hat{\nabla}_\theta^K \mathcal{L}_s\right] \neq \mathbb{E}_s\left[\nabla_\theta \mathcal{L}_s\right]$. In practice, the truncation length $K$ is chosen heuristically to be "large enough" to capture the memory in the underlying process, in hope that the bias of $\hat{g}_K(\theta)$ does not affect the convergence of SGD. In general, there are no guarantees on the size of this bias or the convergence of the overall optimization for fixed $K < T$.

*Efficiently Computing TBPTT*

To efficiently compute TBPTT in practice, we calculate the gradients for multiple consecutive losses simultaneously. Following Williams and Zipser (1995), we denote BPTT$(K_1, K_2)$ to be truncated backpropagation for $K_2$ consecutive losses $\mathcal{L}_{s-K_2+1:s}$ backpropagated over $K_1$ steps, that is

$$\text{BPTT}(K_1, K_2) = \frac{1}{K_2} \sum_{k'=0}^{K_2-1} \sum_{k=s-t}^{K_1} \frac{\partial \mathcal{L}_{s-k'}}{\partial h_{s-k}} \cdot \frac{\partial h_{s-k}}{\partial \theta} \quad . \tag{2.37}$$

Observe that BPTT$(K_1, K_2)$ can be computed efficiently using the recursion

$$b_k = \begin{cases} b_{k-1} \cdot \dfrac{\partial h_{s-k+1}}{\partial h_{s-k}} + \dfrac{\partial \mathcal{L}_{s-k}}{\partial h_{s-k}} & \text{if } k < K_2 \\ b_{k-1} \cdot \dfrac{\partial h_{s-k+1}}{\partial h_{s-k}} & \text{if } k \geq K_2 \end{cases}, \tag{2.38}$$

where $\text{BPTT}(K_1, K_2) = \frac{1}{K_2} \sum_{k=0}^{K_1} b_k \cdot \frac{\partial h_{s-k}}{\partial \theta}$.

When $K_1 = K$ and $K_2 = 1$, then we obtain $\text{BPTT}(K, 1) = \hat{g}_K$. However, individually calculating $S$ samples of $\hat{g}_K$ using BPTT$(K, 1)$ takes $\mathcal{O}(JK)$ computation; whereas the same

gradients (plus extra lags) can be computed using BPTT($J + K, K$) in $\mathcal{O}(J + K)$ time. In practice a popular default setting found in Tensorflow is to set $K_1 = K_2 = K$; however this overweights small lags, as the $k$-th loss is only backpropagated only $K - k$ steps. To ensure all $K$ losses are backpropagated at least $K$ steps, we use BPTT($2K, K$) in Chapter 4.

# Chapter 3

# STOCHASTIC GRADIENT MCMC FOR STATE SPACE MODELS

## *3.1 Introduction*

To help scale inference in SSMs, we consider stochastic gradient Markov chain Monte Carlo (SGMCMC), a popular method for scaling Bayesian inference to large data sets (Chen et al., 2015a; Ma et al., 2015; Welling and Teh, 2011) (see Chapter 2.1.2 and 2.2 for background). The key idea of SGMCMC is to employ stochastic gradient estimates based on subsets or 'minibatches' of data, avoiding costly computation of gradients on the full dataset, such that the resulting dynamics produce samples from the posterior distribution over SSM parameters. This approach has found much success in *independent* data models, where the stochastic gradients are *unbiased* estimates of the true gradients, Eq. (2.10). However, when applying SGMCMC to SSMs, such naive stochastic gradients are *biased*, as subsampling the data breaks dependencies in the SSM's latent state sequence. This bias can destroy the dynamics of SGMCMC causing it to fail when applied to SSMs. The challenge is to correct these stochastic gradients for SSMs while maintaining the computational benefits of SGMCMC.

In this chapter, we develop computationally efficient stochastic gradient estimators for inference in general discrete-time SSMs. To control the bias of stochastic gradients, we marginalize the latent state sequence in a *buffer* around each subsequence, propagating critical information from outside each subsequence to its local gradient estimate while avoiding costly full-chain computations. Similar buffering ideas have been previously considered for belief propagation (Gonzalez et al., 2009) and variational inference (Foti et al., 2014), but all are limited to discrete latent states. Here, we present buffering as an approximation to *Fisher's identity* (Cappé et al., 2005), allowing us to naturally extend buffering trick to con-

tinuous and mixed-type latent states. When analytic marginalization is not possible – such as in nonlinear SSMs– we extend our buffered gradient estimators with a modified particle smoother and Gibbs sampling.

We further develop analytic bounds on the bias and MSE of our proposed gradient estimator and analyze how this error propagates to estimating posterior means with SGMCMC. Specifically, we show under mild conditions, that the bias in our gradient estimator decay geometrically in the buffer size. These geometrically decaying bounds guarantee that we only need a small buffer size in practice, allowing scalable inference in SSMs. To obtain these bias bounds we prove that the latent state sequence posterior distribution has an *exponential forgetting* property (Cappé et al., 2005; Del Moral et al., 2010). However unlike classic results which prove a geometric decay between the approximate and exact marginal posterior distributions in total variation distance, we use Wasserstein distance (Villani, 2008) to allow analysis of continuous and mixed-type latent state SSMs. Our approach is similar to proofs of Wasserstein ergodicity in homogeneous Markov chains (Durmus and Moulines, 2015; Madras et al., 2010; Rudolf and Schweizer, 2018); however we extend these ideas to the *nonhomogeneous* Markov chains defined by the latent state sequence posterior distribution.

Although our proposed gradient estimator can be generally applied to any stochastic gradient method, here, we develop SGMCMC samplers for Bayesian inference in a variety of SSMs such as hidden Markov models (HMM), linear Gaussian SSMs (LGSSM), stochastic volatility models (SVM) and switching linear dynamical systems (SLDS) (Cappé et al., 2005; Fox, 2009). We also derive preconditioning matrices to take advantage of information geometry, which allows for more rapid mixing and convergence of our samplers (Girolami and Calderhead, 2011; Patterson and Teh, 2013).

This chapter is organized as follows. First, we present our framework for buffered stochastic gradient estimators for SSMs in Chapter 3.2. Second, we present the error analysis for our proposed method in Chapter 3.3. Third, we present example applications of our proposed method for a variety of different SSMs in Chapter 3.4 Finally, we validate our algorithms and theory on a variety of synthetic and real data experiments, finding that our gradient

estimator can provide orders of magnitude run-time speed ups compared to batch sampling, in Chapter 3.5

## *3.2 Buffered Stochastic Gradients for SSMs*

Our framework for scalable Bayesian inference in SSMs with long observation sequences is to extend SGMCMC to SSMs by developing a gradient estimator that ameliorates the issue of broken temporal dependencies. In particular, we develop a computationally efficient gradient estimator that uses a *buffer* to avoid breaking crucial dependencies, only breaking weak dependencies. We first present a (computationally prohibitive) unbiased estimator of $g(\theta) = \nabla \log p(y \,|\, \theta)$ for SSMs using Fisher's identity. We then derive a general computationally efficient gradient estimate $\hat{g}(\theta)$ that accounts for the dependence in observations using a buffer. We also preconditioning matrices for SGRLD with SSMs. Finally, we present our general SGMCMC pseudocode for SSMs.

### *3.2.1 Unbiased Gradient Estimate*

The main challenge in constructing an efficient estimate $\hat{g}(\theta)$ of $g(\theta)$ for SSMs is handling the lack of independence (marginally) in $y$. Because the observations in SSMs are not independent, we cannot produce an unbiased estimate of $g(\theta)$ with a randomly selected subset of data points as in Eq.(2.10). For example, a naive estimate is to take the gradient of a random contiguous *subsequence* $\mathcal{S} = \{t_1, \ldots, t_S\} \subset 1:T$ with $t_i = t_{i-1} + 1$

$$\hat{g}(\theta) = -\frac{1}{\Pr(\mathcal{S})} \nabla \log p(y_{\mathcal{S}} \,|\, \theta) - \nabla \log p(\theta) \ , \tag{3.1}$$

where $p(y_{\mathcal{S}} \,|\, \theta)$ is computed with $p(u_{t_0}) = p_0(u_{t_0})$. This estimate only requires $\mathcal{O}(S)$ time compared to the $\mathcal{O}(T)$ for $g(\theta)$. However because the marginal likelihood does not factorize as in the independent observations case, this estimate is biased $\mathbb{E}_{\mathcal{S}}[\hat{g}(\theta)] \neq g(\theta)$. In addition, as $\mathcal{S}$ are contiguous subsequences of $1:T$, the scaling factor $\Pr(\mathcal{S})^{-1}$ is no longer correct as time points in the center of $\mathcal{T}$ are sampled more frequently than the endpoints; instead each time point should be scaled point-wise.

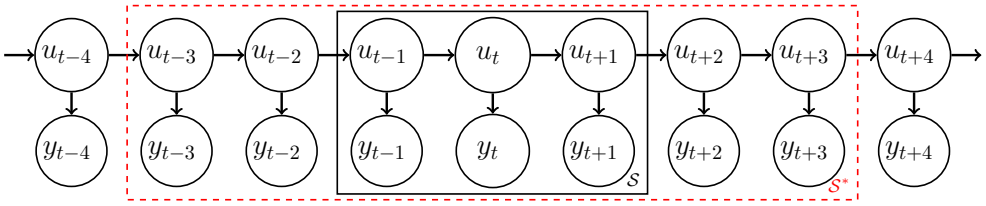


Figure 3.1: Graphical model of a buffered subsequence with $S = 3$ and $B = 2$.

To obtain an unbiased estimate for $g(\theta)$, we use Fisher's identity Eq. (2.24) to rewrite $g(\theta)$ in terms of the complete-data loglikelihood as a sum over time points

$$\begin{aligned} g(\theta) &= -\nabla \log p(y \,|\, \theta) - \nabla \log p(\theta) \\ &= -\mathbb{E}_{\,u|y,\theta} \left[\nabla \log p(y, u \,|\, \theta)\right] - \nabla \log p(\theta) \\ &= -\sum_{t \in 1:T} \mathbb{E}_{\,u|y,\theta} \left[\nabla \log p(y_t, u_t \,|\, u_{t-1}, \theta)\right] - \nabla \log p(\theta) \end{aligned} \tag{3.2}$$

From this, we straightforwardly identify an unbiased estimator for a subsequence $\mathcal{S}$

$$\bar{g}(\theta) = -\sum_{t \in \mathcal{S}} \frac{1}{\Pr(t \in \mathcal{S})} \mathbb{E}_{\,u|y,\theta} \left[\nabla \log p(y_t, u_t \,|\, u_{t-1}, \theta)\right] - \nabla \log p(\theta) \ , \tag{3.3}$$

where $\Pr(t \in \mathcal{S})$ is the probability $t$ is in the random subsequence $\mathcal{S}$.

Although Eq. (3.3) reduces the number of gradient terms to compute from $T$ to $S$, the summation terms require calculating expectations of $u \,|\, y, \theta$. More specifically, Eq. (3.3) requires expectations with respect to the pairwise marginal posteriors $p(u_t, u_{t-1} \,|\, y_{1:T})$ for $t \in \mathcal{S}$. Recall that computing these marginals take $\mathcal{O}(T)$ time to pass messages over the entire sequence $1:T$. This defeats the purpose of using a subsequence. If we instead only pass messages over the subsequence $\mathcal{S}$, then the pairwise marginals are $p(u_t, u_{t-1} \,|\, y_{\mathcal{S}})$ and we return to a biased gradient estimator

$$\hat{g}(\theta) = -\sum_{t \in \mathcal{S}} \frac{1}{\Pr(t \in \mathcal{S})} \mathbb{E}_{\,u|y_{\mathcal{S}},\theta} [\nabla \log p(y_t, u_t \,|\, u_{t-1}, \theta)] - \nabla \log p(\theta) \ . \tag{3.4}$$

### *3.2.2 Approximate Gradient Estimate*

We instead propose passing messages over a *buffered* subsequence $\mathcal{S}^* := \{t_{-B}, \dots, t_{S+B}\}$ for some positive buffer size $B$, with $\mathcal{S} \subset \mathcal{S}^* \subset 1:T$ (see Figure 3.1). The idea is that there exists a large enough $B$ such that $p(u_{\mathcal{S}} \mid y_{\mathcal{S}^*}, \theta) \approx p(u_{\mathcal{S}} \mid y_{1:T}, \theta)$. Our *buffered gradient estimator* sums only over $\mathcal{S}$, but takes expectations over $u_{\mathcal{S}} \mid y_{\mathcal{S}^*}, \theta$ instead of $u_{\mathcal{S}} \mid y_{1:T}, \theta$

$$\hat{g}_{S,B}(\theta) = -\sum_{t \in \mathcal{S}} \frac{1}{\Pr(t \in \mathcal{S})} \cdot \mathbb{E}_{u|y_{\mathcal{S}^*},\theta} \left[\nabla \log p(y_t, u_t \mid u_{t-1}, \theta)\right] - \nabla \log p(\theta) \ , \tag{3.5}$$

where $p(u_{t_{-B-1}}) = p_0(u_{t_{-B-1}})$. When $B = 0$ this is equivalent to the biased estimator $\hat{g}(\theta)$ of Eq. (3.4). When $B = T$ this is equivalent to the unbiased estimator $\bar{g}(\theta)$ of Eq. (3.3).

For a fixed subsequence size $S$, the trade-off between accuracy (bias) and runtime depends on the size of the buffer $B$ and current model parameters $\theta^{(s)}$. Intuitively, when $\theta^{(s)}$ produces pairwise marginals that are similar to i.i.d. data, we can use a small buffer $B$. When $\theta^{(s)}$ produces strongly dependent pairwise marginals, we must use a larger buffer $B$. In Chapter 3.3, we analyze, for a fixed value of $\theta$, how the bias of $\hat{g}_{S,B}(\theta)$ changes with $B$ and its effect on accuracy of SGMCMC. We show a geometric decay in bias

$$\left\| g(\theta) - \mathbb{E}_{\mathcal{S}}[\hat{g}_{S,B}(\theta)] \right\|_2 \leq C_\theta \rho_\theta^{-B} \ , \quad \text{for some } C_\theta > 0 \ , \tag{3.6}$$

where $\rho_\theta$ is large for i.i.d. data and small for strongly dependent data. The term $C_\theta$ depends on the smoothness of $g(\theta)$ and how accurately $p_0(u_{t_{-B-1}})$ approximates $p(u_{t_{-B-1}} \mid y_{1:T \setminus \mathcal{S}^*})$. For a gradient accuracy of $\epsilon$, we only need a logarithmic buffer size $B \propto \log \epsilon^{-1}$.[1] Therefore our buffered gradient estimator reduces the computation time from $\mathcal{O}(T)$ to $\mathcal{O}(S + \log \epsilon^{-1})$. By using buffered stochastic gradients $\hat{g}_{S,B}$ with an appropriate buffer size $B$ in SGMCMC (Eq. (2.9) or (2.12)), we can generate samples $\theta^{(s)}$ that are close to the samples that would be generated if we were to use the unbiased (but intractable) stochastic gradients $\bar{g}$. In our experiments (Chapter 3.5), we find that modest buffers significantly correct for bias.

Our approach is similar to fixed-lag smoothing methods in the particle filter literature (Chan et al., 2016; Del Moral et al., 2017; Olsson et al., 2008), which approximate

[1] As $\epsilon \geq C_\theta \rho_\theta^{-B} \Rightarrow B \geq -\log \epsilon / \log \rho_\theta + \log C_\theta / \log \rho_\theta \Rightarrow B \propto \log \epsilon^{-1}$.

$p(u_t \mid y_{1:T}, \theta)$ using a right buffer $p(u_t \mid y_{1:t+B}, \theta)$ in a streaming fashion. However, our approach, Eq. (3.5), differs by using both a left and a right buffer $p(u_t \mid y_{1:T}, \theta) = p(u_t \mid y_{t-B:t+B})$, which allow us to avoid a full passes over the data.

### *3.2.3 Preconditioning and Fisher Information*

The desirable properties for the preconditioning matrix $D(\theta)$ for SGRLD in Eq. (2.12) are (i) the resulting dynamics takes advantage of the geometric structure of $\theta$, (ii) both $D(\theta)$ and $\Gamma(\theta)$ can be efficiently computed, and (iii) neither $D(\theta)g(\theta)$ nor $\Gamma(\theta)$ are numerically unstable.

The *expected Fisher information* $\mathcal{I}_y$ is the Riemannian metric proposed in (Girolami and Calderhead, 2011)

$$D^{-1}(\theta) = \mathcal{I}_y = \mathbb{E}_{y \mid \theta}\left[\nabla^2 \log p(y \mid \theta)\right] \quad . \tag{3.7}$$

Unfortunately for SSMs, the lack of independence in the marginal likelihood requires a double sum over $1:T$ to compute $\mathcal{I}_y$, which is computationally intractable for long time series. We instead replace $I_y$ with the *complete data Fisher information* $I_{u,y}$

$$\mathcal{I}_{u,y} = \mathbb{E}_{u,y \mid \theta}\left[\nabla^2 \log p(y, u \;\; \theta)\right] = T \cdot \mathbb{E}_{u,y \mid \theta}\left[\nabla^2 \log p(y_t, u_t \mid u_{t-1}, \theta)\right] \quad . \tag{3.8}$$

we use $D(\theta) = I_{u,y}^{-1}$ when $I_{u,y}$ can be calculated analytically and approximations of $I_{u,y}^{-1}$ when it can not not (see Appendix for details). In our experiments, we find that in practice, using preconditioning works well and outperforms vanilla SGLD.

### *3.2.4 Sampling Subsequences*

We now discuss how to sample contiguous stochastic subsequences $\mathcal{S} = \{t_1, \ldots, t_S\}$ for our gradient estimator.

A simple method for sampling subsequences when $T$ is a multiple of $S$ is to sample $\mathcal{S}$ from a strict partition of $\mathcal{T}$. That is if $T/S = L$ is a whole number, then $\Pr(t_1 = t) = 1/L$ for $t \in \{1 + kL \mid k = 0, 1, \ldots, L-1\}$ and

$$\Pr(t \in \mathcal{S}) = \frac{1}{L} = \frac{S}{T} \quad . \tag{3.9}$$

Another method for sampling subsequences is to sample uniformly from all $T - S + 1$ possible contiguous subsequences. That is, $\Pr(t_1 = t) = 1/(T - S + 1)$ for $t \in \{1, \ldots, T - S + 1\}$and $\Pr(t \in \mathcal{S})$ is given by

$$\Pr(t \in \mathcal{S}) = \frac{\min\{t, T - t + 1, S, T - S + 1\}}{T - S + 1} \ . \tag{3.10}$$

We found both methods work well in practice, but found empirically that the latter has reduced variance in the stochastic gradient estimates $\hat{g}(\theta)$; therefore we sample $\mathcal{S}$ using Eq. (3.10) in our experiments.

### *3.2.5 Algorithm Pseudocode*

Algorithm 3 summarizes our buffered stochastic gradient estimator method for SSMs that can be used in SGMCMC (Algorithm 1).[2]

**Algorithm 3** Buffered Stochastic Gradient

Input: data $y$, parameters $\theta$, subsequence length $S$, error tolerance $\epsilon$

$B = \texttt{BufferSize}(\theta, S, \epsilon)$

$\mathcal{S}, \mathcal{S}^* = \texttt{GetBufferedSubsequence}(y, S, B)$ // Eq. (3.10)

$p(u_{\mathcal{S}} \,|\, y_{\mathcal{S}^*}, \theta) = \texttt{ForwardBackward}(y,\, \mathcal{S}^*, \theta)$ // Message Passing

$\tilde{g}(\theta) = -\sum_{t \in \mathcal{S}} \frac{1}{\Pr(t \in \mathcal{S})} \mathbb{E}_{\, u_{\mathcal{S}} | y_{\mathcal{S}^*}, \theta}[\nabla_\theta \log p(y_t, u_t | u_{t-1})]$ // Eq. (3.5)

Return $\tilde{g}(\theta)$

To select the buffer size $B$ in Algorithm 3, we choose $B$ large enough such that the error using $B$ and a larger buffer size $B^*$ is small

$$B = \min \left\{ \hat{B} \in [0, B^*] \,:\, \mathbb{E}_{\mathcal{S}} \| \, \hat{g}_{S,\hat{B}}(\theta) - \hat{g}_{S,B^*}(\theta) \| < \epsilon \right\} \tag{3.11}$$

where $\epsilon$ is some error tolerance and the expectation over $\mathcal{S}$ is approximated with an empirical average over $N$ subsequences. Eq. (3.11) uses $\hat{g}_{S,B^*}(\theta)$ as a proxy for $\hat{g}_{S,T}(\theta)$. As the error

[2]Python code for our methods for SSMs is available at `https://github.com/aicherc/sgmcmc_ssm_code`

decays geometrically (Chapter 3.3), we found using $B^* = 100$ was conservative in practice. Calculating $B$ using Eq. (3.11) at every iteration for a new $\theta^{(s)}$ is impractical; therefore for our experiments, we use a fixed $B$, estimated using $\theta$ from a pilot run with $B = B^*$ and $N_S = 1000$. In addition, instead of evaluating each $\hat{B}$ in $[0, B^*]$, we can estimate the required $B$ for a target error tolerance $\epsilon$ after estimating the error $\hat{\epsilon}$ of a single $\hat{B}$, by taking advantage of the geometric error scaling rate, Eq. (3.6), to obtain $B = \hat{B} + \log_{\rho_\theta}(\hat{\epsilon}/\epsilon)$ where $\rho_\theta$ is a bound on the geometric decay rate from theory.

### *3.2.6 Extension for nonlinear SSMs*

We now discuss how to extend the buffered stochastic gradient estimator (3.5) to nonlinear SSMs, where the expectation over latent variables can not be computed analytically. We approximate the expectation using the particle smoothers described in Chapter 2.2.3.

To compute our *particle buffered stochastic gradient* $g^{\mathrm{PF}}_{S,B,N}$, we approximate the expectation over $p(u|y_{\mathcal{S}^*}, \theta)$ in Eq. (3.5) using Algorithm 2 over $\mathcal{S}^*$ with $p(u_{t_{-B-1}}) = p_0(u_{t_{-B-1}})$. Specifically, the expectation decomposes into a sum of pairwise statistics

$$\phi = \sum_{t \in \mathcal{S}^*} \phi_t(u_t, u_{t-1}) \ , \tag{3.12}$$

where

$$\phi_t(u_t, u_{t-1}) = \begin{cases} \dfrac{\nabla_\theta \log p(u_t, y_t \mid u_{t-1}, \theta)}{\Pr(t \in \mathcal{S})} & \text{if } t \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

We emphasize that the statistic $\phi_t$ is zero for $t$ in the left and right buffers $\mathcal{S}^* \backslash \mathcal{S}$. Although cumulative sum is not updated by $\phi_t$ for $t$ in the buffer $\mathcal{S}^* \backslash \mathcal{S}$, running the particle smoother over the buffer is *crucial* to reducing the bias.

Algorithm 4 summarizes our particle buffered stochastic gradient estimate $g^{\mathrm{PF}}_{S,B,N}$. In addition to the normal speedup gains from using a subsequence over a batch, our method also reduces the number of particles required to control the variance of the particle smoother as the particle smoother is only run over $\mathcal{S}^*$, which has length $S + 2B \ll T$. This allows us to

approximate $\hat{g}_{S,B}$ with $g^{\mathrm{PF}}_{S,B,N}$ using a modest number of particles $N$ – avoiding the particle degeneracy and full sequence runtime bottlenecks.

**Algorithm 4** Particle Buffered Stochastic Gradient

1: Input: data $y_{1:T}$, parameters $\theta$, subsequence size $S$, buffer size $B$, particle size $N$
2: $\mathcal{S}, \mathcal{S}^* = \texttt{GetBufferedSubsequence}(y, S, B)$ // Eq. (3.10)
3: Calculate $g^{\mathrm{PF}}$ over $\mathcal{S}^*$ using Algorithm 2 on Eq. (3.13). // Particle Smoother
4: Return $g^{\mathrm{PF}}$

## *3.3 Error Analysis*

In this section, we analyze the error of our buffered stochastic gradient $\hat{g}_{S,B}$ and the effect of this error on our SGMCMC schemes.

We first present error bounds for approximating posterior means using SGLD with biased gradients (Theorem 1). We then present bounds on the gradient bias and MSE of $\hat{g}_{S,B}$ (Theorem 2). To do so, we prove an error bound between $\hat{g}_{S,B}$ and $\bar{g} = \hat{g}_{S,T}$ that decays geometrically in $B$ (Theorem 3) and bound this geometric rate for a wide class of SSMs (Theorem 5). Finally, we discuss extensions of these bounds when the expectation over the latent variables in Eq. (3.5) must be approximated, such as for $g^{\mathrm{PF}}_{S,B,N}$ (Theorem 6).

### *3.3.1 Error of Biased SGLD's Finite Sample Averages*

We consider the estimation error of the posterior expected value of some test function of the parameters $\phi : \Theta \to \mathbb{R}$ using samples $\theta^{(k)}$ drawn using SGLD with a fixed stepsize $h$ and stochastic gradients $\hat{g}(\theta)$ from Chapter 2.1.2.

As first presented in Eq. (2.11), $\bar{\phi}$ is the posterior expected value of $\phi$ and $\hat{\phi}_{K,h}$ is the $K$-sample estimator for $\bar{\phi}$ after running SGLD with a fixed stepsize $h$

$$\bar{\phi} = \mathbb{E}_{p(\theta|y)}[\phi(\theta)] \quad \text{and} \quad \hat{\phi}_{K,h} = \frac{1}{K}\sum_{k=1}^{K} \phi(\theta^{(k)}) \ . \tag{3.14}$$

Recall the error of the *finite sample average* $|\hat{\phi}_{K,h} - \bar{\phi}|$ has been previously studied for SGLD with *unbiased* gradients by Vollmer et al. (2016) and Chen et al. (2015a). We now present Theorem 1, which bounds the error of finite sample average of SGLD when the stochastic gradients $\hat{g}_\theta$ are potentially *biased*.

**Theorem 1** (Error of Finite Sample Average)**.** *If the gradient $g$ is smooth in $\theta$, the test function $\phi$ satisfies a bounded moment condition (described in Appendix A.1.1) and the bias and MSE of the stochastic gradient estimator $\hat{g}$ is uniformly bounded, that is,*

$$\|\mathbb{E}\,\hat{g} - g\| \leq \delta \text{ and } \mathbb{E}\,\|\hat{g} - g\|^2 \leq \sigma^2 \text{ for all } \theta \ , \tag{3.15}$$

*then the bias and MSE of $\hat{\phi}_{K,h}$ satisfy*

$$|\mathbb{E}\,\hat{\phi}_{K,h} - \bar{\phi}\,| = \mathcal{O}\left(\frac{1}{Kh} + h + \delta\right) \ , \tag{3.16}$$

$$\mathbb{E}\,|\hat{\phi}_{K,h} - \bar{\phi}\,|^2 = \mathcal{O}\left(\frac{1}{Kh} + h^2 + \frac{\sigma^2}{K} + \delta^2 + \frac{\delta}{h} + \delta h\right) \ . \tag{3.17}$$

The bias bound, Eq. (3.16), is a direct application of Theorem 2 in Chen et al. (2015a). The MSE bound, Eq. (3.17), is an extension of Theorem 3 in Chen et al. (2015a) when the stochastic gradient estimates $\hat{g}(\theta)$ are biased (i.e. $\delta \neq 0$). The additional bias terms $\delta$ arise from keeping track of additional cross terms in $(\hat{\phi}_{K,h} - \bar{\phi})^2$. The proof of Theorem 1 is presented in Appendix A.1.1.

From Theorem 1, we see that the error bounds on $\hat{\phi}_{K,h}$ are more sensitive to the bias $\delta$ of $\hat{g}$ than the variance $\sigma^2$: the term involving $\sigma^2$ decays with increasing $K$, while terms involving $\delta$ do not decay regardless of stepsize $h$ or number of samples $K$. Therefore, for the samples from SGLD to be useful for estimating $\bar{\phi}$, it is important for the bias of $\hat{g}$ to be small.

### *3.3.2 Bias and MSE Bounds for SSM Stochastic Gradients*

We now develop bounds on the bias $\delta$ and MSE $\sigma^2$ of our buffered stochastic gradients $\hat{g}_{S,B}$, which allow us to apply Theorem 1 with $\hat{g} = \hat{g}_{S,B}$.

**Theorem 2** (Bias and MSE Bounds for $\hat{g}_{S,B}$)**.** *For fixed $\theta$, if the model's forward and backward smoothing kernels are Lipschitz in Wasserstein distance, the gradient of the complete-data loglikelihood is Lipschitz in $u$, and there is a bound on the variance and autocorrelation of terms in Fisher's identity (see Appendix A.1.2), then the bias $\delta$ and MSE $\sigma^2$ of $\hat{g}_{S,B}$ are bounded by*

$$\delta \leq \zeta \cdot \left[C_1 (L_\theta)^B\right] \quad , \tag{3.18}$$

$$\sigma^2 \leq 2\zeta^2 \cdot \left[C_1^2 (L_\theta)^{2B} + C_2 S\right] \quad , \tag{3.19}$$

*where $L_\theta, C_1, C_2$ are constants with respect to $S, B, T$ depending on $\theta$ and $\zeta = \max_t \Pr(t \in \mathcal{S})^{-1}$.*

From Theorem 2, we see that the bias $\delta$ (Eq. (3.18)) can be controlled by selecting large enough $B$. The assumptions for Theorem 2 determine the constants $L_\theta, C_1, C_2$, which are discussed below. Explicit forms of the constants $L_\theta, C_1, C_2$ are detailed in Chapter 3.4 for select SSMs.

We now sketch the proof of Theorem 2 and discuss its assumptions. The complete proof can be found in Appendix A.1.2. We decompose the error between $\hat{g}_{S,B}(\theta)$ and the full gradient $g(\theta)$ through $\bar{g}(\theta) = \hat{g}_{S,T}(\theta)$ into two error sources:

$$\|\hat{g}_{S,B}(\theta) - g(\theta)\| \leq \underbrace{\|\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\|}_{\text{buffering error (I)}} + \underbrace{\|\hat{g}_{S,T}(\theta) - g(\theta)\|}_{\text{subsequence error (II)}}. \tag{3.20}$$

(I) *Buffering error*: error in approximating the latent state posterior $p(u_{1:T}|y_{1:T})$ with $p(u_{1:T}|y_{\mathcal{S}^*})$. In Chapter 3.3.3, we show that if the smoothing kernels $\{\vec{\Psi}_t, \overleftarrow{\Psi}_t\}$ are contractions for all $t$ (i.e. $L_\theta < 1$) and the gradient of the complete-data loglikelihood is Lipschitz in $\theta$, then the error in this term is proportional to $\zeta \cdot (L_\theta)^B$ (Theorem 3). In Chapter 3.3.4, we show sufficient conditions for $L_\theta < 1$ (Theorem 5).

(II) *Subsequence error*: the error in approximating Fisher's identity with a stochastic subsequence. The error in this term depends on the subsequence size $S$ and how

subsequences are sampled. Because we sample random *contiguous* subsequences of size $S$, the MSE scales $\mathcal{O}(\zeta^2 \cdot S/(1-\rho)$, where $\rho$ is a bound on the autocorrelation between terms (see Lemma 4 in Appendix A.1.2 for details).

Combining these error bounds gives us Theorem 2.

We present examples of the asymptotic bias and MSE bounds given by Theorem 2 for four different gradient estimators in Table 3.1. The four gradient estimators are: (i) naive stochastic subsequence (without buffering) $\hat{g}_{S,0}$ (ii) buffered stochastic subsequence $\hat{g}_{S,B}$, (iii) unbiased stochastic subsequence $\hat{g}_{S,T} = \bar{g}$, and (iv) full sequence $\hat{g}_{T,T} = g$. For simplicity, we assume the subsequences $\mathcal{S}$ are sampled from a *strict* partition of $1:T$ such that $\zeta = T/S$ and assume $B$ is on the same order as $S$ (i.e. $B$ is $\mathcal{O}(S)$).

Table 3.1: Asymptotic bias, MSE, and compute cost for four different gradient estimators.

| Gradient | $(S, B)$ | Bias $\delta$ | MSE $\sigma^2$ | Compute |
|---|---|---|---|---|
| Naive Stochastic Subsequence | $(S, 0)$ | $\mathcal{O}(T/S)$ | $\mathcal{O}(T^2/S)$ | $\mathcal{O}(S)$ |
| Buffered Stochastic Subsequence | $(S, B)$ | $\mathcal{O}((L_\theta)^B \cdot T/S)$ | $\mathcal{O}(T^2/S)$ | $\mathcal{O}(S)$ |
| Fully Buffered Subsequence | $(S, T)$ | 0 | $\mathcal{O}(T^2/S)$ | $\mathcal{O}(T)$ |
| Full Sequence | $(T, T, N)$ | 0 | 0 | $\mathcal{O}(T)$ |

From Table 3.1, we see that without buffering, the naive stochastic gradient has a $T/S$ term in the bias bound $\delta$. The fully buffered subsequence and full sequence gradients remove the bias entirely at the cost of $\mathcal{O}(T)$ computation instead of $\mathcal{O}(S)$. Instead, our proposed buffered stochastic gradient controls the bias, with the geometrically decaying factor $(L_\theta)^B$, using only $\mathcal{O}(S)$ computation.

### *3.3.3 Error Bounds for Buffering*

In this section, we establish a bound on the expected error between the unbiased gradient estimator $\bar{g}(\theta) = \hat{g}_{S,T}(\theta)$ and our buffered gradient estimator $\hat{g}_{S,B}(\theta)$, which is necessary in

proving Theorem 2.

Our approach is to bound $\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$ in terms of the Wasserstein distance between the exact posterior $\gamma_t(u_t) = p(u_t \,|\, y_{1:T}, \theta)$ and our approximate posterior $\widetilde{\gamma}_t(u_t) = p(u_t \,|\, y_{\mathcal{S}^*}, \theta)$ and then show this Wasserstein distance decays geometrically as $B$ increases. To bound the Wasserstein distance, we follow existing work on bounding Markov processes in Wasserstein distance (Durmus and Moulines, 2015; Madras et al., 2010; Rudolf and Schweizer, 2018). However, unlike previous work that focuses on the homogeneous Markov process of the joint model $\{u, y \,|\, \theta\}$, we instead focus on the induced *nonhomogeneous* Markov process of the conditional model $\{u \,|\, y, \theta\}$. To do so, we use the forward ($\vec{\psi}_t$) and backward ($\overleftarrow{\psi}_t$) *random maps* of $\{u \,|\, y, \theta\}$ (Diaconis and Freedman, 1999)

$$u_t \sim p(u_t \,|\, y, \theta) \;\; \Rightarrow \;\; (\vec{\psi}_t(u_t), u_t) \sim p(u_{t+1}, u_t \,|\, y, \theta) \tag{3.21}$$

$$u_t \sim p(u_t \,|\, y, \theta) \;\; \Rightarrow \;\; (\overleftarrow{\psi}_t(u_t), u_t) \sim p(u_{t-1}, u_t \,|\, y, \theta) \;\; , \tag{3.22}$$

If $\vec{\psi}_t$ and $\overleftarrow{\psi}_t$ satisfy a contractive property, then we can bound the Wasserstein distance between $\gamma_t, \widetilde{\gamma}_t$ in terms of $\gamma_{t-1}, \widetilde{\gamma}_{t-1}$ and $\gamma_{t+1}, \widetilde{\gamma}_{t+1}$ respectively (e.g. Eq. (2.32) in Chapter 2.2.4). Bounding the error of the induced nonhomogeneous Markov process has been previously studied in the SSM literature using total variation (TV) distance (Cappé et al., 2005; Del Moral et al., 2010; Le Gland and Mevel, 2000; Tong and Van Handel, 2012). These works bound the error in total variation distance by quantifying how quickly the smoothed posterior forgets the initial condition. However, these bounds typically require stringent regularity conditions, which are hard to prove outside of finite or compact spaces[3]. In particular, these bounds are not immediately applicable for LGSSMs. In contrast, we bound the error in Wasserstein distance by proving contraction properties of $\vec{\psi}_t$ and $\overleftarrow{\psi}_t$, allowing us to handle continuous and mixed-type SSMs (see Chapter 3.3.4).

Our main result is that if, for each fixed $\theta$, the gradient of $\log p(y, u \,|\, \theta)$ satisfies a Lipschitz condition and the random maps $\{\vec{\psi}_t, \overleftarrow{\psi}_t\}_{t \in \mathcal{S}^*}$ all satisfy a contraction property, then the

[3]These bounds have been extended to non-compact spaces for the *filtered* posterior, when the SSM satisfies a multiplicative drift condition (Whiteley, 2013).

error $\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$ decays geometrically in the buffer size $B$. For our analysis, we first consider the simple case of uniformly sampling a single sequence from $T/S$ separate subsequences (i.e. $\Pr(t \in \mathcal{S}) = S/T$ for all $t$) and assume the prior $p_0$ is stationary (i.e. $p_0(u_t) = \int p(u_t|u_{t-1})p_0(u_{t-1})du_{t-1}$).

**Theorem 3.** *Let $\epsilon_\rightarrow$ and $\epsilon_\leftarrow$ be the 1-Wasserstein distances between $\gamma_t$ and $\widetilde{\gamma}_t$ at the left and right ends of $\mathcal{S}^*$ respectively. Let $\epsilon_1 = \max_{\mathcal{S}^* \subset 1:T}\{\epsilon_\rightarrow, \epsilon_\leftarrow\}$. If the gradients of $\log p(y_t, u_t \mid u_{t-1}, \theta)$ are all Lipschitz in $u_{t-1:t}$ with constant $C_\theta$, and random maps $\vec{\psi}_t$ and $\overleftarrow{\psi}_t$ are all Lipschitz in $u_t$ with constant $L_\theta < 1$, then we have*

$$\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2 \leq T \cdot C_\theta \cdot \frac{1 + L_\theta}{1 - L_\theta} \cdot \frac{1 - (L_\theta)^S}{S} \cdot (L_\theta)^B \cdot 2\epsilon_1 \ . \tag{3.23}$$

A similar result for when the gradient of the complete data loglikelihood is Lipschitz in $uu^T$ instead of $u$ (as needed for LGSSM) is presented in Theorem 4 at the end of this section.

As $L_\theta < 1$, Theorem 3 states that the error of the buffered gradient estimator decays geometrically as $\mathcal{O}((L_\theta)^B)$. Therefore, the required buffer size $B$ for an error tolerance of $\delta$ scales logarithmically as $\mathcal{O}(\log \delta^{-1})$. In contrast, the error of the gradient estimator decays only linearly in the subsequence length, $\mathcal{O}(S^{-1})$; therefore much longer subsequences, $\mathcal{O}(\delta^{-1})$, are required to reduce bias. This agrees with the intuition that the bias is dominated by the error at the endpoints of subsequence.

Theorem 3 requires bounding the Lipschitz constants of the gradient of the complete data loglikelihood and the random maps $\vec{\psi}_t, \overleftarrow{\psi}_t$ given the parameters $\theta$ and observations $y_{1:T}$. Theorem 3 also depends on the maximum Wasserstein distance $\epsilon_1$ between $\gamma_t$ and $\widetilde{\gamma}_t$ for all $t \in 1:T$, which is finite.

We now briefly discuss relaxations of the assumptions on $\Pr(t \in \mathcal{S})$ and $p_0$. If the contiguous subsequences are not sampled from a strict partition (i.e. $\Pr(t \in \mathcal{S}) \neq S/T$ for all $t$), then we can replace the factor of $T/S$ in Theorems 3 and 4 with $\max_t \Pr(t \in \mathcal{S})^{-1}$. If the initial distribution for $u_{t_{-B-1}}$ of our buffered stochastic gradient, $p_0$, is not stationary, then our approximate posterior over the latent states $\tilde{\gamma}_t(u_t)$ is not equal to $p(u_t \mid y_{\mathcal{S}^*}, \theta)$. However Theorems 3 and 4 will still apply; the choice of initial distribution only affects the

Wasserstein distance between $\gamma_t, \tilde{\gamma}_t$ and therefore the terms $\epsilon_1, \epsilon_2$ in the Theorems. In fact, the optimal initial distribution is $p(u_{t_{-B}}|y_{1:T\setminus\mathcal{S}^*})$, which minimizes the Wasserstein distance of $\gamma, \tilde{\gamma}$.

Finally, we now present a similar result for when $\nabla \log p(y, u_t \,|\, u_{t-1}\theta)$ is Lipschitz in $uu^T$, which relies on bounding the 1-Wasserstein distance of $uu^T$ in terms of the 2-Wasserstein distance of $u$.

**Theorem 4.** *Let $\epsilon_\rightarrow$ and $\epsilon_\leftarrow$ be the 2-Wasserstein distances between $\gamma_t$ and $\widetilde{\gamma}_t$ at the left and right ends of $\mathcal{S}^*$ respectively. Let $\epsilon_2 = \max_{\mathcal{S}^*\subset 1:T}\{\epsilon_\rightarrow, \epsilon_\leftarrow\}$. If the gradients are Lipschitz in $uu^T$ with constant $C_\theta$, and random maps $\vec{\psi}_t$ and backwardmap$_t$ are all Lipschitz in $u_t$ with constant $L_\theta < 1$, then with $C'_\theta = (2\sqrt{\mathbb{E}_\gamma \|u\|_2^2} + 1)C_\theta$*

$$\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2 \leq T \cdot C'_\theta \cdot \frac{\sqrt{1+(L_\theta)^2}}{1-(L_\theta)^{1/2}} \cdot \frac{1-(L_\theta)^{S/4}}{S/2} \cdot (L_\theta)^{B/2} \cdot \max_{r\in\{1/2,\,1\}} (2\epsilon_2)^r \,.$$

Similar to Theorem 3, Theorem 4 states that the squared error of the buffered gradient estimator decays geometrically if the complete-data loglikelihood is Lipschitz in $uu^T$ instead of $u$. However, the price we pay is a square-root: the error decays $\mathcal{O}((L_\theta)^{B/2})$ instead of $\mathcal{O}((L_\theta)^B)$.

Complete details for the proofs of Theorems 3 and 4 can be found in Appendix A.1.3.

### *3.3.4 Smoothing Kernel Lipschitz Constant Bounds*

Theorems 3 and 4 require the Lipschitz constant $L_\theta$ of smoothing kernels $\{\vec{\Psi}_t, \overleftarrow{\Psi}_t\}$ to be less than 1. We now discuss how to calculate these bounds. For discrete latent states, these bounds immediately follow from existing *Dobrushin coefficient* analysis; however for continuous latent states this can be nontrivial. In particular, for nonlinear SSMs, the forward and backward smoothing kernel $\vec{\Psi}_t$ and $\overleftarrow{\Psi}_t$ are not available in closed form. To obtain bounds for nonlinear SSMs, we derive Theorem 5 for logconcave models, which bounds the Lipschitz constants of the smoothing maps to the prior and filtering kernels.

*Discrete Latent Variable SSMs*

In the finite discrete variable case, conditions for bounding the Lipschitz coefficient of the smoothing kernels $\vec{\Psi}_t, \overleftarrow{\Psi}_t$ (as needed for Chapter A.1.3) are equivalent to conditions for bounding their *Dobrushin coefficients* (Cappé et al., 2005; Del Moral et al., 2010). The Dobrushin coefficient for a transition kernel $\mathcal{Q}$ is

$$\delta(\mathcal{Q}) = \sup_{z,z'} \frac{1}{2} \|\mathcal{Q}(z,\cdot) - \mathcal{Q}(z',\cdot)\|_{TV} = \frac{\|\mathcal{Q}(z,\cdot) - \mathcal{Q}(z',\cdot)\|_{TV}}{\|\delta_z - \delta_{z'}\|_{TV}} \ . \tag{3.24}$$

The final term of Eq. (3.24) show the connection between Dobrushin coefficients and Lipschitz coefficients: it is the ratio of the distance of between kernels $\mathcal{Q}(z,\cdot), \mathcal{Q}(z',\cdot)$ with the distance between point masses at $z$ and $z'$. Therefore for discrete latent states, $L_\theta = \max_t \max\{\delta(\vec{\Psi}_t, \delta(\overleftarrow{\Psi}_t)\}$.

In the discrete case, sufficient conditions for $L_\theta < 1$ are well known (see Cappé et al. (2005) Chapter 4.3). If the transition matrix $\Pi$ satisfies the *strong mixing condition*, that is, there exists constants $\sigma^-$ and $\sigma^+$ with $0 < \sigma^- \leq \sigma^+$ and a probability distribution $\kappa \in \Delta^K$ over $z$ such that $\sigma^- \kappa(z') \leq \Pi_{z,z'} \leq \sigma^+ \kappa(z')$ and $\mathbb{E}_\kappa[p(y \,|\, z)] < \infty$, then the Dobrushin coefficients are bounded by $L = 1 - \sigma^-/\sigma^+$. Relaxations of this condition can be found in (Cappé et al., 2005; Del Moral et al., 2010). Alternatively, we can obtain tighter bounds for HMMs via estimating the Lyapunov exponents for the underlying random dynamical systems defined by random maps $\vec{\psi}_t$ and $\overleftarrow{\psi}_t$ (Ye et al., 2017; Ma et al., 2017).

Finally, the Lipschitz constant $C_\theta$ for Lemma 5 is

$$C_\theta = \max_{t \in \mathcal{S}, z_t, z_t'} \|\nabla \log p(y_t, z_t \,|\, z_{t-1}, \theta) - \nabla \log p(y_t, z_t' \,|\, z_{t-1}', \theta)\| \ . \tag{3.25}$$

This is easy to compute since at each iteration $y$ and $\theta = \theta^{(s)}$ are fixed. Given these bounds on $L_\theta$, we can use Theorem 3 to select the buffer size $B$ to ensure approximate convergence to the stationary distribution.

*Continuous Latent Variable SSMs*

For continuous SSMs, we can calculate $L_\theta$ from the average Lipschitz constant of $\vec{\psi}_t$ and $\overleftarrow{\psi}_t$ when they are available in closed-form (e.g., LGSSM). However for nonlinear SSMs, the smoothing kernels $\vec{\Psi}_t, \overleftarrow{\Psi}_t$ are not available in closed-form and therefore directly bounding the Lipschitz constant is difficult. We now show that when the model's transition and emission densities are *log-concave* in $u_t, u_{t-1}$, we can bound the Lipschitz constant of $\vec{\Psi}_t, \overleftarrow{\Psi}_t$ in terms of the Lipschitz constant of either the *prior kernels* $\vec{\Psi}_t^{(0)}, \overleftarrow{\Psi}_t^{(0)}$, or the *filtered kernels* $\vec{\Psi}_t^{(1)}, \overleftarrow{\Psi}_t^{(1)}$

$$\begin{aligned} \vec{\Psi}_t^{(0)} &:= p(u_t \,|\, u_{t-1}, \theta), & \vec{\Psi}_t^{(1)} &:= p(u_t \,|\, u_{t-1}, y_t, \theta), \\ \overleftarrow{\Psi}_t^{(0)} &:= p(u_t \,|\, u_{t+1}, \theta), & \overleftarrow{\Psi}_t^{(1)} &:= p(u_t \,|\, u_{t+1}, y_t, \theta), \end{aligned} \tag{3.26}$$

Unlike the smoothing kernels, the prior kernels are typically defined by the model and are therefore available in closed form. If the filtered kernels are available, then they can be used to obtain even tighter bounds.

**Theorem 5** (Lipschitz Kernel Bound)**.** *Assume the prior for $u_0$ is log-concave in $u$. If the transition density $p(u_t \,|\, u_{t-1}, \theta)$ is log-concave in $(u_t, u_{t-1})$ and the emission density $p(y_t \,|\, u_t)$ is log-concave in $u_t$, then*

$$\|\vec{\Psi}_t\|_{Lip} \;\le\; \|\vec{\Psi}_t^{(1)}\|_{Lip} \;\le\; \|\vec{\Psi}_t^{(0)}\|_{Lip} \tag{3.27}$$

$$\|\overleftarrow{\Psi}_t\|_{Lip} \;\le\; \|\overleftarrow{\Psi}_t^{(1)}\|_{Lip} \;\le\; \|\overleftarrow{\Psi}_t^{(0)}\|_{Lip}. \tag{3.28}$$

*Therefore*

$$L_\theta = \max_t\{\|\vec{\Psi}_t\|_{Lip}, \|\overleftarrow{\Psi}_t\|_{Lip}\} \le \max_t\{\|\vec{\Psi}_t^{(1)}\|_{Lip}, \|\overleftarrow{\Psi}_t^{(1)}\|_{Lip}\} \le \max_t\{\|\vec{\Psi}_t^{(0)}\|_{Lip}, \|\overleftarrow{\Psi}_t^{(0)}\|_{Lip}\} \tag{3.29}$$

This theorem lets us bound $L_\theta$ with the Lipschitz constant of either the prior kernels or filtered kernels. The proof of Theorem 5 is provided in Appendix A.1.4 and uses Caffarelli's log-concave perturbation theorem (Villani, 2008; Colombo et al., 2015). Examples of SSMs that are log-concave include LGSSMs, SVMs, or any linear SSM with log-concave transition

and emission distributions. Examples of SSMs that are not log-concave include the GARCH model or any linear SSM with a transition or emission distribution that is not log-concave (e.g. Student's $t$).

Theorem 5 lets us calculate analytic bounds on $L_\theta$ for the buffering error of Theorems 2, 3 and 4, which we use for the models in Chapter 3.4.

### *3.3.5 Error Analysis for Non-Analytic Message Passing*

For the buffered stochastic gradient $\hat{g}_{S,B}$, Eq. (3.5), if we cannot calculate the expectation over the latent variables $u_{\mathcal{S}}|y_{\mathcal{S}^*}, \theta$ in closed form – that is analytic message passing is not tractable – then approximating this expectation (e.g. with a particle smoother) introduces additional bias and variance into the gradient estimate. We now discuss extending Theorem 2 when we approximate the expectation.

Consider the error in between the particle buffered stochastic gradient $g^{\mathrm{PF}}_{S,B,N}(\theta)$ and the true gradient $g(\theta)$. The error decomposes into three terms. Compared to Eq. (3.20), there is an additional *particle error* term.

$$\|g^{\mathrm{PF}}_{S,B,N}(\theta) - g(\theta)\| \leq \underbrace{\|g^{\mathrm{PF}}_{S,B,N}(\theta) - \hat{g}_{S,B}(\theta)\|}_{\text{particle error (III)}} + \underbrace{\|\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\|}_{\text{buffering error (I)}} + \underbrace{\|\hat{g}_{S,T}(\theta) - g(\theta)\|}_{\text{subsequence error (II)}}. \quad (3.30)$$

(III) *Particle error*: the Monte Carlo error of the particle smoother. From Kantas et al. (2015), the asymptotic bias and MSE of a particle approximation to the sum of $R$ test functions (using Algorithm 2) is $\mathcal{O}(R/N)$ and $\mathcal{O}(R^2/N)$ respectively. Since $g^{\mathrm{PF}}(S, B, N)$ is a particle approximation to the sum of $R = S + 2B$ test functions (i.e., $\phi_t(u_t, u_{t-1})$), we have

$$\begin{aligned} \|\mathbb{E}\, g^{\mathrm{PF}}_{S,B,N} - \hat{g}_{S,B}\| &= \mathcal{O}\left(\gamma \cdot \frac{S+2B}{N}\right), \\ \mathbb{E}\, \|g^{\mathrm{PF}}_{S,B,N} - \hat{g}_{S,B}\|^2 &= \mathcal{O}\left(\gamma^2 \cdot \frac{(S+2B)^2}{N}\right). \end{aligned} \quad (3.31)$$

where $\gamma$ is a upper bound on the sampling scale factor $\gamma = \max_t \Pr(t \in \mathcal{S})^{-1}$.

Using a more advanced particle filter, such as the PaRIS algorithm, Corollary 6 of Olsson and Westerborn (2017) gives a tighter bound for the MSE

$$\mathbb{E}\,\|g^{\mathrm{PF}}_{S,B,N} - \hat{g}_{S,B}\|^2 = \mathcal{O}\left(\gamma^2 \cdot \frac{S+2B}{N}\right) \quad .$$

However in our experiments, we found that the improved MSE of PaRIS was not worth the additional computational overhead of the PaRIS algorithm for the small subsequences we considered $S + 2B \lesssim 100$ (see Chapter 3.5.2).

Including this error term to Theorem 2 gives us Theorem 6.

**Theorem 6** (Bias and MSE Bounds for $g^{\mathrm{PF}}_{S,B,N}$)**.** *Under the same conditions as Theorem 2, the bias $\delta$ and MSE $\sigma^2$ of $g^{\mathrm{PF}}_{S,B,N}$ is bounded by*

$$\delta \leq \gamma \cdot \left[C_1 (L_\theta)^B + \mathcal{O}\left(\frac{S+2B}{N}\right)\right] \quad , \tag{3.32}$$

$$\sigma^2 \leq 3\gamma^2 \cdot \left[C_1^2 (L_\theta)^{2B} + C_2 S + \mathcal{O}\left(\frac{(S+2B)^2}{N}\right)\right] \quad , \tag{3.33}$$

*where $\gamma = \max_t \Pr(t \in \mathcal{S})^{-1}$ and $L_\theta, C_1, C_2$ are constants with respect to $S, B, N$.*

From Theorem 6, we see that the bias $\delta$ can be controlled by selecting large enough $N$ and $B$. In practice, we found that moderate $N$ was sufficient for the small subsequences sizes $|\mathcal{S}^*| = 2S + B$ we considered.

### *3.4 Applications to Models*

In this section, we provide examples of how to apply the stochastic gradient framework of Chapter 3.2 and error bounds of Chapter 3.3 to common SSMs: discrete latent variable SSMs (Gaussian HMM, ARHMM), continuous latent variable SSMs (LGSSM, SVM, GARCH), and mixed latent variable SSMs (SLDS).

### *3.4.1 Gaussian HMM*

We consider discrete latent state HMMs with Gaussian emissions introduced in Chapter 2.2.1. The complete data likelihood of a Gaussian HMM is as follows

$$p(y, z \mid \theta) = \prod_{t=1}^{T} \Pi_{z_{t-1}, z_t} \cdot \mathcal{N}(y_t \mid \mu_{z_t}, \Sigma_{z_k}) \ , \tag{3.34}$$

where $y_t \in \mathbb{R}^m$ are the observations, $u_t \equiv z_t \in \{1, \ldots, K\}$ are the discrete latent variables, and $\theta = \{\Pi, \mu, \Sigma\}$ are the parameters with $\Pi_k \in \Delta^K$ (simplex over $K$ states), $\mu_k \in \mathbb{R}^m$, $\Sigma_k \in \mathbb{S}_+^m$ (positive definite matrices) for $k = 1, \ldots, K$. In practice, we use the *expanded mean* parameters of $\Pi$ instead of $\Pi$ (as in (Patterson and Teh, 2013)) and the *Cholesky decomposition* of $\Sigma_k^{-1}$ instead of $\Sigma_k$ to ensure positive definiteness. As the latent states are discrete over a finite space, the forward backward algorithm for an HMM can be done in closed-form; thus, pairwise latent marginals $\gamma_{t-1:t}(z_{t-1}, z_t)$, gradients $\nabla U(\theta)$ and preconditioning terms $D(\theta)$ and $\Gamma(\theta)$ are straightforward to calculate.

**Forward Backward** The forward and backward recursions (Eqs. (2.25) and (2.26)) for an HMM are

$$\alpha_t := p(z_t, y_{\leq t}) = \alpha_{t-1} \cdot \Pi \cdot P_t \tag{3.35}$$

$$\beta_t := p(y_{>t} \mid z_t) = \Pi \cdot P_{t+1} \beta_{t+1} \ , \tag{3.36}$$

where $\alpha_{-T} = \mathbf{1}/K$, $\beta_T = \mathbf{1}$, and

$$P_t := \operatorname{diag}\{\mathcal{N}(y_t \mid \mu_k, \Sigma_k)\}_{k=1}^K \ . \tag{3.37}$$

Given the messages $\alpha_t, \beta_t$, the marginal and pairwise posteriors of the latent states are computed as

$$\gamma_t(z_t) := p(z_t \mid y) \propto \alpha_t \odot \beta_t \tag{3.38}$$

$$\gamma_{t:t-1}(z_{t-1}, z_t) := p(z_{t-1}, z_t \mid y) \propto \operatorname{diag}(\alpha_{t-1}) \cdot \Pi \cdot P_t \cdot \operatorname{diag}(\beta_t) \ . \tag{3.39}$$

**Gradient Estimator** As stated in Sec. 3.4.1, we use the 'expanded mean' parameters of $\Pi$ instead of $\Pi$ as in (Patterson and Teh, 2013) and the Cholesky decomposition of $\Sigma_k^{-1}$ instead of $\Sigma_k$ to ensure positive definiteness. The expanded mean parametrization is $\phi \in \mathbb{R}_+^{K\times K}$ where $\Pi_{k,\cdot} = \phi_{k,\cdot} / \sum_{k'} \phi_{k,k'}$. The Cholesky decomposition of the precision $\Sigma_k^{-1}$ is $\psi_{\Sigma_k}$ such that $\psi_{\Sigma_k}\psi_{\Sigma_k}^T = \Sigma_k^{-1}$.

The gradient of the marginal loglikelihood takes the form

$$\nabla_{\phi_k} \log p(y \,|\, \theta) = \sum_{t\in\mathcal{T}} \mathbb{E}_{\, z_t, z_{t-1}|y}[\mathbb{I}(z_{t-1} = k) \cdot \phi_k^{-1} \odot (\vec{e}_{z_t} - \Pi_k)] \tag{3.40}$$

$$\nabla_{\mu} \log p(y|\theta) = \sum_{t=1}^{T} \mathbb{E}_{\, z_t|y} \left[\Sigma_{z_t}^{-1}(y_t - \mu_{z_t})\right] \tag{3.41}$$

$$\nabla_{\psi_\Sigma} \log p(y \,|\, \theta) = \sum_{t\in\mathcal{T}} \mathbb{E}_{\, z_t|y} \left[\left(\Sigma_{z_t} - (y_t - \mu_{z_t})(y_t - \mu_{z_t})^T\right) \psi_{\Sigma_{z_t}}\right] \quad . \tag{3.42}$$

As $z$ is discrete and these expectations only involve pairwise elements of $z$, they can be tractably computed as weighted average using $\gamma(z_t, z_{t-1})$ from forward backward.

**Preconditioning** For the Gaussian HMM, the complete-data Fisher information matrix is block diagonal. With some algebra, the Fisher information matrix, precondition matrices, and correction term are

$$\mathcal{I}_{\phi_k} = (\mathrm{diag}(\Pi_k) - 11^T) \cdot (1^T\phi_k)^{-2} \;\Rightarrow\; D(\theta)_{\phi_k} = \mathrm{diag}(\phi_k) \;\text{ and }\; \Gamma(\theta)_{\phi_k} = 1 \tag{3.43}$$

$$\mathcal{I}_{\mu_k} = \Sigma_k^{-1} \;\Rightarrow\; D(\theta)_{\mu_k} = \Sigma_k \;\text{ and }\; \Gamma(\theta)_{\mu_k} = 0 \tag{3.44}$$

$$\mathcal{I}_{\psi_{\Sigma_k}} = 2(I_m \otimes \Sigma_k) \;\Rightarrow\; D(\theta)_{\psi_{\Sigma_k}} = \frac{1}{2}(I_m \otimes \Sigma_k^{-1}) \;\text{ and }\; \Gamma(\theta)_{\psi_{\Sigma_k}} = \psi_{\Sigma_k} \tag{3.45}$$

For $\phi_k$, we use $D(\theta)_{\phi_k} = \mathrm{diag}(\phi_k)$ and $\Gamma(\theta)_{\phi_k} = 1$, following past work (Patterson and Teh, 2013; Ma et al., 2017). However, we observed that $\phi_k$ will be absorbed at 0, whenever $\phi_k$ falls sufficiently close to 0. To fix this we recommend adding a small identity matrix $\nu_\phi I_K$ (for some $\nu_\phi > 0$) to $D(\theta)_\phi$. An alternative solution is to use a stochastic Cox-Ingersoll-Ross process to sample *sparse* $\pi$ instead (Baker et al., 2018).

### *3.4.2 Autoregressive HMM*

We now consider ARHMMs, a generalization of the discrete state HMM where each observation depends not only on the latent state, but also on the last $p$ observations. Specifically, the discrete latent state $z_t$ determines which AR($p$) process models the dynamics of $y$ at time $t$. The generative process for the ARHMM is

$$\begin{aligned} z_t \,|\, z_{t-1}, \theta &\sim \text{Categorical}(z_t \,|\, \Pi_{z_{t-1}}) \\ y_t \,|\, z_t, y_{<t}, \theta &\sim \mathcal{N}\left(y_t \,|\, A_{z_t} \overline{y_t}\,,\, Q_{z_t}\right), \end{aligned} \tag{3.46}$$

where $y_t \in \mathbb{R}^m$ are the observations, $\overline{y_t} = y_{t-1:t-p}$ are the $p$-lagged observations, $u_t \equiv z_t \in \{1, \ldots, K\}$ are the discrete latent variables, and $\theta = \{\Pi, A, Q\}$ are the parameters with $\Pi_k \in \Delta^K$, $A_k \in \mathbb{R}^{m \times mp}$, $Q_k \in \mathbb{S}_+^m$ for $k = 1, \ldots, K$.

The complete data likelihood of an ARHMM is as follows

$$p(y, z \,|\, \theta) = \prod_{t=1}^{T} \Pi_{z_{t-1}, z_t} \cdot \mathcal{N}(y_t \,|\, A_{z_t} \overline{y_t}, Q_{z_k}) \ , \tag{3.47}$$

From Eq. (3.47), we see that the ARHMM is a time-dependent mixture of $K$ AR processes of order $p$. The pairwise latent marginals, gradients, and preconditioning terms for an ARHMM are calculated similarly to the Gaussian HMM.

**Forward Backward** The forward backward recursions for the ARHMM are identical to the Gaussian HMM Eqs. (3.35)-(3.39), where $P_t$ is now

$$P_t := \text{diag}\{\mathcal{N}(y_t \,|\, A_k \overline{y_t}, Q_k)\}_{k=1}^K \ . \tag{3.48}$$

**Gradient Estimator** The gradient of the marginal loglikelihood is similar to the Gaussian HMM Eqs. (3.40)-(3.42) with $\mu_k$ replaced with $A_k \overline{y}_t$

$$\nabla_{\phi_k} \log p(y \,|\, \theta) = \sum_{t \in \mathcal{T}} \mathbb{E}_{\, z_t, z_{t-1}|y} [\mathbb{I}(z_{t-1} = k) \cdot \phi_k^{-1} \odot (\vec{e}_{z_t} - \Pi_k)] \tag{3.49}$$

$$\nabla_A \log p(y|\theta) = \sum_{t=1}^{T} \mathbb{E}_{\, z_t|y} \left[ Q_{z_t}^{-1} (y_t - A_{z_t} \overline{y}_t) \overline{y}_t^T \right] \tag{3.50}$$

$$\nabla_{\psi_Q} \log p(y \,|\, \theta) = \sum_{t \in \mathcal{T}} \mathbb{E}_{\, z_t|y} \left[ \left( Q_{z_t} - (y_t - A_{z_t} \overline{y}_t)(y_t - A_{z_t} \overline{y}_t)^T \right) \psi_{Q_{z_t}} \right] \ . \tag{3.51}$$

**Preconditioning** The preconditioning terms for the ARHMM is similar to the Gaussian HMM

$$\mathcal{I}_{\phi_k} = (\mathrm{diag}(\Pi_k) - 11^T) \cdot (1^T \phi_k)^{-2} \ \Rightarrow \ D(\theta)_{\phi_k} = \mathrm{diag}(\phi_k) \ \text{ and } \ \Gamma(\theta)_{\phi_k} = 1 \tag{3.52}$$

$$\mathcal{I}_{A_k} = \mathbb{E}_{\, y,z|\theta} [\overline{y}_t \overline{y}_t^T] \otimes Q_k^{-1} \ \Rightarrow \ D(\theta)_{A_k} = I_m \otimes Q_k \ \text{ and } \ \Gamma(\theta)_{A_k} = 0 \tag{3.53}$$

$$\mathcal{I}_{\psi_{Q_k}} = 2(I_m \otimes Q_k) \ \Rightarrow \ D(\theta)_{\psi_{Q_k}} = \frac{1}{2}(I_m \otimes Q_k^{-1}) \ \text{ and } \ \Gamma(\theta)_{\psi_{Q_k}} = \psi_{Q_k} \tag{3.54}$$

The expectation $\mathbb{E}\,[\overline{y}_t \overline{y}_t^T]$ does not have a closed form as the expectation is over $z$ is a combinatorial sum. Therefore, we choose to replace $\mathbb{E}\,[\overline{y}_t \overline{y}_t^T]$ with the empirical estimate $I_m$ (assuming $y_t$ has been preprocessed by whitening to unit covariance) in our preconditioning matrix $D(\theta)_A$.

### *3.4.3 Linear Gaussian SSM*

A linear Gaussian SSM (LGSSM), also called a linear dynamical system (LDS), consists of a latent Gaussian (vector) autoregressive process over states $u_t \equiv x_t \in \mathbb{R}^n$ and conditionally Gaussian emissions $y_t \in \mathbb{R}^m$ (Bishop, 2006; Lütkepohl, 2005). See Chapter 2.2.1 for the generative process of a LGSSM. The complete data likelihood of a LGSSM is

$$p(y, x \,|\, \theta) = \prod_{t=1}^{T} \mathcal{N}(x_t \,|\, A x_{t-1}, Q) \cdot \mathcal{N}(y_t \,|\, C x_t, R) \ , \tag{3.55}$$

where $A \in \mathbb{R}^{n \times n}$ is the latent state transition matrix, $Q \in \mathbb{S}_+^n$ is the transition noise covariance, $C \in \mathbb{R}^{m \times n}$ is the emission matrix, and $R \in \mathbb{S}_+^m$ is the emission noise covariance.

Together $A, Q, C, R$ are the model parameters $\theta$. The matrices $A$, $C$, and $Q$ are unidentifiable without additional restriction, as applying an orthonormal transformation $M$ gives an equivalent representation $\tilde{A} = MAM^{-1}$, $\tilde{C} = CM^{-1}$, $\tilde{Q} = MQM^T$. To enforce identifiability, we choose to restrict the first $\min(n, m)$ rows and columns of $C$ to the identity matrix. In practice, we use the Cholesky decompositions $\psi_Q, \psi_R$ of $Q^{-1}, R^{-1}$ (respectively) instead of $Q, R$.

**Forward Backward** The recursions for the forward backward algorithm for LGSSMs is known as the Kalman smoother (Cappé et al., 2005; Bishop, 2006; Fox, 2009). Because the transition and emission processes are linear Gaussian, all forward messages, backward messages, and pairwise latent marginals $\gamma_{t-1:t}(x_t, x_{t-1})$ are Gaussian.

$$\alpha_t := p(x_t, y_{\leq t}) = \mathcal{N}(x_t \,|\, \mu_{\alpha_t} = \Lambda_{\alpha_t}^{-1} h_{\alpha_t}, \Sigma_{\alpha_t} = \Lambda_{\alpha_t}^{-1}) \tag{3.56}$$

$$\beta_t := p(y_{>t} \,|\, x_t) \propto \mathcal{N}(x_t \,|\, \mu_{\beta_t} = \Lambda_{\beta_t}^{-1} h_{\beta_t}, \Sigma_{\beta_t} = \Lambda_{\beta_t}^{-1}) \ , \tag{3.57}$$

where $h_{\alpha_t}, \Lambda_{\alpha_t}$ are the Gaussian natural parameters of $\alpha$ that satisfy the recursion

$$\Lambda_{\alpha_t} = C^T R^{-1} C + (Q + A\Lambda_{\alpha_{t-1}}^{-1} A^T)^{-1} \tag{3.58}$$

$$h_{\alpha_t} = C^T R^{-1} y_t + (Q + A\Lambda_{\alpha_{t-1}}^{-1} A^T)^{-1} A\Lambda_{\alpha_{t-1}}^{-1} h_{\alpha_{t-1}} \ , \tag{3.59}$$

and $h_{\beta_t}, \Lambda_{\beta_t}$ are the Gaussian natural parameters of $\beta$ that satisfy the recursion

$$\Lambda_{\beta_t} = A^T Q^{-1} A - A^T Q^{-1} (Q^1 + C^T R^{-1} C + \Lambda_{\beta_{t+1}})^{-1} Q^{-1} A \tag{3.60}$$

$$h_{\beta_t} = A^T Q^{-1} (Q^{-1} + C^T R^{-1} C + \Lambda_{\beta_{t+1}})^{-1} (C^T R^{-1} y_{t+1} + h_{\beta_{t+1}}) \ . \tag{3.61}$$

Given the messages $\alpha_t, \beta_t$ the marginal and pairwise posteriors of the latent states $x_t$ and $(x_{t-1}, x_t)$ are computed as

$$\begin{aligned} \gamma_t(x_t) := p(x_t \,|\, y) &\propto \alpha_t(x_t)\beta_t(x_t) \\ &\propto \mathcal{N}(x_t \,|\, \mu = \Sigma(h_{\alpha_t} + h_{\beta_t}), \Sigma = (\Lambda_{\alpha_t} + \Lambda_{\beta_t})^{-1}) \end{aligned} \tag{3.62}$$

$$\gamma_{t-1,t}(x_{t-1}, x_t) := p(x_{t-1}, x_t \,|\, y) \propto \alpha_{t-1}(x_{t-1} p(y_t, x_t \,|\, x_{t-1}) \beta_t(x_t) \tag{3.63}$$
$$\propto \mathcal{N}\Big( \begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix} \Big| \mu = \Sigma \cdot \begin{bmatrix} h_{\alpha_{t-1}} \\ C^T R_{-1} y_t + h_{\beta_t} \end{bmatrix},$$
$$\Sigma = \begin{bmatrix} \Lambda_{\alpha_{t-1}} + A^T Q^{-1} A & A^T Q^{-1} \\ Q^{-1} A & C^T R^{-1} C + Q^{-1} + \Lambda_{\beta_t} \end{bmatrix}^{-1} \Big) \ .$$

**Gradient Estimator** We compute the gradient of marginal loglikelihood via Fisher's identity

$$\nabla_A \log p(y|\theta) = \sum_{t=1}^{T} \mathbb{E}_{x|y} \left[ Q^{-1}(x_t - A x_{t-1}) x_{t-1}^T \right] \tag{3.64}$$

$$\nabla_{\psi_Q} \log p(y|\theta) = \sum_{t=1}^{T} \mathbb{E}_{x|y} \left[ (Q - (x_t - A x_{t-1})(x_t - A x_{t-1})^T) \psi_Q \right] \tag{3.65}$$

$$\nabla_C \log p(y|\theta) = \sum_{t=1}^{T} \mathbb{E}_{x|y} \left[ R^{-1}(y_t - C x_t) x_t^T \right] \tag{3.66}$$

$$\nabla_{\psi_R} \log p(y|\theta) = \sum_{t=1}^{T} \mathbb{E}_{x|y} \left[ (R - (y_t - C x_t)(y_t - C x_t)^T) \psi_R \right] \tag{3.67}$$

Because each gradient is linear with respect to first and second order terms (e.g. $x_t$, $x_t x_t^T$ and $x_t x_{t-1}^T$), their expectation of each of these terms is easily computable given $\gamma(x_t, x_{t-1})$.

Let $\gamma_{t,t-1}(x_t, x_{t-1})$ be the Gaussian pairwise marginal posterior from forward backward (see Eq. (3.63))

$$\gamma_{t-1,t}(x_{t-1}, x_t) = \mathcal{N}\left( \begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix} \Big| \mu = \begin{bmatrix} \mu_{t-1} \\ \mu_t \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{t-1,t-1} & \Sigma_{t-1,t} \\ \Sigma_{t,t-1} & \Sigma_{t,t} \end{bmatrix} \right) \ . \tag{3.68}$$

Let $M = \Sigma + \mu\mu^T$ be the second moment of $\gamma_{t-1,t}$, that is $M_{t,t'} := \mathbb{E}\left[x_t, x_{t'}^T\right]$.

Then the expectations in the summations of Eqs. (3.64)-(3.67) are

$$\mathbb{E}_{x|y}\left[(x_t - Ax_{t-1})x_{t-1}^T\right] = M_{t,t-1} - AM_{t-1,t-1} \tag{3.69}$$

$$\mathbb{E}_{x|y}\left[(x_t - Ax_{t-1})(x_t - Ax_{t-1})^T\right] = M_{t,t} - AM_{t-1,t} - M_{t,t-1}A^T + AM_{t-1,t-1}A^T \tag{3.70}$$

$$\mathbb{E}_{x|y}\left[(y_t - Cx_t)x_t^T\right] = y_t\mu_t^T - CM_{t,t} \tag{3.71}$$

$$\mathbb{E}_{x|y}\left[(y_t - Cx_t)(y_t - Cx_t)^T\right] = y_t y_t^T - C\mu_t y_t^T - y_t\mu_t^T C^T + CM_{t,t}C^T \ . \tag{3.72}$$

**Preconditioning** For the LGSSM, the complete data Fisher information matrix is block diagonal. With some algebra, the Fisher information matrix, precondition matrices, and correction term are

$$\mathcal{I}_A = \mathbb{E}\left[x_t x_t\right]^T \otimes Q^{-1} \ \Rightarrow \ D_A = I_n \otimes Q \ \text{ and } \ \Gamma(\theta)_A = 0 \tag{3.73}$$

$$\mathcal{I}_{\psi_Q} = 2(I_n \otimes Q) \ \Rightarrow \ D_{\psi_Q} = \frac{1}{2}(I_n \otimes Q^{-1}) \ \text{ and } \ \Gamma(\theta)_{\psi_Q} = \psi_Q \tag{3.74}$$

$$\mathcal{I}_C = \mathbb{E}\left[x_t x_t\right]^T \otimes R^{-1} \ \Rightarrow \ D_C = I_n \otimes R \ \text{ and } \ \Gamma(\theta)_C = 0 \tag{3.75}$$

$$\mathcal{I}_{\psi_R} = 2(I_m \otimes R) \ \Rightarrow \ D_{\psi_R} = \frac{1}{2}(I_m \otimes R^{-1}) \ \text{ and } \ \Gamma(\theta)_{\psi_R} = \psi_R \ , \tag{3.76}$$

where $\mathbb{E}\left[x_t x_t\right]^T = \sum_{s=0}^{\infty} A^s Q (A^s)^T$ for the LGSSM. In our experiments we chose to replace $\mathbb{E}\left[x_t x_t\right]^T$ with the identity matrix $I_n$ to match the ARHMM setup.

**Error Bound Coefficients** The random maps of an LGSSM are strict contractions under mild conditions (Lemmas 1, 2) and the gradients are Lipschitz in $xx^T$ (Lemma 3). Therefore, Theorem 4 applies.

**Lemma 1.** *The forward random maps of an LGSSM are Gaussian linear maps. Specifically,* $\vec{\psi}_t(x_t) = F_t^f x_t + \zeta_t^f$*, where* $\zeta_t^f$ *is a Gaussian random intercept and* $F_t^f$ *is a matrix function of* $\theta$ *and* $y_{>t}$*. As a linear map, the Lipschitz constant of* $\vec{\psi}_t$ *is*

$$\|\vec{\psi}_t\|_{Lip} = \|F_t^f\|_2 \leq \|A(I_n + QC^T R^{-1} C)^{-1}\|_2 \ . \tag{3.77}$$

*As* $\|(I_n + QC^T R^{-1} C)^{-1}\|_2 < 1$*, if* $\|A\|_2 < 1$*, then* $\|\vec{\psi}_t\|_{Lip} < 1$ *for all* $t$*.*

**Lemma 2.** *The backward random maps of an LGSSM are Gaussian linear maps. Specifically,* $\overleftarrow{\psi}_t(x_t) = F_t^b x_t + \zeta_t^b$*, where* $\zeta_t^b$ *is a Gaussian random intercept and* $F_t^b$ *is a matrix function of* $\theta$ *and* $y_{<t}$*. As a linear map, the Lipschitz constant of* $\overleftarrow{\psi}_t$ *is*

$$\|\overleftarrow{\psi}_t\|_{Lip} = \|F_t^b\|_2 \leq \|A(QA^TQ^{-1}A + QC^TR^{-1}C)^{-1}\|_2 \ . \tag{3.78}$$

*If* $\|A\|_2 < \|(QA^TQ^{-1}A + QC^TR^{-1}C)^{-1}\|_2$*, then* $\|\vec{\psi}_t\|_{Lip} < 1$ *for all* $t$*. In addition, when the variance of the prior* $p_0(x)$ *is less than the steady state variance* $V_\infty = Q + AV_\infty A^T$ *and* $A$ *commutes with* $Q$*, we obtain a tighter bound*

$$\|\overleftarrow{\psi}_t\|_{Lip} = \|F_t^b\|_2 \leq \|A(I_n + QC^TR^{-1}C)^{-1}\|_2 \ . \tag{3.79}$$

*In this case, if* $\|A\|_2 < 1$*, then* $\|\overleftarrow{\psi}_t\|_{Lip} < 1$ *for all* $t$*.*

Lemmas 1 and 2 agree with intuition, when $\|A\|_2 \approx 0$ (no connection between $x_{t-1}$ and $x_t$) or $\|Q\|_2 \gg \|R\|_2$ (transition noise is much larger than emission noise), then $L_\theta \approx 0$ (observations can be treated independently). Conversely, when $\|A\|_2 \approx 1$ and $\|Q\|_2 \ll \|R\|_2$, then $L_\theta \approx 1$ and buffering is necessary. The proofs of the Lemmas take advantage of Theorem 5 and are in Appendix A.1.5.

The following lemma bounds the Lipschitz constant of the gradient.

**Lemma 3.** *As* $x, y$ *are jointly Gaussian in the LGSSM, the gradient of the complete data loglikelihood is a quadratic form in* $xx^T$ *with matrices*

$$\begin{aligned}\Omega = \{&I_n \otimes Q^{-1}, I_n \otimes Q^{-1}A, Q^{-1/2} \otimes I_n, Q^{-1/2}A \otimes I_n, Q^{-1/2} \otimes A, Q^{-1/2}A \otimes A, \\ &I_n \otimes R^{-1}, I_n \otimes R^{-1}C, R^{-1/2}C \otimes C\} \ ,\end{aligned} \tag{3.80}$$

*where* $Q^{-1/2} = \psi_Q$ *and* $R^{-1/2} = \psi_R$*. Therefore a bound for the Lipschitz constant is* $L'_U = \max_{\omega \in \Omega} \|\omega\|_2$*. This bound grows in* $\|A\|, \|C\|, \|Q\|^{-1}, \|R\|^{-1}$*.*

**Particle Smoothing** Although the LGSSM has closed form message passing, in our experiments we also consider using the particle buffered stochastic gradient $g^{\mathrm{PF}}_{S,B,N}$ during

SGMCMC to assess the impact of the particle smoother on buffered subsequences. When applying the particle smoother, Algorithm 2, to the LGSSM, there are two common proposal densities $q(\cdot|\cdot)$:

- The prior (transition) kernel

$$q(x_t \,|\, x_{t-1}) = \mathcal{N}(x_t \,|\, \phi x_{t-1}, \sigma^2), \tag{3.81}$$

  where the weight update, Eq. (2.28), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(\frac{-(y_t - x_t^{(i)})^2}{2\tau^2}\right). \tag{3.82}$$

- The 'optimal instrumental kernel'

$$q(x_t \,|\, x_{t-1}, y_t) = \mathcal{N}\left(x_t \,\middle|\, \frac{\tau^2\phi x_{t-1} + \sigma^2 y_t}{\sigma^2 + \tau^2}, \frac{\sigma^2\tau^2}{\sigma^2 + \tau^2}\right), \tag{3.83}$$

  where the weight update, Eq. (2.28), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi(\sigma^2 + \tau^2)}} \exp\left(\frac{-(y_t - \phi x_{t-1}^{(a_i)})^2}{2(\sigma^2 + \tau^2)}\right). \tag{3.84}$$

In our experiments with the LGSSM, we use the optimal instrumental kernel.

### *3.4.4 Stochastic Volatility Model*

The *stochastic volatility model* (SVM) (Shephard, 2005) is

$$\begin{aligned} x_t \,|\, x_{t-1}, \theta &\sim \mathcal{N}(x_t \,|\, \phi x_{t-1}\,,\, \sigma^2), \\ y_t \,|\, x_t, \theta &\sim \mathcal{N}(y_t \,|\, 0\,,\, \exp(x_t)\tau^2), \end{aligned} \tag{3.85}$$

with parameters $\theta = (\phi, \sigma, \tau)$.

The elementwise complete data loglikelihood is

$$\log p(y_t, x_t \,|\, x_{t-1}, \theta) = -\log(2\pi) - \log(\sigma) - \log(\tau) - \frac{(x_t - \phi x_{t-1})^2}{2\sigma^2} - 0.5x_t - \frac{(y_t)^2}{2\exp(x_t)\tau^2}. \tag{3.86}$$

**Particle Smoothing** For the particle filter, we use the prior kernel as the proposal density

$$q(x_t \,|\, x_{t-1}) = \mathcal{N}(x_t \,|\, \phi x_{t-1}, \sigma^2), \tag{3.87}$$

with weight update

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(\frac{-y_t^2}{2\exp(x_t^{(i)})\tau^2}\right). \tag{3.88}$$

**Gradient Estimator** The gradient of the complete data loglikelihood is then,

$$\begin{aligned}
\nabla_\phi \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(x_t - \phi x_{t-1}) \cdot x_{t-1}}{\sigma^2}, \\
\nabla_\sigma \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(x_t - \phi x_{t-1})^2 - \sigma^2}{\sigma^3}, \\
\nabla_\tau \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{y_t^2/\exp(x_t) - \tau^2}{\tau^3}.
\end{aligned} \tag{3.89}$$

We parametrize with $\sigma^{-1}$ and $\tau^{-1}$ to obtain,

$$\begin{aligned}
\nabla_{\sigma^{-1}} \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{\sigma^2 - (x_t - \phi x_{t-1})^2}{\sigma}, \\
\nabla_{\tau^{-1}} \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{\tau^2 - y_t^2/\exp(x_t)}{\tau}.
\end{aligned} \tag{3.90}$$

**Error Bound Coefficients** For the SVM, the transition and emission distributions are log-concave in $x$, allowing Theorem 5 to apply. In the Appendix, we show that the prior kernels $\{\vec{\Psi}_t^{(0)}, \overleftarrow{\Psi}_t^{(0)}\}$ of the SVM are bounded with the Lipschitz constant $L_\theta = |\phi|$. Thus, the buffering error decays geometrically with increasing buffer size $B$ when $|\phi| < 1$.

### *3.4.5 GARCH Model*

The *generalized autoregressive conditional heteroskedasticity* (GARCH) model is a classic model of financial time series (Bollerslev, 1986). Here, we consider a GARCH(1,1) model (with noise)

$$\begin{aligned}
\sigma_t^2(x_{t-1}, \sigma_{t-1}^2, \theta) &= \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2, \\
x_t \,|\, x_{t-1}, \sigma_t^2, \theta &\sim \mathcal{N}(x_t \,|\, 0,\, \sigma_t^2), \\
y_t \,|\, x_t, \theta &\sim \mathcal{N}(y_t \,|\, x_t\,,\, \tau^2),
\end{aligned} \tag{3.91}$$

with parameters $\theta = (\alpha, \beta, \gamma, \tau)$. Unlike the LGSSM and SVM, the noise between $x_t$ and $x_{t-1}$ is multiplicative in $x_{t-1}$ rather than additive.

The elementwise complete data loglikelihood is

$$\begin{aligned}\log p(y_t, x_t, \sigma_t^2 \,|\, x_{t-1}, \sigma_{t-1}^2, \theta) = &- \log(2\pi) - \frac{\log(\alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2)}{2} \\ &- \frac{x_t^2}{2(\alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2)} - \log(\tau) - \frac{(y_t - x_t)^2}{2\tau^2}. \end{aligned} \tag{3.92}$$

**Particle Smoothing** We consider two proposal densities $q(\cdot|\cdot)$ for the GARCH model:

- The prior kernel

$$q\left(\begin{bmatrix} x_t \\ \sigma_t^2 \end{bmatrix} \;\middle|\; \begin{bmatrix} x_{t-1} \\ \sigma_{t-1}^2 \end{bmatrix}\right) = \begin{bmatrix} \mathcal{N}(x_t \,|\, 0, \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2) \\ \delta(\sigma_t^2 \,|\, \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2) \end{bmatrix} . \tag{3.93}$$

  where the weight update, Eq. (2.28), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(\frac{-(y_t - x_t^{(i)})^2}{2\tau^2}\right). \tag{3.94}$$

- The optimal instrumental kernel

$$q\left(\begin{bmatrix} X_t \\ \sigma_t^2 \end{bmatrix} \;\middle|\; \begin{bmatrix} x_{t-1} \\ \sigma_{t-1}^2 \end{bmatrix}, y_t\right) = \begin{bmatrix} \mathcal{N}(x_t \,|\, \sigma_t^2 y_t/(\sigma_t^2 + \tau^2), \sigma_t^2 \tau^2/(\sigma_t^2 + \tau^2)) \\ \delta(\sigma_t^2 \,|\, \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2) \end{bmatrix} . \tag{3.95}$$

  where the weight update, Eq. (2.28), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi((\sigma_t^{(i)})^2 + \tau^2)}} \exp\left(\frac{-y_t^2}{2((\sigma_t^{(i)})^2 + \tau^2)}\right). \tag{3.96}$$

In our experiments with the GARCH model, we use the optimal instrumental kernel.

**Gradient Estimator** Let $\mathcal{L}_t = \log p(y_t, x_t, \sigma_t^2 | x_{t-1}, \sigma_{t-1}^2, \theta)$ and set $C_t = \frac{x_t^2 - \sigma_t^2}{2\sigma_t^4}$. Then the gradient of the complete data log-likelihood $\nabla \mathcal{L}_t$ is

$$\begin{aligned}
\nabla_\tau \mathcal{L}_t &= \frac{(y_t - x_t)^2 - \tau^2}{\tau^3}, \\
\nabla_{\log \mu} \mathcal{L}_t &= C_t \cdot (1 - \phi) \cdot \mu, \\
\nabla_{\operatorname{logit} \phi} \mathcal{L}_t &= C_t \cdot (\lambda x_{t-1}^2 + (1 - \lambda)\sigma_{t-1}^2 - \mu) \cdot \phi(1 - \phi), \\
\nabla_{\operatorname{logit} \lambda} \mathcal{L}_t &= C_t \cdot (\phi x_{t-1}^2 - \phi \sigma_{t-1}^2) \cdot \lambda(1 - \lambda).
\end{aligned} \tag{3.97}$$

where parameters are $\theta = (\log \mu, \operatorname{logit} \phi, \operatorname{logit} \lambda, \tau)$ for $\alpha = \mu(1 - \phi)$, $\beta = \phi\lambda$, $\gamma = \phi(1 - \lambda)$. Note that $\sigma_t^2 = \mu(1 - \phi) + \phi(\lambda x_{t-1}^2 + (1 - \lambda)\sigma_{t-1}^2)$.

**Error Bound Coefficients** This model is *not* log-concave and therefore our theory (Theorem 5) does not hold. However, we see empirically in the experiments (Chapter 3.5.2) that buffering can help reduce the error of $\hat{g}_{S,B}$ for the GARCH model.

### *3.4.6 Switching Linear Dynamical System*

Switching linear dynamical systems (SLDSs) are an example of a state space model with both discrete and continuous latent state sequences. The form of SLDS models that we consider is

$$\begin{aligned}
z_t \,|\, z_{t-1}, \theta &\sim \text{Categorical}(z_t \,|\, \Pi_{z_{t-1}}) \\
x_t \,|\, x_{t-1}, z_t, \theta &\sim \mathcal{N}(x_t \,|\, A_{z_t} x_{t-1}, Q_{z_t}) \\
y_t \,|\, x_t, \theta &\sim \mathcal{N}\left(y_t \,|\, C x_t\,,\, R\right),
\end{aligned} \tag{3.98}$$

where $y_t \in \mathbb{R}^m$ are the observations, $u_t \equiv (x_t, z_t) \in \mathbb{R}^n \times \{1, \ldots, K\}$ are the mixed-type latent state sequence, and $\theta = \{\Pi, A, Q, C, R\}$ the model parameters with $\Pi_k \in \Delta^K$, $A_k \in \mathbb{R}^{n \times n}$, $Q_k \in \mathbb{S}_+^n$ for $k = 1, \ldots, K$, $C \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{S}_+^m$. The complete data likelihood of SLDS models is

$$p(y, x, z \,|\, \theta) = \prod_{t=1}^{T} \Pi_{z_{t-1}, z_t} \cdot \mathcal{N}(x_t \,|\, A_{z_t} x_{t-1}, Q_{z_t}) \cdot \mathcal{N}(y_t \,|\, C x_t, R)\ , \tag{3.99}$$

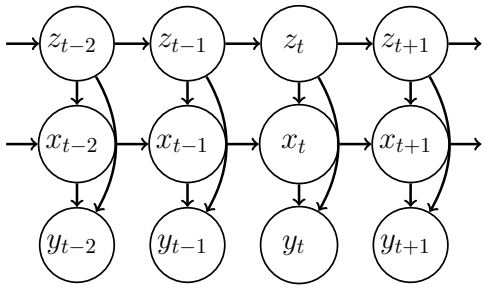


Figure 3.2: Graphical Model of a SLDS.

The SLDS of Eq. (3.99) can be viewed either as a latent AR(1)-HMM with conditional Gaussian emissions or as hidden Markov switches of a LGSSM. As an extension of the ARHMM, the latent continuous state sequence $x_t$ can *smooth* noisy observations. As an extension of the LGSSM, the latent discrete state sequence $z_t$ allows modeling of more complex dynamics by *switching* between different states (or regimes).

*Blocked Gibbs*

Unlike previous models, the forward-backward algorithm for the latent variables $(x, z)$ in an SLDS does not have a closed form. Specifically, the transition kernel for $x$ is a Gaussian mixture, so the forward and backward messages of $x$ are Gaussian mixtures with an exponentially increasing number of components (e.g. $\alpha_t$ has $K^t$ components). Because the forward-backward algorithm is intractable for SLDSs, we rely on sampling $(x, z)$ and forming a Monte Carlo estimate of the expectation in Fisher's identity Eq. (3.5). We consider various options of this Monte Carlo estimate below. To sample $(x, z)$, we use a blocked Gibbs scheme as in (Fox et al., 2011), detailed in the Appendix.

Given a collection of $N$ samples from blocked Gibbs $\{x^{(r)}, z^{(r)}\} \sim x, z \,|\, y_{\mathcal{S}^*}, \theta$, we construct three different estimators for the marginal loglikelihood. The first estimator, replaces the expectation in Eq. (3.5) with a Monte Carlo average

$$\mathbb{E}_{x,z|y,\theta}[\nabla \log p(y, x, z \,|\, \theta)] \approx \frac{1}{N} \sum_{r=1}^{N} \nabla \log p(y, x^{(r)}, z^{(r)} \,|\, \theta) \ . \tag{3.100}$$

We construct two additional estimators by analytically integrating out either one of the two latent variables. These estimators are the *Rao-Blackwellization* of the naive Monte Carlo estimate (Cappé et al., 2005). Integrating out either $x$ or $z$, gives us

$$\mathbb{E}_{x,z|y,\theta}[\nabla \log p(y,x,z \,|\, \theta)] = \frac{1}{N}\sum_{r=1}^{N} \mathbb{E}_{x|y,z^{(r)},\theta}[\nabla \log p(y,x,z^{(r)} \,|\, \theta)] \tag{3.101}$$

$$\mathbb{E}_{x,z|y,\theta}[\nabla \log p(y,x,z \,|\, \theta)] = \frac{1}{N}\sum_{r=1}^{N} \mathbb{E}_{z|y,x^{(r)},\theta}[\nabla \log p(y,x^{(r)},z \,|\, \theta)] \ . \tag{3.102}$$

Because Eq. (3.101) integrates out $x$, it has lower variance for the gradient terms involving $x$ (i.e. $A$, $Q$ $R$). Similarly, because Eq. (3.102) integrates out $z$, it has lower variance for the gradient terms involving $z$ (i.e. $\Pi$).

Selecting one of the above Monte Carlo estimates of $\nabla U(\theta)$, we can deploy the same buffered subsampling estimator Eq. (3.5), obtaining Algorithm 5. Algorithm 5 replaces the forward-backward subroutine in Algorithm 3 with blocked Gibbs sampling over $\mathcal{S}^*$. Although this is more computationally costly than the exact forward-backward algorithms of the previous sections, it still provides memory saving and runtime speed ups compared to running a full blocked Gibbs sampler over $1:T$. The explicit forms of Eqs. (3.100)-(3.102), precondition matrix $D(\theta)$, and correction term $\Gamma(\theta)$ for SLDS used in Alg. 1 are a combination of those for ARHMMs and LGSSMs.

**Blocked Gibbs Conditional Distributions** As the SLDS does not have a closed form forward-backward algorithm, we instead present the details for the blocked Gibbs sampling scheme (conditional distributions and Initialization) used in Algorithm 5.

The conditional posterior distribution of $x$ given $y$ and $z$ follows a time-varying LGSSM. To sample $x$, we can use the time-varying Kalman filter (Hamilton, 1994). We first calculate the forward messages $\alpha_t(x_t)$ using the Kalman filter recursion Eq. (3.56) with $A_t = A_{z_t}$, $C_t = C$, $Q_t = Q_{z_t}$, and $R_t = R$. Given $\alpha_t(x_t) \propto \mathcal{N}(x_t \,|\, \mu_{\alpha_t}, \Sigma_{\alpha_t})$, we sample $x$ using the

**Algorithm 5** `NoisyGradient` using blocked Gibbs (SLDS)

**input:** data $y$, parameters $\theta$, subsequence length $S$, error tolerance $\epsilon$,

$B = \texttt{BufferLength}(\theta, S, \epsilon)$ // From Theory or Adaptive

$\mathcal{S}, \mathcal{S}^* = \texttt{GetBufferedSubsequence}(y, S, B)$

$z_{\mathcal{S}^*}^{(0)} = \texttt{InitLatent}(\mathcal{S}^*, \theta)$ // With 'burn-in'

**for** $r = 1, 2, \ldots, N$ **do**

sample $x_{\mathcal{S}^*}^{(r)} \sim x_{\mathcal{S}^*} \,|\, y_{\mathcal{S}^*}, z_{\mathcal{S}^*}^{(r-1)}$ // Blocked Gibbs

sample $z_{\mathcal{S}^*}^{(r)} \sim z_{\mathcal{S}^*} \,|\, y_{\mathcal{S}^*}, x_{\mathcal{S}^*}^{(r)}$

**end for**

calculate $\widetilde{U}(\theta)$ using a Monte Carlo estimate // Eq. (3.100), (3.101), or (3.102)

**return** $\nabla \widetilde{U}(\theta)$

backward sampler (starting from $t = T$ and descending)

$$x_t \,|\, x_{t-1} \sim \begin{cases} \mathcal{N}\left(x_T \,|\, \mu = \mu_{\alpha_T}\,,\, \Sigma = \Sigma_{\alpha_T}\right) \text{ if } t = T \;\;, \text{ otherwise} \\ \mathcal{N}\left(x_t \,\middle|\, \mu = \Sigma(\Sigma_{\alpha_t}^{-1}\mu_{\alpha_t} + A_{z_{t+1}}^T Q_{z_{t+1}}^{-1} x_{t+1})\,,\, \Sigma = (\Sigma_{\alpha_t}^{-1} + A_{z_{t+1}}^T Q_{z_{t+1}}^{-1} A_{z_{t+1}})^{-1}\right) \end{cases} \tag{3.103}$$

The conditional posterior distribution of $z$ given $y$ and $x$ follows the ARHMM. To sample $z$, we apply a similar sampler for the ARHMM. We first calculate the backward messages $\beta_t(z_t)$ using the ARHMM forward messages Eq. (3.36), replacing $y$ with $x$. Given $\alpha_t(z_t)$, we then sample $z$ sequentially in ascending order using the forward sampler

$$p(z_t = k \,|\, z_{t-1}, x, y) \propto p(x_t, y_t \,|\, x_{t-1}, z_t = k, \theta) \odot \Pi_{z_{t-1},k} \odot \beta_t(k) \;\; . \tag{3.104}$$

Finally, the conditional posterior distribution of $z_t$ given $y$ and $z_{\setminus t}$ can be calculated using the forward backward algorithm to marginalize $x$. Specifically,

$$p(z_t = k \,|\, z_{\setminus t}, y) \propto \Pi_{z_{t-1},k} \Pi_{k,z_t} \cdot \int \alpha_{t-1}(x_{t-1}) p(y_t, x_t \,|\, x_{t-1}, z_t = k) \beta_t(x_t)\, dx_t dx_{t-1} \;\; , \tag{3.105}$$

where $\alpha_{t-1}, \beta_t$ are calculated using Eqs. (3.56)-(3.57) with $A_{t'} = A_{z_{t'}}, Q_{t'} = Q_{z_{t'}}$ for all $t' \in \mathcal{S}^* \setminus \{t\}$.

Note that Eq (3.105) requires $\mathcal{O}(|\mathcal{S}^*|)$ time per time point $z_t$; therefore one pass over $z_{\mathcal{S}^*}$ requires $\mathcal{O}(|\mathcal{S}^*|^2)$.

**Initialization of Blocked Gibbs Sampler** To sample $z$ from the filtered process, we recursively sample from the conditional distribution $z_t \,|\, y_t, z_{t-1}$

$$p(z_t = k \,|\, z_{t-1}, y_t) \propto \Pi_{z_{t-1},k} \cdot \int \alpha_{t-1}(x_{t-1}) p(y_t, x_t \,|\, x_{t-1}, z_t = k)\, dx_t dx_{t-1} \ , \tag{3.106}$$

where $\alpha_{t-1}$ is calculated using Eq. (3.56) with $A_{t'} = A_{z_{t'}}, Q_{t'} = Q_{z_{t'}}$ for all $t' < t$. Because we do not condition on $y_{>t}$ when $z_t$ is sampled, we emphasize that this distribution is not the posterior $z \,|\, y$ (it is the *filtered* distribution, not the *smoothed* distribution). However, it provides a better initialization point than sampling $z$ from the prior.

Alternatively, when $\dim(x) = n \leq \dim(y) = m$, we can initialize $z^{(0)}$ by sampling $z \,|\, x', y, \theta$ using Eq. (3.104) with $x' = y$.

**Preconditioning** For the SLDS, the precondition matrices are similarly a combination of those for the ARHMM Eqs. (3.52)-(3.54) and the LGSSM Eqs. (3.73)-(3.76).

$$D(\theta)_{\phi_k} = \mathrm{diag}(\phi_k) \ \text{ and } \ \Gamma(\theta)_{\phi_k} = 1 \tag{3.107}$$

$$D(\theta)_{A_k} = I_m \otimes Q_k \ \text{ and } \ \Gamma(\theta)_{A_k} = 0 \tag{3.108}$$

$$D(\theta)_{\psi_{Q_k}} = \frac{1}{2} I_n \otimes Q_k^{-1} \ \text{ and } \ \Gamma(\theta)_{\psi_{Q_k}} = \psi_{Q_k} \tag{3.109}$$

$$D(\theta)_C = I_n \otimes R \ \text{ and } \ \Gamma(\theta)_Q = 0 \tag{3.110}$$

$$D(\theta)_{\psi_{R_k}} = \frac{1}{2} I_m \otimes R_k^{-1} \ \text{ and } \ \Gamma(\theta)_{\psi_{R_k}} = \psi_{R_k} \ . \tag{3.111}$$

**Error Bounds** There are two primary challenges for the error analysis of the SLDS: (i) the forward and backward smoothing kernels for the SLDS are mixtures and (ii) the error from the finite-step blocked Gibbs sampler needs to be quantified. Conditions for contraction in the forward and backward smoothing random maps of switching models may follow from the conditions in (Cloez et al., 2015). Combining the convergence rate of the blocked Gibbs

sampler with the error bound is an area we leave for future work. Our experiments in Chapter 3.5 provide empirical evidence of the potential benefits of the algorithm.

## *3.5 Experiments*

We evaluate the performance of our proposed stochastic gradient estimators (Chapter 3.2) using both synthetic and real data. We first present the metrics we use for evaluation in Chapter 3.5.1. We then empirically evaluate the stochastic gradient error for our stochastic gradient estimators in Chapter 3.5.2. We finally apply our stochastic gradient estimators in SGMCMC in synthetic data in Chapter 3.5.3 and real data in Chapter 3.5.4.

### *3.5.1 Evaluation Metrics*

We evaluate the effectiveness of our samplers given a fixed computation budget by measuring (i) performance on a test set using heldout and predictive loglikelihoods, (ii) sample quality using kernel Stein discrepancy, and (iii) recovery of latent states and parameters using finite sample averages.

We assess the performance of our samplers on a test sequence using both heldout and predictive loglikleihoods. Given a sampled parameter value $\theta$ the heldout loglikelihood is

$$\sum_{t=1}^{T} \log p(y_t \mid y_{<t}, \theta) \approx \sum_{t=1}^{T} \sum_{i=1}^{N} w_{t-1}^{(i)} \log p(y_t \mid x_{t-1}^{(i)}, \theta), \tag{3.112}$$

and the $r$-step ahead predictive loglikelihood is

$$\sum_{t=1}^{T} \log p(y_{t+r} | y_{<t}, \theta) \approx \sum_{t=1}^{T} \sum_{i=1}^{N} w_{t-1}^{(i)} \log p(y_{t+r} | x_{t-1}^{(i)}, \theta), \tag{3.113}$$

where the exact value is calculated for analytic SSMs and the approximation is used for nonlinear SSMs with $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ obtained using a particle filter on the test sequence.

We measure the sample quality of our MCMC chains $\{\theta^{(k)}\}_{k=1}^K$ using the *kernel Stein discrepancy* (KSD) for equal compute time (Gorham and Mackey, 2017; Liu et al., 2016). We choose to use KSD rather than classic MCMC diagnostics such as effective sample size

(ESS) (Gelman et al., 2013), because KSD penalizes the bias present in our MCMC chains. Given a sample chain (after burnin and thinning) $\{\theta^{(k)}\}_{k=1}^{\tilde{K}}$, let $\hat{p}(\theta|y)$ be the empirical distribution of the samples

$$\hat{p}(\theta|y) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\theta=\theta^{(i)}} \ . \tag{3.114}$$

Then the KSD between $\hat{p}(\theta|y)$ and the posterior distribution $p(\theta|y)$ is

$$\mathrm{KSD}(\hat{p}, p) = \sum_{d=1}^{\dim(\theta)} \sqrt{\sum_{k,k'=1}^{\tilde{K}} \frac{\mathcal{K}_0^d(\theta^{(k)}, \theta^{(k')})}{\tilde{K}^2}}, \tag{3.115}$$

where

$$\begin{aligned}\mathcal{K}_0^d(\theta_i, \theta_{i'}) = \nabla_{\theta_d} \log p(\theta_i) \mathcal{K}(\theta_i, \theta_{i'}) \nabla_{\theta_d} \log p(\theta_{i'}) + \nabla \log p(\theta_{i'}) \nabla_x \mathcal{K}(\theta_i, \theta_{i'}) \\ + \nabla \log p(\theta_i) \nabla_y \mathcal{K}(\theta_i, \theta_{i'}) + \nabla_x \nabla_y \mathcal{K}(\theta_i, \theta_{i'})\end{aligned} \tag{3.116}$$

and $\mathcal{K}(\cdot, \cdot)$ is a valid kernel function. Following Gorham and Mackey (2017), we use the inverse multiquadratic kernel $\mathcal{K}(\theta, \theta') = (1 + \|\theta - \theta'\|_2^2)^{-0.5}$ in our experiments. Since Eq. (3.116) requires full gradient evaluations of $\log p(\theta|y)$ that are computationally intractable, we replace these terms with corresponding stochastic estimates using $g_\theta^{\mathrm{PF}}$.

To measure the recovery of discrete latent state variables $z_t$ when the true latent states are known (e.g. in synthetic experiments), we use normalized mutual information (NMI). NMI is an information theoretic measure of similarity between discrete assignments (Vinh et al., 2010).

$$\mathrm{NMI}(Z_i, Z_*) = \frac{I(Z_i, Z_*)}{\sqrt{H(Z_i) H(Z_*)}} \ , \text{ with } Z_i = (z_1^{(i)}, \ldots, z_T^{(i)}) \ , \tag{3.117}$$

where $I(X, Y)$ is mutual information and $H(X)$ is entropy. NMI is maximized at 1 when the assignments are equal up to a permutation and minimized at 0 when the assignments share no information. This serves as 'clustering' or segmentation metric for measuring the coherence between our model's inferred latent states and the true latent states.

To measure the recovery of continuous latent state variables $x_t$ when the true latent states are known, we use root mean-squared error (RMSE) $\mathrm{RMSE}(x, x') = \sum_t \|x_t - x_t'\|_2$.

For synthetic data, we also measure the MSE of SGLD's finite sample averages $\hat{\theta}_{(K)} = \sum_{k \leq K} \theta^{(k)}/K$ to the true parameters $\theta^*$.

*Hyperparameters and Initialization of SGMCMC*

As with all gradient-based methods, our SGMCMC methods require a hyper-parameter search over the fixed step-size tuning parameter $h$. We present results for the best step-size as assessed via KSD. As the loglikelihood for SSMs is non-convex, initialization is important. For the HMM and ARHMM, we initialize the parameters $\Pi, A, Q$ using $z$ given from $K$-means clustering of the observations $y$ (or $y_{t-p:t}$). For the LGSSM, SVM, and GARCH, we initialize the parameters from the prior. For the mixed-type SLDS, we first sample $R$ from the prior and initialize $\Pi, A, Q$ using $z$ from $K$-means. Finally, in our experiments, we use flat and non-informative priors for $\theta$. See the Appendix for more details.

For batch MCMC, we consider block-Gibbs sampling (Gibbs) and unadjusted Langevin Monte-Carlo – both with preconditioning (RLD) and without precondition (LD). Note that LD and RLD are SGLD and SGRLD with $S = T$.

### *3.5.2 Stochastic Gradient Error*

We first empirically test the error of our buffered stochastic gradient estimate $\hat{g}_{S,B}$ when analytic message passing is possible. We then empirically test the error of our particle buffered stochastic gradient estimate $g^{\text{PF}}_{S,B,N}$ for nonlinear SSMs. These experiments illustrate the geometric decay in buffer size $B$ of the error bounds of Chapter 3.3.

*Analytic Message Passing*

**ARHMM** We consider synthetic data generated from a 2-state ARHMM in two dimensions $m = 2$. The true model parameters $\theta^*$ are

$$\Pi = \begin{bmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{bmatrix} \quad , \quad Q_1 = Q_2 = 0.1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_1 = 0.9 \cdot \begin{bmatrix} \cos(-\vartheta) & -\sin(-\vartheta) \\ \sin(-\vartheta) & \cos(-\vartheta) \end{bmatrix} \quad , \quad A_2 = 0.9 \cdot \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix} \quad .$$

The model's two states are alternating rotations of $y \in \mathbb{R}^2$ with angle $\vartheta = \pi/4$ and the latent state sequence has a high transition rate $\Pr(z_t \neq z_{t-1}) = 0.9$. We generate time series of length $T = 10^3$.

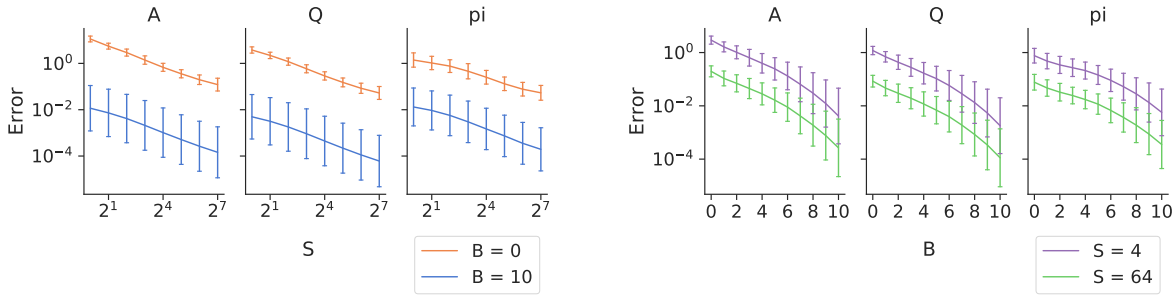


Figure 3.3: Stochastic gradient error $\mathbb{E}_{\mathcal{S}}\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$. (Left) varying subsequence length $S$ for no-buffer $B = 0$ and buffer $B = 10$. (Right) varying buffer size $B$ for $S = 4$ and $S = 64$ subsequence lengths. Error bars are SD over 100 datasets.

Figure 3.3 are plots of the stochastic gradient error $\mathbb{E}_{\mathcal{S}}\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$ between the unbiased and buffered estimates evaluated at the true model parameters $\theta = \theta^*$. From Figure 3.3 (left), we see that the error decays $\mathcal{O}(1/S)$ and that the error in estimates without buffering $B = 0$ (orange) are orders of magnitude larger than the estimates with moderate buffering $B = 10$ (blue). From Figure 3.3 (right), we see that the error decays geometrically in buffer size $\mathcal{O}(L^B)$.

**LGSSM** We consider synthetic data from a LGSSM with observations and latent state dimension $m = n = 2$. In particular, we consider, a rotating state sequence with noisy observations. The true model parameter $\theta^*$ are

$$A = 0.7 \cdot \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix} \quad , \quad Q = 0.1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad ,$$

where $\vartheta = \pi/4$. Because the transition error $Q$ is smaller than the emission error $R$, inclusion of previous and future observations is necessary to accurately infer the continuous latent state $x_t$. We generate time series of length $T = 10^3$.

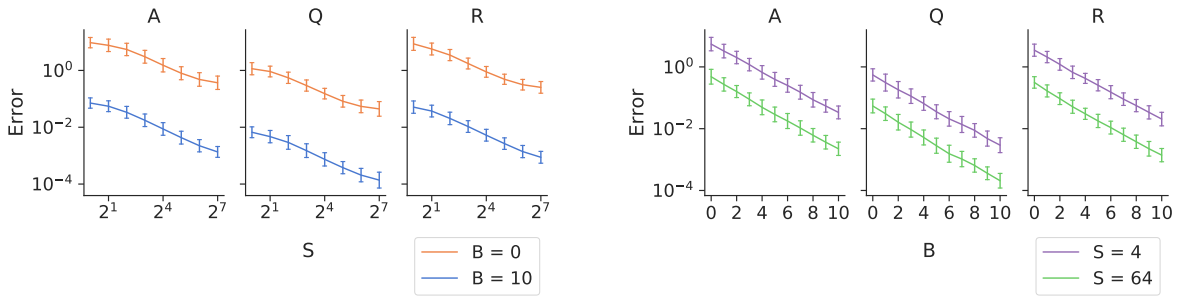


Figure 3.4: Stochastic gradient error $\mathbb{E}_{\mathcal{S}}\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$. (Left) varying subsequence length $S$ for no-buffer $B = 0$ and buffer $B = 10$. (Right) varying buffer size $B$ for $S = 4$ and $S = 64$ subsequence lengths. Error bars are SD over 100 datasets.

Figure 3.4 are plots of the stochastic gradient error $\mathbb{E}_{\mathcal{S}}\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$ between the unbiased and buffered estimates evaluated at the true model parameters $\theta = \theta^*$. Similar to the ARPHMM, we see that the error decays $\mathcal{O}(1/S)$ and that moderate buffering (e.g. $B = 10$) deceases the error by orders of magnitude in Figure 3.4 (left). And we see that the error decays geometrically in buffer size $\mathcal{O}(L^B)$ in Figure 3.4 (right).

*Blocked Gibbs*

**SLDS** We now consider synthetic data from a model we can view as switching extension of the LGSSM or as a noisy version of the ARHMM. The true model parameters $\theta^*$ are

$$\Pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \quad Q_1 = Q_2 = 0.1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 0.1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$A_1 = 0.9 \cdot \begin{bmatrix} \cos(-\vartheta) & -\sin(-\vartheta) \\ \sin(-\vartheta) & \cos(-\vartheta) \end{bmatrix}, \quad A_2 = 0.9 \cdot \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix},$$

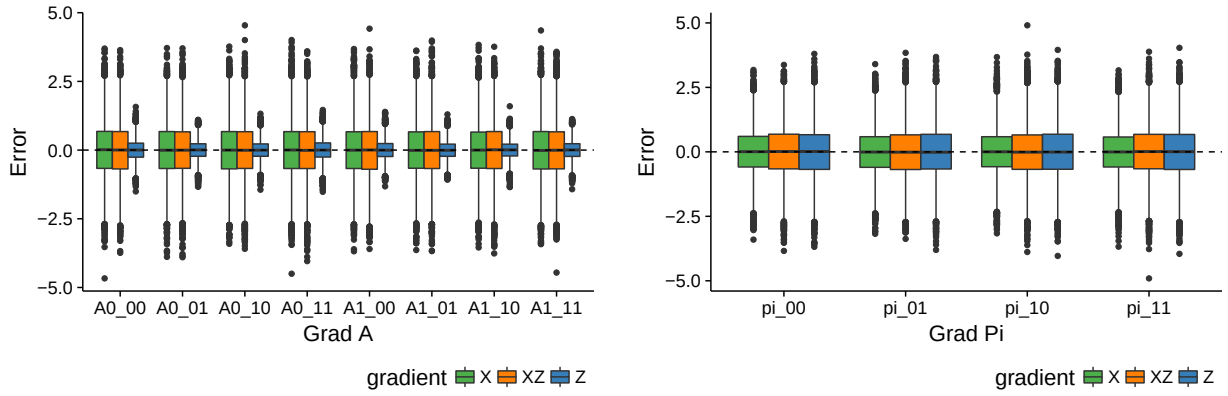


Figure 3.5: SLDS gradient error for the different estimators Eqs. (3.100)-(3.102). (Left) Boxplots of $\hat{g}_{S,B}(\theta)_A - g(\theta)_A$. (Right) Boxplots of $\hat{g}_{S,B}(\theta)_\Pi - g(\theta)_\Pi$.

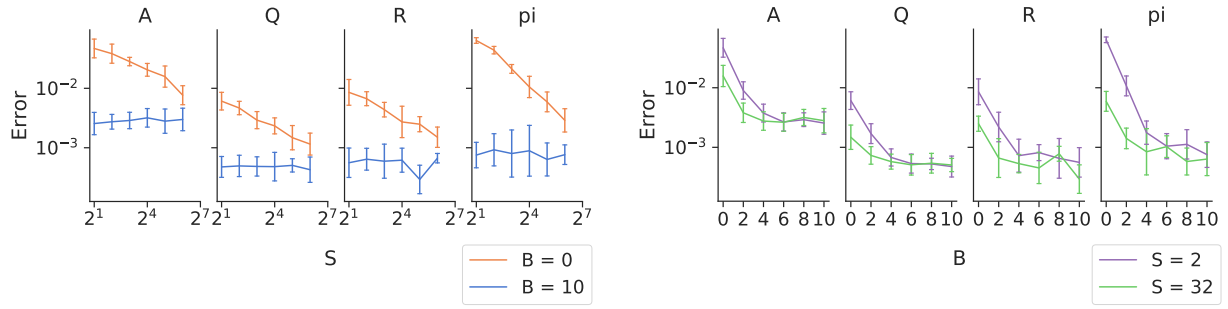


Figure 3.6: Stochastic gradient error $\mathbb{E}_{\mathcal{S}}\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$ for `z Gradient`. (Left) error varying subsequence length $S$ for no-buffer $B = 0$ and buffer $B = 4$. (Right) error varying buffer size $B$ for small $S = 2$ and long $S = 32$ subsequences. Error bars are SD over 100 datasets.

where again $\vartheta = \pi/4$. We generate sequences of length $T = 10^3$.

We first compare the variance of the three difference Monte-Carlo gradient estimators for SLDS: using $(x, z)$ samples (`xz Gradient`) as in Eq. (3.100), only using $z$ samples (`z Gradient`) as in Eq. (3.101), and only using $x$ samples (`x Gradient`) as in Eq. (3.102). Figure 3.5 presents boxplots of $\hat{g}_{S,B}(\theta) - g(\theta)$ for the three different estimators at $\theta = \theta^*$.

From Figure 3.5 (left), we see that `z Gradient` (blue) has much lower variance than the other two estimators for the gradient of $A$. This also holds for the gradients of $Q$ and $R$ (see Appendix). From Figure 3.5 (right), we see that all three estimators have similar variance for the gradient of $\Pi$ (with `x Gradient` (green) slightly better than the other two). This agrees with intuition described in Chapter 3.4.6. Because `z Gradient` has lower variance than the other two estimators, we can use larger step-sizes, leading to faster convergence and mixing.

Figure 3.6 are plots of the stochastic gradient error $\mathbb{E}_{\mathcal{S}}\|\bar{g}(\theta) - \hat{g}_{S,B}(\theta)\|_2$ between the unbiased and buffered estimates (for `z Gradient`) evaluated at the true model parameters $\theta = \theta^*$. For short buffered subsequences (e.g. small $S$ and $B$), the error decays as expected $\mathcal{O}(L^B/S)$; however, for longer buffered subsequences the error is dominated by the Monte Carlo error in the number of Gibbs steps used in sampling $z$ for calculating $\hat{g}_{S,B}$ in Eq. (3.101).

*Particle Smoothers*

We now compare the error of our particle buffered stochastic gradient estimates $g^{\text{PF}}_{S,B,N}$ using a buffered subsequence with $S = 16$, while varying $B$ and $N$ on synthetic data. We generated synthetic data of length $T = 256$ using $(A = 0.9, Q = 0.5, R = 1.0)$ for the LGSSM, $(\phi = 0.9, \sigma = 0.5, \tau = 0.5)$ for the SVM, and $(\alpha = 0.1, \beta = 0.8, \gamma = 0.05, \tau = 0.3)$ for the GARCH model.

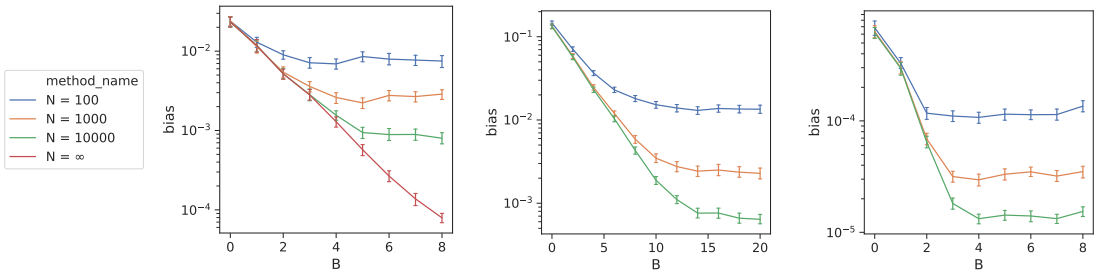


Figure 3.7: Stochastic gradient bias varying buffer size $B$ for $S = 16$ for different values of $N$. (left) LGSSM $\phi$, (middle) SVM $\phi$, (right) GARCH $\beta$. Error bars are 95% CI over 1000 replications.

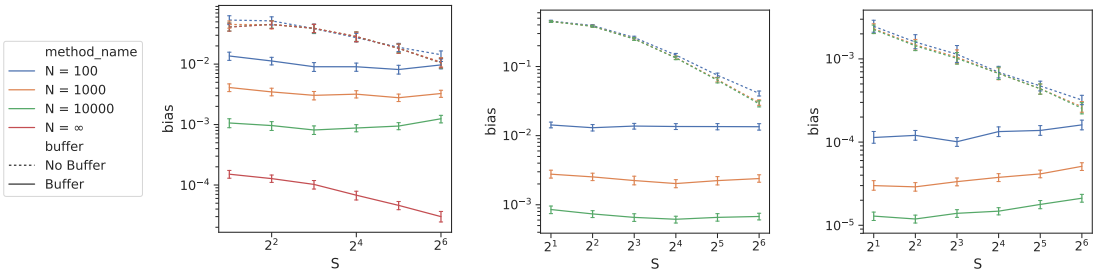


Figure 3.8: Stochastic gradient bias varying subsequence size $S$ for No Buffer ($B = 0$) and Buffer ($B > 0$) for different values of $N$. (left) LGSSM $\phi$, (middle) SVM $\phi$, (right) GARCH $\beta$. The buffer size $B = 8$ for LGSSM and GARCH and $B = 16$ for the SVM. Error bars are 95% CI over 1000 replications.

Figures 3.7-3.9 display the bias of our particle buffered stochastic gradient $g_\theta^{\mathrm{PF}}(S, B, N)$ and $g_\theta$ averaged over 1000 replications. We evaluate the gradients at $\theta$ equal to the data generating parameters. We vary the buffer size $B \in [0, 16]$, the subsequence size $S \in [1, T]$ and the number of samples $N \in \{100, 1000, 10000\}$. For the LGSSM, we also consider $N = \infty$, by calculating $g_\theta^{\mathrm{PF}}(S, B, \infty) = \hat{g}_{S,B}$ using the Kalman smoother, which is tractable in the linear setting. We calculate $g_\theta$ using the Kalman smoother for the LGSSM, and use $g_\theta \approx g_\theta^{\mathrm{PF}}(T, 0, 10^7)$ for the SVM and the GARCH model, assuming that $N = 10^7$ particles is sufficient for an accurate approximation in these 1-dimensional settings.

Figure 3.7 shows the bias as we vary the buffer size $B$ for different $N$ and $S = 16$. From Figure 3.7, we see the trade-off between the buffering error (I) and the particle error (III) in the bias bound, Eq. (3.30) of Theorem 6. For all $N$, when $B$ is small, the buffering error (I) dominates, and therefore the MSE decays exponentially as $B$ increases. However for $N < \infty$, the particle error (III) dominates for larger values of $B$. In fact, the bias slightly increases due to particle degeneracy, as $|\mathcal{S}^*| = S + 2B$ increases with $B$. For $N = \infty$ in the LGSSM case, we see that the bias continues to decreases exponentially with large $B$ as there is no particle filter error when using the Kalman filter.

Figure 3.8 shows the bias as we vary the subsequence size $S$ for different $N$ and with and

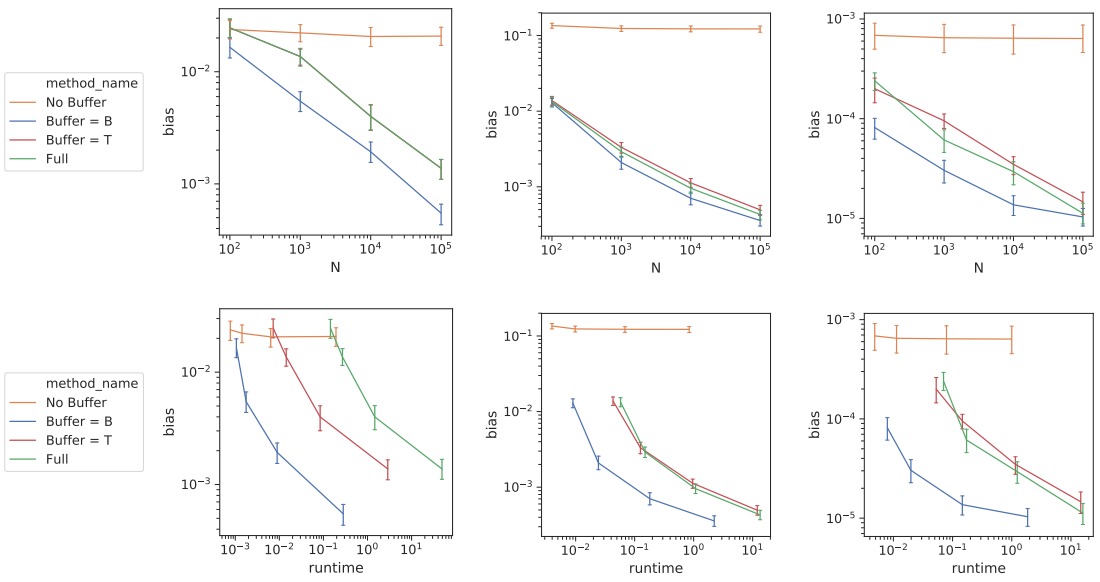


Figure 3.9: Stochastic gradient bias varying $N$ for different $S, B$. (left) LGSSM $\phi$, (middle) SVM $\phi$, (right) GARCH $\beta$. (top) $x$-axis is $N$, (bottom) $x$-axis is runtime in seconds. **No Buffer** is $g^{\mathrm{PF}}(16, 0, N)$, **Buffer** $B = B$ is $g^{\mathrm{PF}}(16, B, N)$, **Buffer** $B = T$ is $g^{\mathrm{PF}}(16, T, N)$, and **Full** is $g^{\mathrm{PF}}(T, T, N)$. The moderate buffer size $B = 8$ for LGSSM and GARCH and $B = 16$ for the SVM. Error bars are 95% CI over 1000 replications.

without buffering. We see that buffering helps regardless of subsequence size (as the bias for all buffered methods are lower than the no buffer methods for all $S \in [2, 64]$). We also see that increasing $S$ can increase the bias for fixed $N$ (when buffering) as the particle error (III) dominates.

Figure 3.9 shows the bias as we vary the number of particles $N$ for the four different methods correspond to Table 3.1. In the top row, we compare the bias against $N$ and in the bottom row, we compare the bias against the runtime required to calculate $g_\theta^{\mathrm{PF}}$. We see that the method without buffering (orange) is significantly biased regardless of $N$, where as buffering with moderate $B$ (blue), buffering with large $B = T$ (red), and using the full sequence (green) have similar (lower) bias as we increase $N$. However the runtime plots show that buffering with moderate $B$ takes significantly less time.

In summary, Figures 3.7-3.9 show that buffering cannot be ignored in these three example models: there is high bias for $B = 0$ regardless of $N$. However, buffering has diminishing returns when $B$ is excessively large relative to $N$.

**Comparison with other particle smoothers** We now compare different particle smoothers on the LGSSM data. Figure 3.10 compares the stochastic gradient bias of the naive PF with PaRIS (Olsson and Westerborn, 2017) on the LGSSM data.

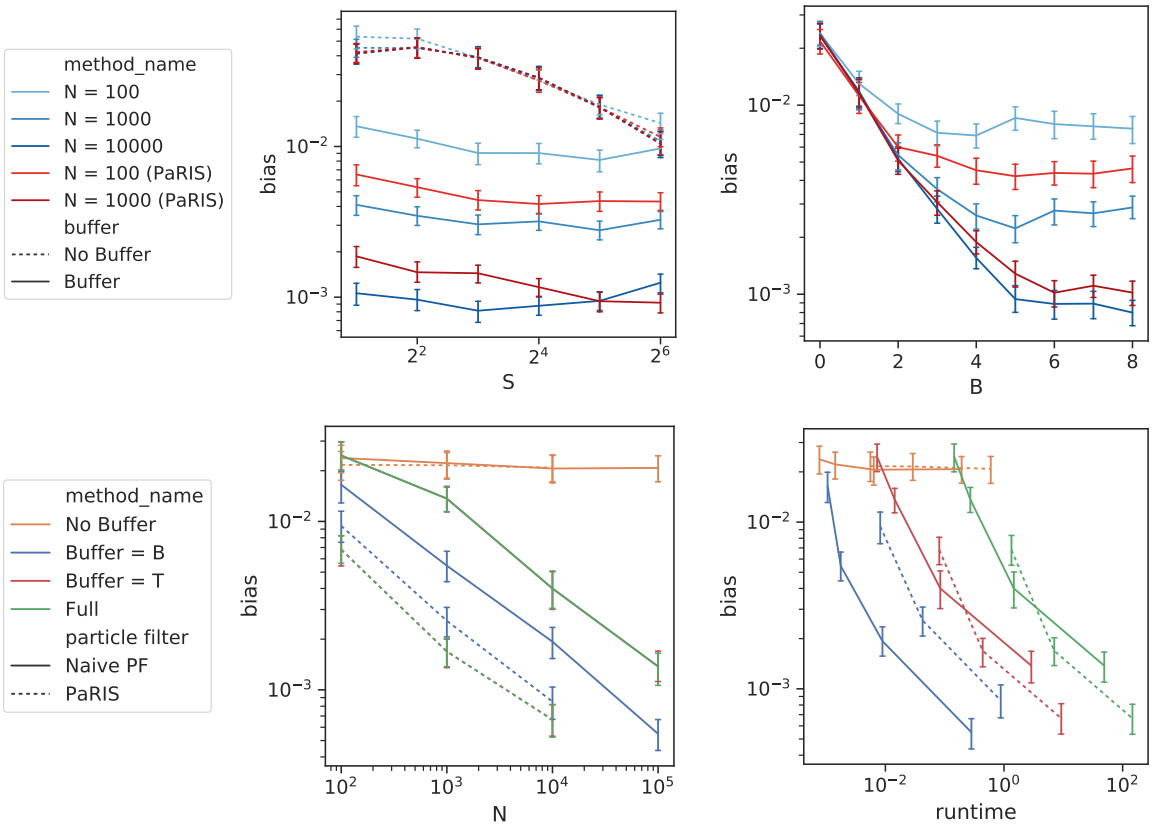


Figure 3.10: Stochastic gradient bias varying $B, S, N$ for the naive PF and PaRIS on the LGSSM data. (Top-left) bias vs $S$, (top-right) bias vs $B$, (bottom-left) bias vs $N$, (bottom-right) bias vs runtime in seconds. Error bars are 95% CI over 1000 replications.

From Figure 3.10 (top) and (bottom-left), we see that the naive PF (blue or solid line) performs similarly to PaRIS (red or dashed line) when using 10 time the number of particles

(e.g. naive PF with $N = 10000$ performs similarly to PaRIS with $N = 1000$). However, Figure 3.10 (bottom-right) shows that PaRIS performs similarly to the naive PF for small subsequence lengths (Buffer = $B$ in blue), while taking $\approx 10$ times longer to run (due to additional overhead).

### *3.5.3 Synthetic Data: SGMCMC*

Having examined the stochastic gradient bias, we now examine using our buffered stochastic gradient estimators in SGMCMC (Algorithm 1) on synthetic data. We first present experiments using $\hat{g}_{S,B}$ for SSMs with analytic message passing. We then present experiments using $g^{\text{PF}}_{S,B,N}$ using the particle smoother.

#### *Analytic Message Passing*

For the synthetic data, we use the models and parameters introduced in Chapter 3.5.2.

**ARHMM** In Figures 3.11 and 3.12, we compare subsequence-based MCMC methods: SGLD (no-buffer and buffer) and SGRLD (no-buffer and buffer), with full-sequence MCMC methods: LD, RLD, and Gibbs. We fit our samplers on one training sequence and evaluate performance on one test sequence. We consider two training sequences of lengths $T = 10^4$ and $T = 10^6$ and evaluate on the same test sequence of length $T = 10^4$. For the SGMCMC methods we use a subsequence size of $S = 2$ and a buffer size of $B = 0$ (no-buffer) or $B = 2$ (buffer). We ran the subsequence methods for 6 hours and full-sequence methods for 144 hours.

From Figure 3.11, we see that our buffered SGMCMC (blue) helps convergence and mixing orders of magnitude faster than the full-sequence gradient MCMC (green). We also see that buffering is necessary to properly estimate $\Pi$ as the no-buffer SGMCMC methods (orange) do not properly learn $\Pi$ resulting in large oscillations in the heldout loglikelihood and MSE. We also see that preconditioning helps convergence and mixing as SGRLD (solid)

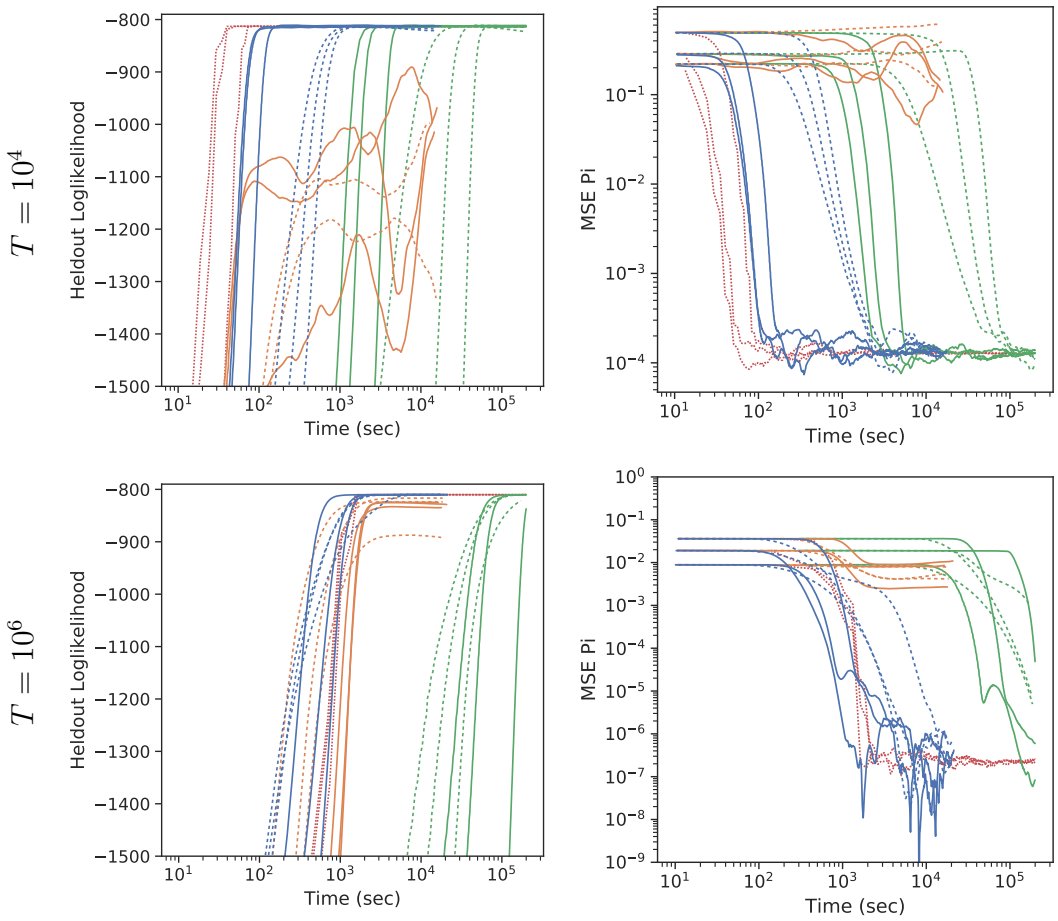


Figure 3.11: Metrics vs Runtime on ARHMM data with $T = 10^4$ (top), $T = 10^6$ (bottom), for different methods over different initializations: (Gibbs), (Full), (No Buffer) and (Buffer) SGMCMC. For SGMCMC methods, solid (—) and dashed (- -) lines indicate SGRLD and SGLD respectively. The different metrics are: (left) heldout loglikelihoood and (right) transition matrix estimation error $MSE(\hat{\Pi}^{(s)}, \Pi^*)$.

outperforms SGLD (dashed). Although Gibbs outperforms SGMCMC for $T = 10^4$, Gibbs performs worse for $T = 10^6$, as each iteration requires a full pass over the data set.

Figure 3.12 are boxplots comparing the marginal distribution for the different methods on the synthetic ARHMM data $T = 10^6$. From Figure 3.12, we see that SGRLD with buffering in 6 hours is comparable to RLD or Gibbs in 144 hours; however, SGRLD without buffering

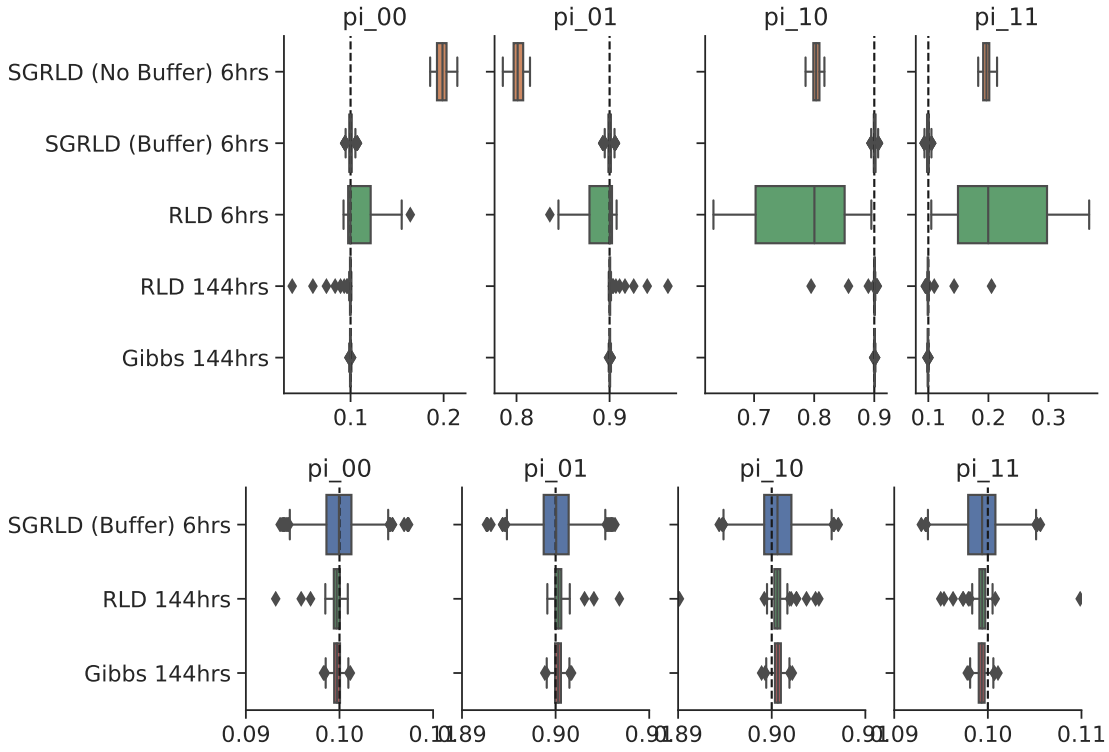


Figure 3.12: Boxplot of MCMC samples for ARHMM data $T = 10^6$. (Top) comparison of all samplers, (bottom) zoom-in for top three. The half of each chain is discarded as burn-in. SGRLD with buffering in 6 hrs is comparable to RLD or Gibbs in 144 hrs.

is biased and RLD in 6 hours has not had enough time to mix.

Table 3.2 displays the KSD of the samples to the posterior after discarding half the samples as burn-in. The standard deviation is over MCMC chains with different initializations. Although RLD and Gibbs perform well for $T = 10^4$, both perform worse for larger $T = 10^6$ due to the increased time between samples. We also see that the non-buffered methods do poorly for all $T$ due to sampling from the incorrect distribution. Although SGLD (buffer) and SGRLD (buffer) perform comparably after burn-in, Figure 3.11 suggests SGRLD converges more rapidly.

Table 3.2: $\log_{10}$(KSD) by variable of ARHMM samplers at 6 hrs. Mean and (SD) over runs in Figure 3.11.

| | Sampler | $\pi$ | $A$ | $\Sigma$ |
|---|---|---|---|---|
| $T = 10^4$ | SGLD (No Buffer) | 3.15 (0.46) | 2.47 (0.51) | 2.33 (0.30) |
| | SGLD (Buffer) | 0.99 (0.13) | 1.60 (0.20) | 1.80 (0.13) |
| | LD | 1.77 (0.72) | 1.86 (0.32) | 2.12 (0.36) |
| | SGRLD (No Buffer) | 3.15 (0.39) | 2.02 (0.24) | 1.91 (0.24) |
| | SGRLD (Buffer) | 0.89 (0.04) | 1.53 (0.10) | 1.60 (0.30) |
| | RLD | 0.67 (0.27) | 2.02 (0.14) | 1.60 (0.18) |
| | Gibbs | 0.36 (0.07) | 1.30 (0.20) | 0.61 (0.13) |
| $T = 10^6$ | SGLD (No Buffer) | 4.73 (0.07) | 4.07 (0.22) | 3.67 (0.25) |
| | SGLD (Buffer) | 2.62 (0.06) | 3.30 (0.20) | 2.77 (0.31) |
| | LD | 3.59 (0.22) | 4.73 (0.33) | 4.78 (0.34) |
| | SGRLD (No Buffer) | 4.75 (0.15) | 4.02 (0.06) | 3.61 (0.12) |
| | SGRLD (Buffer) | 2.27 (0.08) | 3.38 (0.08) | 2.89 (0.09) |
| | RLD | 3.31 (0.05) | 4.22 (0.12) | 3.56 (0.07) |
| | Gibbs | 3.17 (0.30) | 4.18 (0.07) | 3.30 (0.07) |

**LGSSM** In Figures 3.13 and 3.14, we compare SGLD (no-buffer and buffer), SGRLD (no-buffer and buffer), LD, RLD, and a blocked Gibb sampler. We fit our samplers on one training sequence and evaluate performance on one test sequence. We consider two training sequences of lengths $T = 10^4$ and $T = 10^6$ and evaluate on the same test sequence of length $T = 10^4$. For the SGMCMC methods, we use a subsequence size of $S = 20$ with $B = 0$ (no buffer) and $B = 10$ (buffer). We see that even with a large subsequence size, buffering is crucial for accurate inference as SGMCMC methods without buffering converge to a different stationary distribution than the posterior.

In Table 3.3, we evaluate the KSD of the different MCMC methods. We see that

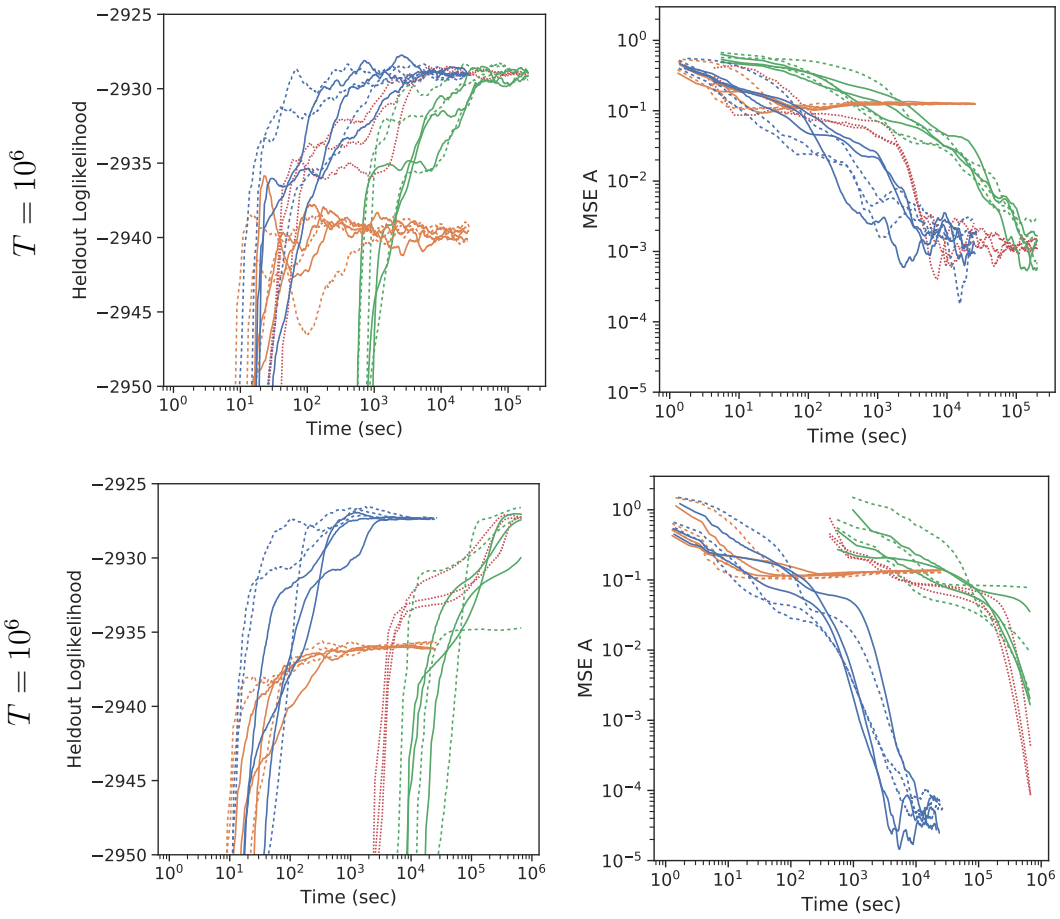


Figure 3.13: Metrics vs Runtime on LGSSM with $T = 10^4$ (top), $T = 10^6$ (bottom) for different methods: (Gibbs), (Full), (No Buffer) and (Buffer) SGMCMC. For SGMCMC methods, solid (—) and dashed (- -) lines indicate SGRLD and SGLD respectively. The different metrics are: (left) heldout loglikelihoood and (right) transition matrix estimation error $MSE(\hat{A}^{(s)}, A^*)$.

SGMCMC with buffering slightly outperforms the full sequence methods for $T = 10^4$ and significantly outperforms the full sequence methods for $T = 10^6$, while SGMCMC without buffering performs poorly due to bias.

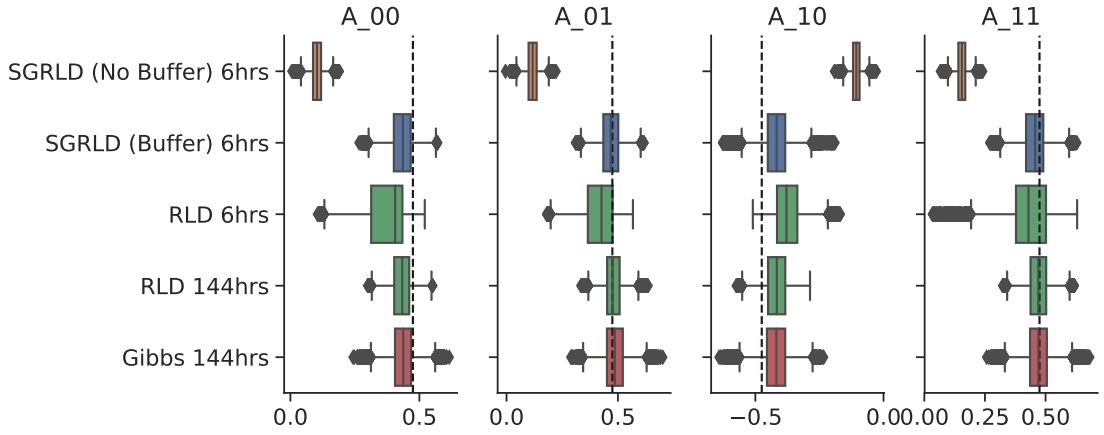


Figure 3.14: Boxplot of MCMC samples of transition matrix $A$ for LGSSM data $T = 10^4$. SGRLD with buffering in 6 hours is comparable to RLD or Gibbs in 144 hours. SGRLD without buffering is biased and RLD in 6 hours has not fully mixed.

*Blocked Gibbs*

**SLDS** In Figure 3.15, we compare SGRLD (with buffer) using each of the gradient estimators Eqs. (3.100)-(3.102), and a blocked Gibb sampler. We run our samplers on one training sequence and evaluate performance on another test sequence. For all SGRLD samplers, we used subsequence size of $S = 10$ and $B = 10$. As the marginal loglikelihood is not available in closed form for SLDSs, we instead use a Monte Carlo approximation of the EM lower bound $\log p(y \,|\, \theta) \geq \mathbb{E}_{x,z|y,\theta}[\log p(y, x, z \,|\, \theta)]$ where the expectation is approximated with samples of $x, z$ drawn using blocked Gibbs for each fixed $\theta$. From Figure 3.15, we see that SGRLD methods perform similarly to Gibbs for $T = 10^4$, but vastly outperform Gibbs for $T = 10^6$.

*Particle Smoothers*

We now evaluate our particle buffered stochastic gradient $g^{\mathrm{PF}}_{S,B,N}$ when used in SGLD.

**LGSSM** To assess the effect of using particle filters with buffered stochastic gradients, we first focus on SGLD on synthetic LGSSM data, where calculating $\widehat{g}_\theta(S, B)$ is possible. We

Table 3.3: $\log_{10}$(KSD) by variable of LGSSM samplers at 6 hrs. Mean and (SD) over runs in Figure 3.13.

| | Sampler | $A$ | $Q$ | $R$ |
|---|---|---|---|---|
| $T = 10^4$ | SGLD (No Buffer) | 2.39 (0.01) | 1.73 (0.03) | 1.48 (0.03) |
| | SGLD (Buffer) | 0.88 (0.11) | 0.41 (0.11) | 0.86 (0.08) |
| | LD | 0.99 (0.13) | 1.12 (0.19) | 1.10 (0.17) |
| | SGRLD (No Buffer) | 2.38 (0.01) | 1.70 (0.02) | 1.43 (0.02) |
| | SGRLD (Buffer) | 0.85 (0.08) | 0.18 (0.12) | 0.77 (0.14) |
| | RLD | 0.99 (0.12) | 0.90 (0.19) | 1.10 (0.17) |
| | Gibbs | 0.74 (0.20) | 0.33 (0.18) | 1.06 (0.27) |
| $T = 10^6$ | SGLD (No Buffer) | 4.32 (0.01) | 3.79 (0.02) | 3.50 (0.02) |
| | SGLD (Buffer) | 2.30 (0.19) | 1.61 (0.18) | 2.84 (0.03) |
| | LD | 4.26 (0.35) | 4.00 (0.39) | 4.14 (0.19) |
| | SGRLD (No Buffer) | 4.27 (0.01) | 3.77 (0.02) | 3.23 (0.03) |
| | SGRLD (Buffer) | 2.17 (0.33) | 1.64 (0.21) | 3.03 (0.12) |
| | RLD | 4.34 (0.23) | 3.76 (0.25) | 4.03 (0.23) |
| | Gibbs | 3.46 (0.28) | 3.52 (0.14) | 3.50 (0.28) |

generate training sequences of length $T = 10^3$ or $10^6$ and test sequences of length $T = 10^3$ using $(A = 0.9, Q = 0.5, R = 1.0)$.

We consider three pairs of different gradient estimators: **Full** ($S = T$), **Buffered** ($S = 40, B = 10$) and **No Buffer** ($S = 40, B = 0$) each with $N = 1000$ particles using the particle filter and with $N = \infty$ using the Kalman filter. To select the stepsize, we performed a grid search over $\epsilon \in \{1, 0.1, 0.01, 0.001\}$ and selected the method with smallest KSD to the posterior on the training set. We present the KSD results (for the best $\epsilon$) in Table 3.4 and trace plots of the metrics in Figure 3.16.

From Figure 3.16, we see that the methods without buffering ($B = 0$) have lower heldout

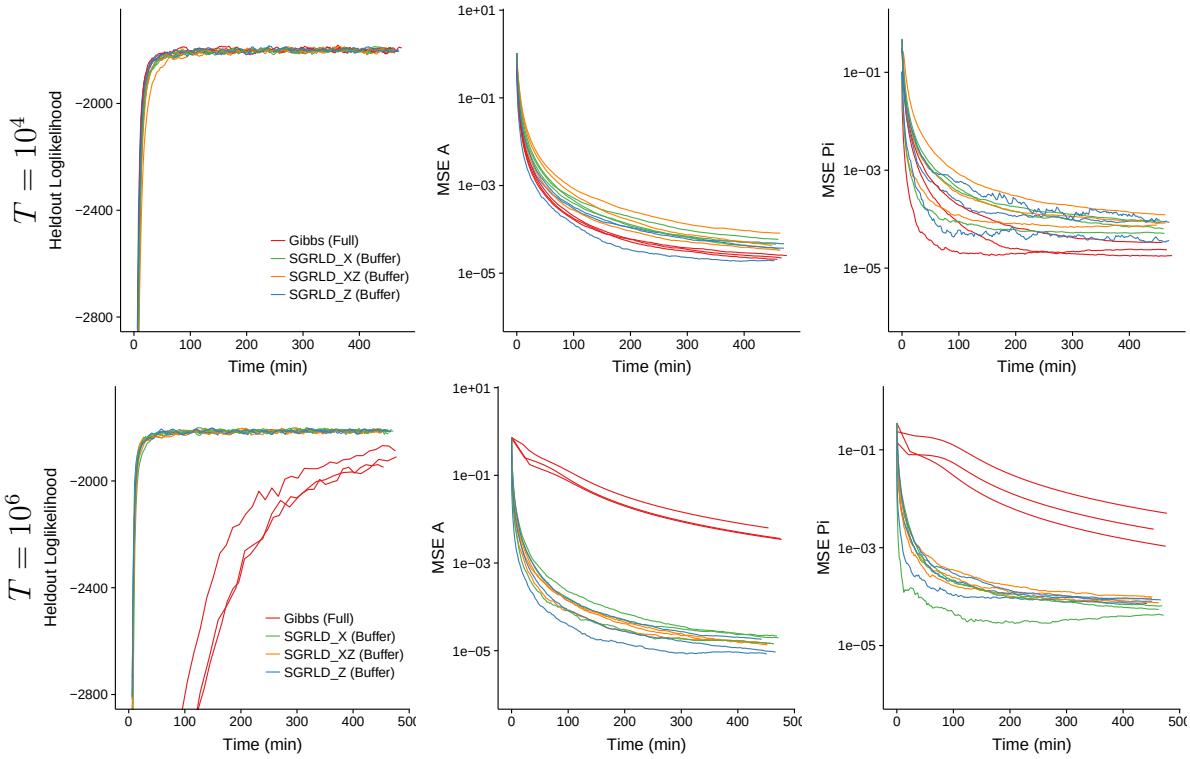


Figure 3.15: Metrics vs Runtime on SLDS data for different inference methods: Gibbs, SGRLD X, SGRLD XZ, and SGLRD Z. (Top) $T = 10^4$ (Bottom) $T = 10^6$. The metrics are: (left) heldout loglikelihood, (center) estimation error $MSE(\hat{A}^{(s)}, A^*)$, (right) estimation error $MSE(\hat{\Pi}^{(s)}, \Pi^*)$.

loglikelihoods on the test sequence and have higher MSE as they are biased. We also see that the full sequence methods ($S = T$) perform poorly for large $T = 10^6$.

The KSD results further support this story. Table 3.4 presents the mean and standard deviation on our estimated $\log_{10}$ KSD for $\theta$. Tables of the marginal KSD for individual components of $\theta$ can be found in the Appendix. The methods without buffering have larger KSD, as the inherent bias of $\widehat{g}_\theta(S, B = 0)$ led to an incorrect stationary distribution. The full sequence methods perform poorly for $T = 10^6$ because of a lack of samples that can be computed in a fixed runtime.

In the Appendix, we present similar results on synthetic SVM and GARCH data. Also

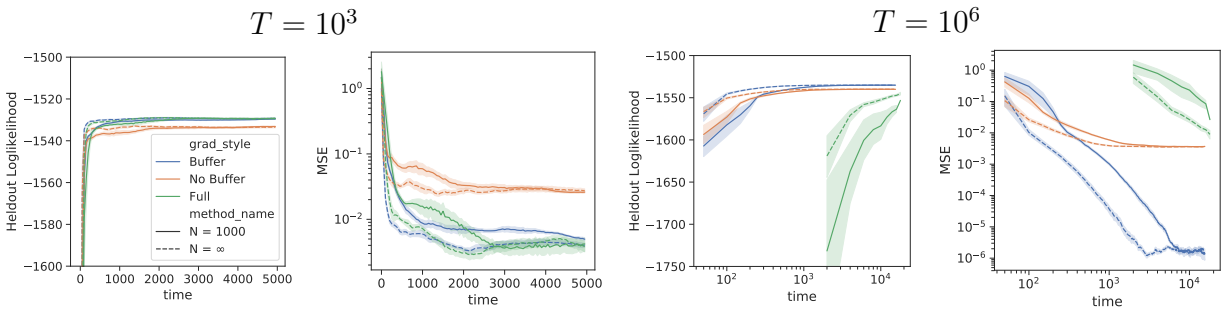


Figure 3.16: Comparison of SGLD with different gradient estimates on synthetic LGSSM data: $T = 10^3$ (left-pair), $T = 10^6$ (right-pair). (Left) heldout-loglikelihood, (Right) MSE of estimated posterior mean to true $A = 0.9$.

in the Appendix, we present results on LGSSM in higher dimensions. As is typical in the particle filtering literature, the performance degrades with increasing dimensions for $N$ fixed.

### *3.5.4 Real Data: SGMCMC*

Finally, we present experiments applying our buffered stochastic gradient estimators to real data sets. We consider ion channel recordings (Rosenstein et al., 2013), canine seizure data (Davis et al., 2016), city weather data (Beniaguev, 2017), and EUR-USD exchange rate data (Aicher et al., 2019c).

#### *Ion Channel Recordings*

We investigate the behavior of SGMCMC samplers on ion channel recording data. In particular, we consider a 1MHz recording of a single alamethicin channel (Rosenstein et al., 2013). This data was previously investigated using a Bayesian nonparametric HMM in (Palla et al., 2014) and (Tripuraneni et al., 2015). In that work, the authors downsample the data by a factor of 100 and only used $10,000$ and $2,000$ observations due to the challenge of scaling computations to the full sequence. We present the results for fitting a Gaussian HMM with $K = 5$ on the data without downsampling (10 million observations), where Gibbs sampling

Table 3.4: KSD for Synthetic LGSSM. Mean and SD.

| | | | $\log_{10}$KSD | |
|---|---|---|---|---|
| $S$ | $B$ | $N$ | $T = 10^3$ | $T = 10^6$ |
| $T$ | – | 1000 | 0.85 (0.08) | 4.92 (0.40) |
| | | $\infty$ | **0.64 (0.17)** | 4.85 (0.36) |
| 40 | 0 | 1000 | 1.58 (0.03) | 4.68 (0.10) |
| | | $\infty$ | 1.55 (0.03) | 4.68 (0.11) |
| 40 | 10 | 1000 | **0.68 (0.25)** | **3.43 (0.19)** |
| | | $\infty$ | **0.61 (0.21)** | **3.25 (0.29)** |

runs into memory issues. Figure 3.17 presents our results, after applying a log-transform and normalizing the observations. We train on the first 90% and evaluate on the last 10%. For our SGMCMC methods we use a subsequence size of $S = 10$ and a buffer size of $B = 0$ (no-buffer) or $B = 10$ (buffer). In addition to heldout loglikelihood, we also evaluate on 10-step ahead predictive loglikelihood $\sum_t \log p(y_{t+10} \mid \theta, y_{\leq t})$, which is more sensitive to $\Pi$. We see that SGRLD quickly converges compared to SGLD. Although the buffered methods take longer to compute ($S + 2B = 30$ vs $S = 10$), we see that buffering is necessary to perform well. In the Appendix, we present results comparing SGMCMC methods with Gibbs sampling on a downsampled version.

*Canine Seizure iEEG*

We now consider applying SGMCMC samplers to intracranial EEG (iEEG) data. In particular, we consider data from a study on canines with epilepsy available at `ieeg.org` (Davis et al., 2016). We focus on one canine, which over the course of 45.1 days was continuously monitored at 200Hz over 16 channels and recorded 90 seizures. This data was analyzed in

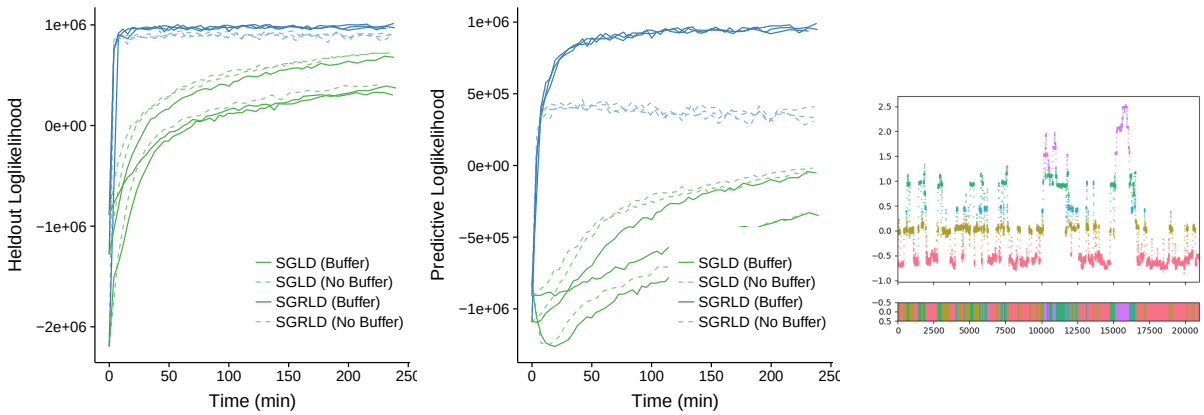


Figure 3.17: Ion Channel Recordings: (Left) heldout loglikelihood vs runtime. (Center) 10-step predictive loglikelihood $\sum_t \log \Pr(y_{t+10} \mid \theta, y_{\leq t})$ vs runtime. (Right) segmentation by SGRLD (Buffer).

prior work that compared a baseline ARHMM to nonparametric extensions using Gibbs sampling (Wulsin, 2013). Following (Wulsin, 2013), we process the data into 4 minute windows around each seizure to focus on the seizure dynamics resulting in 90 time series of 48,000 points in $\mathbb{R}^{16}$. We consider two models: (i) an ARHMM with $K = 5$ latent states and $p = 5$ lags and (ii) a SLDS model with $K = 5$ latent states and $n = 1$ latent continuous dimension. For both models we treat each channel independently. We perform an 80-20 train-test split over 90 seizures, running inference on the training set and evaluating log-likelihood on the heldout test set. We compare SGLD and SGRLD samplers with $S = 100$ and $B = 10$ with the baseline Gibbs sampler on the full data set. Because of the large data size, we also consider a *subset* Gibbs sampler that only uses 10% of the training set seizures.

**ARHMM** In Figure 3.18, we see that SGRLD converges much more rapidly than the other methods for the ARHMM. As each iteration of the Gibbs sampler takes 6 hours, it takes a couple weeks for the Gibbs sampler to converge to the solution SGRLD converges to in a few hours. Although the subset Gibbs sampler is 10x faster than Gibbs, it does not converge to the full data posterior and its generalization error to the heldout test set is poorer than

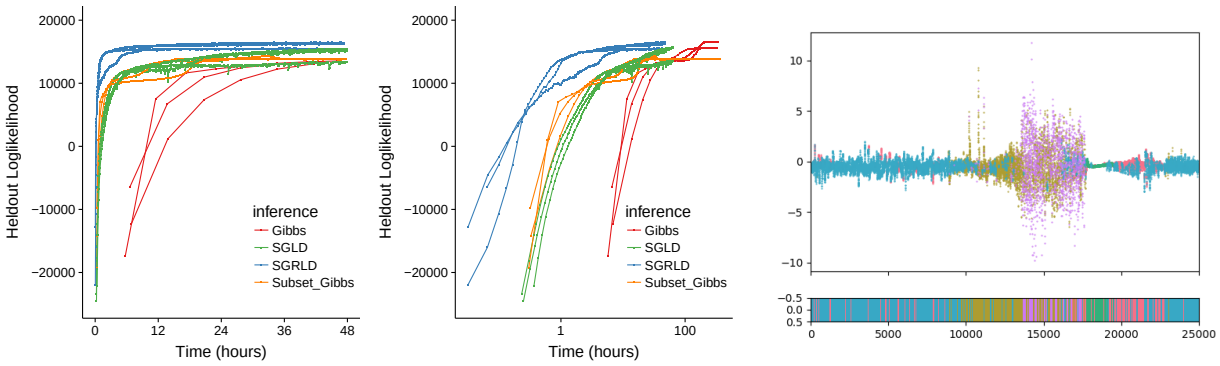


Figure 3.18: ARHMM for Canine Seizure Data: (left) heldout loglikelihood vs time, (center) heldout loglikelihood vs time on log-scale (right) example segmentation of a test seizure channel by SLDS fit with SGRLD. The MCMC methods compared are Gibbs, Subset Gibbs, SGLD, and SGLRD.

the other methods. From this experiment we see that SGMCMC methods provide order of magnitude improvements (compared to subsetting the data).

**SLDS** In Figure 3.19, we see again that the SGRLD sampler converges much more rapidly than the other methods for the SLDS. Furthermore, Gibbs sampling takes even longer than in ARHMM due to spending lots of time in local modes induced by the additional continuous latent state sequence. In comparison to Figure 3.18, we also see that the SLDS is a better model for this data than the ARHMM (as measured by heldout likelihood). Qualitatively, the SLDS segmentations of seizures (Figure 3.19 (right)) is more contiguous than the ARHMM segmentation (Figure 3.18 (right)).

*Cities Weather Data*

We apply SGMCMC to historical city weather data from Kaggle (Beniaguev, 2017). The data consists of hourly temperature, pressure and humidity measurements ($m = 3$) for 20

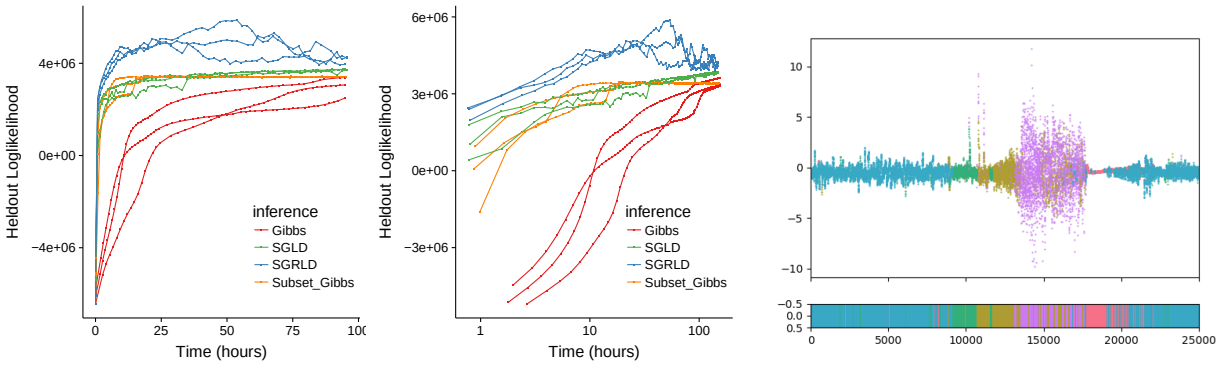


Figure 3.19: SLDS Canine Seizure Data: (left) heldout loglikelihood vs time, (center) heldout loglikelihood vs time on log-scale (right) example segmentation by ARHMM fit with SGRLD. The MCMC methods compared are Gibbs, Subset Gibbs, SGLD, and SGLRD.

US cities over 5 years with $T = 44,000$ hourly observations per city. We fit SLDS models with $n = 3$ and $K = 4$ to both the hourly and daily average observations, treating the cities independently. For both sets of observations, we perform an 80-20 train-test split over 20 cities, running inference on the training set (16 cities) and evaluating loglikelihood on the test set (4 cities).

Figure 3.20 (top-left) shows the heldout loglikelihood vs the runtime for the different samplers on the daily data. From this plot, we see that SGRLD clearly outperforms Gibbs. Although Gibbs converges quickly on the daily data, it gets stuck in local optima. In particular, the Gibbs runs converge to a suboptimal parametrization that mixes over three states, while SGRLD converges to a two state (summer-winter) solution (with the remaining states for sudden shifts or jumps). For example, Figure 3.20 (top-center and right) are fits of the daily model to the Houston time series for both Gibbs and SGRLD respectively. Figure 3.20 (bottom-left) shows the heldout loglikelihood vs the runtime of the different samplers for the hourly data. SGRLD again outperforms Gibbs and, for the hourly data, the Gibbs sampler is significantly slower than the SGMCMC samplers.

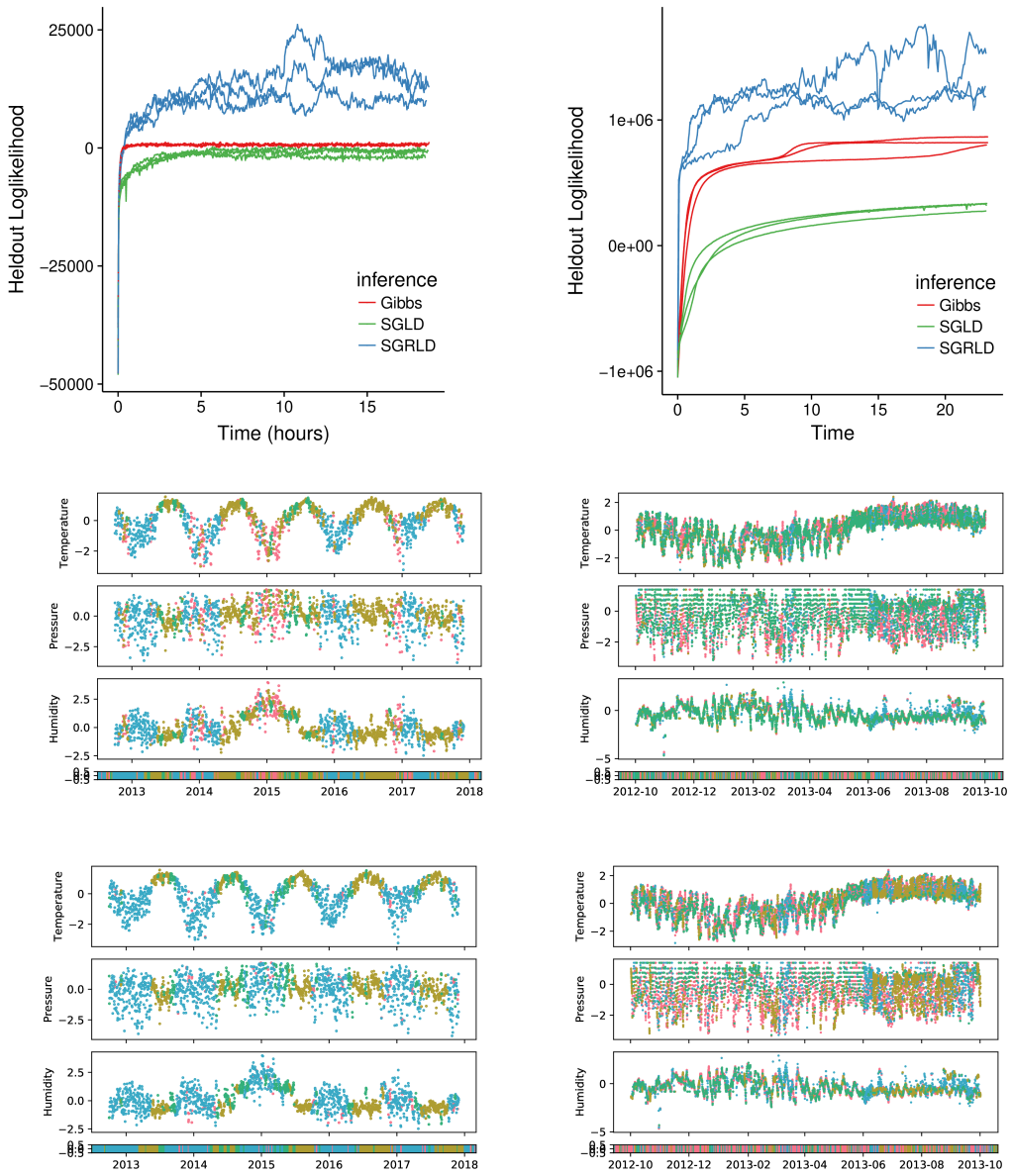


Figure 3.20: SLDS Weather Data. (Left) daily aggregated data, (Right) hourly data. (Top) heldout loglikelihood vs runtime, (center) Gibbs Houston fit, (bottom) SGRLD Houston fit. The MCMC methods compared are Gibbs, SGLD, and SGLRD.

*EUR-USD Exchange Rate*

We now consider fitting the SVM and the GARCH model to EUR-USD exchange rate data at the minute resolution from November 2017 to October 2018. The data consists of 350,000 observations of demeaned log-returns. As the market is closed during non-business hours, we further break the data into 53 weekly segments of roughly 7,000 observations each. In our model, we assume independence between weekly segments and divide the data into a training set of the first 45 weeks and a test set of the last 8 weeks. Full processing details and example plots are in the Appendix. Note that our method easily scales to the unsegmented series; however the abrupt changes between starts of weeks are not adequately modeled by Eq. (3.85)

We fit both the SVM and the GARCH model using SGLD with four different gradient methods: (i) **Full**, the full gradient over all segments in the training set; (ii) **Weekly**, a stochastic gradient over a randomly selected segment in the training set; (iii) **No Buffer**, a stochastic gradient over a randomly selected *subsequence* of length $S = 40$; and (iv) **Buffer**, our buffered stochastic gradient for a subsequence of length $S = 40$ with buffer length $B = 10$. To estimate the stochastic gradients, we use Algorithm 2 with $N = 1000$. To select the stepsize parameter, we performed a grid search over $\epsilon \in \{1, 0.1, 0.01, 0.001\}$ and selected the method with smallest KSD. We present the KSD results in Table 3.5. Figure 3.21 are trace plots of the heldout and predictive loglikelihood for the four different SGLD methods, each averaged over 5 chains.

For the SVM, we see that buffering improves performance on both heldout and predictive loglikelihoods, Figure 3.21 (top), and also leads to more accurate MCMC samples, Table 3.5 (left). In particular, the samples from SGLD without buffering have smaller $\phi, \tau^2$ and a larger $\sigma^2$, indicating that its posterior is (inaccurately) centered around a SVM with larger latent state noise. We also again see that the full sequence and weekly segment methods perform poorly due to the limited number of samples that can be computed in a fixed runtime.

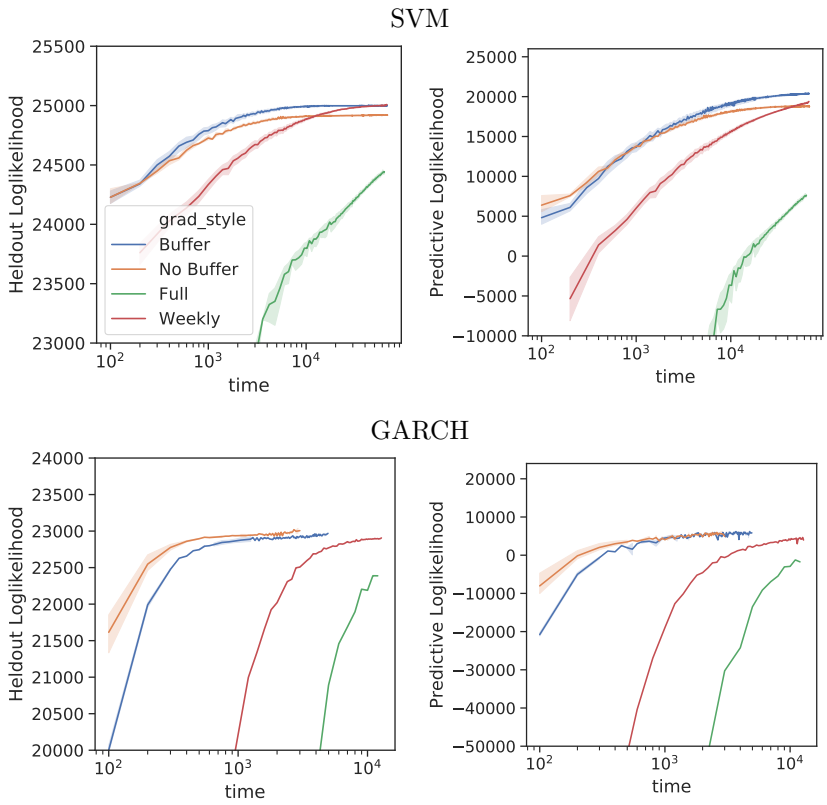


Figure 3.21: Comparison of SGLD with different gradient estimates on the exchange rate data: heldout-loglikelihood (left), 3-step ahead predictive loglikelihood (right) for SVM (top), GARCH (bottom).

For the GARCH model, Figure 3.21 (bottom) and Table 3.5 (right), we see that the subsequence methods out perform the full sequence methods, but unlike in the SVM, buffering does not help with inference on the GARCH data. This is because the GARCH model that we recover on the exchange rate data (for all gradient methods) is close to white noise $\beta \approx 0$. Therefore the model believes the observations are close to independent, hence no buffer is necessary. Although buffering performs worse on a runtime scale, here, it is leading to a more accurate posterior estimate (less bias) in *all* settings.

Table 3.5: KSD for SGLD on exchange rate data. Mean and SD over 5 chains each.

| | $\log_{10}$KSD | |
|---|---|---|
| Method | SVM | GARCH |
| Full | 4.03 (0.14) | 2.84 (0.30) |
| Weekly | 3.87 (0.08) | 2.81 (0.21) |
| No Buffer | 4.48 (0.01) | **2.09 (0.09)** |
| Buffer | **3.56 (0.08)** | **2.19 (0.05)** |

### *3.6 Conclusions*

In this Chapter, we developed stochastic gradient MCMC samplers for state space models of sequential data. We showed that naive stochastic gradients based on subsequences are *biased* and that this bias affects the finite sample average convergence bounds of SGMCMC (Theorem 1). To reduce bias, we proposed buffered stochastic gradients $\hat{g}_{S,B}$, Eq. (3.5), for linear SSMs and particle buffered stochastic gradients $g^{\text{PF}}_{S,B,N}$, Eq. (3.13), for nonlinear SSMs. We showed that the bias for these *buffered* stochastic gradients decay geometrically in the buffer size (Theorem 3 and 4) under mild conditions (Theorem 5). Using these estimators and error bounds, we developed SGMCMC samplers for discrete (Gaussian HMM, ARHMM), continuous (LGSSM), nonlinear (SVM, GARCH), and mixed-type (SLDS) state space models. In our experiments –including both synthetic and a variety of real-world datasets – we found that our methods can control bias with a modest buffer size and provide orders of magnitude run-time speed ups compared to batch MCMC.

There are many interesting directions for future work. This buffered gradient estimator for sequential data could be applied to other stochastic gradient methods such as maximum likelihood estimation or variational inference (Archer et al., 2015; Krishnan et al., 2017). The buffered gradient estimator could likewise be applied to diffusions with control vari-

ates (Baker et al., 2017; Chatterji et al., 2018) or with augmented dynamics, such as using momentum (SGHMC) (Chen et al., 2014) or temperature (SGNHT) (Ding et al., 2014). In terms of analysis, the standard SGLD error analysis could be extended to analyze the optimal trade-off between buffer size and subsequence length and relax the log-concave restriction of Theorem 5.

# Chapter 4

# ADAPTIVELY TRUNCATING BACKPROPAGATION THROUGH TIME

## *4.1 Introduction*

Recurrent neural networks (RNNs) are a popular method of processing sequential data for wide range of tasks such as language modeling, machine translation and reinforcement learning.

Recall from Chapter 2.3, RNNs model complex dynamics using a hidden state sequence $h_{1:T}$ that is updated by a recursive map $h_t = H(h_{t-1}, x_t; \theta)$, where $\theta$ is typically trained with gradient descent. These gradients can be calculated efficiently using *backpropagation through time* (BPTT) which applies backpropagation to the unrolled network (Werbos et al., 1990). For a random index $s$, the BPTT stochastic gradient estimate is

$$\hat{g} := \nabla_\theta \mathcal{L}_s = \frac{\partial \mathcal{L}_s}{\partial \theta} + \sum_{k=0}^{s} \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \cdot \frac{\partial h_{s-k}}{\partial \theta}, \tag{2.35}$$

For long sequential data, BPTT is both computationally and memory intensive, hence approximations based on *truncating* BPTT (TBPTT) have been proposed (Williams and Zipser, 1995; Sutskever, 2013)

$$\hat{g}_K := \widehat{\nabla}_\theta \mathcal{L}_s^K = \frac{\partial \mathcal{L}_s}{\partial \theta} + \sum_{k=0}^{K} \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \frac{\partial h_{s-k}}{\partial \theta}, \tag{2.36}$$

where $K \ll T$. For more details, see Chapter 2.3.3.

However, this truncation causes the gradients to be biased. When the truncation level is not sufficiently large, the bias introduced can cause SGD to not converge. In practice, a large truncation size is chosen heuristically (e.g. larger than the expected 'memory' of the system) or via cross-validation.

Quantifying the bias due to truncation is difficult. Depending on the parameters of the RNN, the gradient bounds for backpropagation either *explode* or *vanish* (Bengio et al., 1994; Pascanu et al., 2013). When the gradient bounds vanish, the bias in TBPTT can be bounded. Recent work has analyzed conditions for the parameters of the RNN to enforce this vanishing gradient condition (Miller and Hardt, 2019). However, these approaches are very restrictive and prevent the RNN from learning long-term dependencies.

To bound the bias in TBPTT, instead of restricting the parameters, we formalize the heuristic assumption that the gradients in backpropagation should rapidly decay for steps beyond the 'memory' of the RNN. Specifically, we assume gradient bounds that decay exponentially *in expectation* rather than uniformly. Under this assumption, we show that the bias in TBPTT decays geometrically and also how to estimate an upper bound for this bias given a minibatch of backpropagated gradients. Using this estimated upper bound, we propose an adaptive truncation scheme to control the bias. In addition, we prove non-asymptotic convergence rates for SGD when the *relative* bias of our gradients is bounded. In particular, we show that when the relative bias, $\delta < 1$, SGD with biased gradients converges at the rate $(1 - \delta)^{-1}$ compared to SGD with exact (unbiased) gradients. In our experiments, we see that (i) our heuristic assumption holds empirically for these tasks, (ii) our adaptive TBPTT method controls the bias, while fixed TBPTT does not, and (iii) that our adaptive TBPTT method is competitive with or outperforms the optimal fixed TBPTT.

The Chapter is organized as follows. First, we review previous work bounding the bias of TBPTT in Chapter 4.1.1. Then, we develop our theoretical results in Chapter 4.2. Using this theory, we develop estimators for the bias and propose an adaptive TBPTT SGD scheme in Chapter 4.3. Finally, we test our proposed adaptive TBPTT training scheme on both synthetic data, language modeling data, and temporal point process data in Chapter 4.4.

### *4.1.1 Previous Work*

To analyze the bias of the TBPTT stochastic gradient estimate $\hat{g}_K$, previous work has focused on the behavior of $\frac{\partial \mathcal{L}_t}{\partial h_{t-k}}$ for large $k$ [1]. Pascanu et al. (2013) observed

$$\frac{\partial \mathcal{L}_t}{\partial h_{t-k}} = \frac{\partial \mathcal{L}_t}{\partial h_t} \prod_{r=1}^{k} \frac{\partial h_{t-r+1}}{\partial h_{t-r}} \ . \tag{4.1}$$

In particular, the repeated product of Jacobian matrices $\frac{\partial h_t}{\partial h_{t-1}}$ cause the gradient to tend to *explode* to infinity or *vanish* to zero. When $\theta$ has an exploding gradient, then the bias of $\hat{g}_K$ is unbounded. When $\theta$ has a vanishing gradient, then the bias of $\hat{g}_K$ is small; however if the gradient decays too rapidly, the RNN cannot learn long-term dependences (Bengio et al., 1994; Pascanu et al., 2013; Miller and Hardt, 2019). In practice, LSTMs and other gated-RNNs find a middle ground where (for appropriate $\theta$ and inputs $x_{1:T}$) the gate variables prevent the gradient from exploding or vanishing (Hochreiter and Schmidhuber, 1997; Belletti et al., 2018). However, *gradient bounds*, based on the Jacobian $\|\frac{\partial h_t}{\partial h_{t-1}}\| \leq \lambda$, either explode or vanish

$$\left\| \frac{\partial \mathcal{L}_t}{\partial h_{t-k}} \right\| \leq \left\| \frac{\partial \mathcal{L}_t}{\partial h_t} \right\| \cdot \lambda^k \ . \tag{4.2}$$

In light of Eq. (4.2), several approaches have been proposed in the literature to restrict $\theta$ to control $\lambda$. *Unitary* training methods have been proposed to restrict $\theta$ such that $\lambda \approx 1$ for all $\theta$, but do not bound the bias of the resulting gradient (Arjovsky et al., 2016; Jing et al., 2017; Vorontsov et al., 2017). *Stable* or *Chaos-Free* training methods have been proposed to restrict $\theta$ such that $\lambda < 1$ (Laurent and von Brecht, 2017; Miller and Hardt, 2019). In particular, Miller and Hardt (2019) call an RNN $H$ *stable* for parameters $\theta$ if it is a contraction in $h$, that is

$$\sup_{\substack{h,h' \in \mathbb{R}^{d_h} \\ x \in \mathbb{R}^{d_x}}} \frac{\|H(h,x,\theta) - H(h',x,\theta)\|}{\|h - h'\|} \leq \lambda < 1 \tag{4.3}$$

[1] For the bias of $\hat{g}_K$, we should focus on $\frac{\partial \mathcal{L}_t}{\partial h_{t-k}} \frac{\partial h_{t-k}}{\theta}$; however it typically assumed that $\frac{\partial h_t}{\theta}$ is bounded for all $t$.

and call an RNN $H$ *data-dependent stable* if the supremum over Eq. (4.3) is restricted to *observed* inputs $x \in X$. Let $\Theta_{\lambda\text{-Stable}}$ be the set of parameters $\theta$ satisfying Eq. (4.3) and $\Theta^{X}_{\lambda\text{-Stable}}$ be the set of parameters $\theta$ satisfying the data-dependent version.

Miller and Hardt (2019) show that if $\theta \in \Theta_{\lambda\text{-Stable}}$ the RNN gradients have an exponential forgetting property (as $\|\frac{\partial H}{\partial h}\| < \lambda$), which prevents the RNN from learning long-term dependences. We desire conditions on $\theta$ where we can bound the bias, but are less restrictive than Eq. (4.3).

## *4.2 Theory for TBPTT Bias*

In this section, we consider bounding the bias in TBPTT when $\theta$ satisfies a *relaxation* of the contraction restriction Eq. (4.3). Under this condition and a bound on $\|\partial h_t/\partial\theta\|$, we show that both the *absolute bias* and *relative bias* are bounded and decay geometrically for large $K$ (Theorem 7). Finally, we prove the convergence rate of SGD for gradients with bounded relative bias (Theorem 8). Full proofs of theorems can be found in Appendix **??**.

### *4.2.1 Geometric Decay for Large Lags*

To reduce notation, let $\phi_k = \|\frac{\partial L_s}{\partial h_{s-k}}\|$ denote the gradient norm of loss $\mathcal{L}_s$ at time $s$ with respect to the hidden state $k$ lags in the past. We emphasize that $\phi_k$ is a random variable as $s$ is the random index for the stochastic gradient of Eqs. (2.35) and (2.36).

Our relaxation of Eq. (4.3) is to assume the norm of the backpropagated gradient $\phi_k$ decays geometrically, *on average* for large enough lags $k$. More formally,

**Assumption (A-1).** *For $\theta$ fixed, there exists $\beta \in (0,1)$ and $\tau \geq 0$ such that*

$$\mathbb{E}\left[\phi_{k+1}\right] \leq \beta \cdot \mathbb{E}\left[\phi_k\right] \ , \ \textit{for all } k \geq \tau \tag{4.4}$$

This generalizes the vanishing gradient condition to hold *in expectation*.

To contrast (A-1) with $\theta \in \Theta_{\lambda\text{-Stable}}$, we observe that if $\theta \in \Theta_{\lambda\text{-Stable}}$ then the gradient norms $\phi_k$ must *uniformly* decay exponentially

$$\phi_{k+1} \leq \lambda \cdot \phi_k \text{ for all } k \ . \tag{4.5}$$

Eq. (4.4) is less restrictive than Eq. (4.5) as $\phi_{k+1} \leq \beta \cdot \phi_k$ only occurs for $k > \tau$ and in expectation rather than uniformly. Denote the set of $\theta$ that satisfy (A-1) with $\beta, \tau$ for inputs $X = x_{1:T}$ as $\Theta^X_{\beta,\tau}$. Then, we have $\Theta_{\lambda\text{-Stable}} \subseteq \Theta^X_{\lambda\text{-Stable}} \subset \Theta^X_{\beta,\tau}$ for $(\beta, \tau) = (\lambda, 0)$. Therefore (A-1) is a more general condition.

For illustration, we present two examples where $\theta \in \Theta^X_{\beta,\tau}$ but $\theta \notin \Theta^X_{\lambda\text{-Stable}}$.

**Nilpotent Linear RNN:** Consider a simple RNN with linear activation $h_t = Wh_{t-1} + Ux_t$ where $W$ is a *nilpotent* matrix with index $k$, that is $W^k = 0$. Then $\partial h_t / \partial h_{t-k} = W^k = 0$ hence $(W, U) \in \Theta^X_{0,k}$; however the norm $\|\partial h_t / \partial h_{t-k}\| = \|W\|$ can be arbitrarily large, thus $(W, U) \notin \Theta_{\lambda\text{-Stable}}$. For this example, although $H$ is not stable, $k$-repeated composition $h_t = (H \circ \cdots \circ H)(h_{t-k}, x_{t-k+1:t})$ is stable.

**Unstable RNN with Resetting:** Consider a generic RNN with $\theta$ chosen such that $H$ is unstable, Lipschitz with constant $\lambda > 1$, but with a *resetting* property $H(h, x) = 0$, whenever $x \in \mathcal{X}_0$. Then,

$$\mathbb{E}_S[\phi_{k+1}] \leq \Pr(x_s, \ldots, x_{s-k+1} \notin \mathcal{X}_0) \cdot \lambda \cdot \mathbb{E}_S[\phi_k]$$

Although the RNN is unstable, if the probability $\{x_s, \ldots, x_{s-k+1}\}$ *hits* the resetting set $\mathcal{X}_0$ is greater than $1 - \beta\lambda^{-1}$ for sufficiently large $k$ and for some $\beta < 1$, then $\Pr(x_s, \ldots, x_{s-k+1} \notin \mathcal{X}_0) \cdot \lambda \leq \beta$ and therefore $\theta \in \Theta_{\beta,k}$ with $\theta \notin \Theta^X_{\lambda\text{-Stable}}$. For this example, although $H$ is not stable, properties of the input distribution of $x_t$ can lead to $H$ have vanishing gradients in *expectation.*

### *4.2.2 TBPTT Bias Bounds*

We now show the usefulness of assumption (A-1), that is if $\theta \in \Theta^X_{\beta,\tau}$, then the bias of TBPTT is bounded and decays geometrically in $K$. To do so, we additionally assume the partial derivatives of the hidden state with respect to the parameters is bounded.

**Assumption (A-2).** *For $\theta$ fixed, there exists $M < \infty$ such that $\|\frac{\partial H(x_t, h_t, \theta)}{\partial \theta}\| \leq M$ for all $t$.*

For most typical RNNs, where $\theta$ are weights and biases, if the inputs $x_t$ and $h_t$ are bounded then $M$ can be bounded and assumption (A-2) holds.

We now show both the bias of TBPTT is guaranteed to decay geometrically for large $K$.

**Theorem 7** (Bias Bound)**.** *If (A-1) and (A-2) hold for $\theta$, then the absolute bias is upper bounded as*

$$\|\mathbb{E}\left[\hat{g}_K(\theta)\right] - g(\theta)\| \leq \mathcal{E}(K,\theta) \ ,$$

*where*

$$\mathcal{E}(K,\theta) = \begin{cases} M \cdot \mathbb{E}\left[\sum_{k=K+1}^{\tau-1} \phi_k + \frac{\phi_\tau}{1-\beta}\right], & K < \tau \\ M \cdot \mathbb{E}\left[\phi_\tau\right] \cdot \frac{\beta^{K-\tau}}{1-\beta}, & K \geq \tau \end{cases} .$$

*And the relative bias is upper bounded by*

$$\frac{\|\mathbb{E}\left[\hat{g}_K(\theta)\right] - g(\theta)\|}{\|g(\theta)\|} \leq \Delta(K,\theta) \ ,$$

*where*

$$\Delta(K,\theta) = \frac{\mathcal{E}(K,\theta)}{\max_{k \leq K} \|\mathbb{E}\,\hat{g}_k(\theta)\| - \mathcal{E}(k,\theta)} \ .$$

*when the denominator is positive.*

Note that $\mathcal{E}(K,\theta)$ decays geometrically for $K \geq \tau$ and therefore $\Delta(K,\theta)$ decays geometrically for large enough $K$ (as the denominator is monotone increasing).

Using this upper bound, we define $\kappa(\delta,\theta)$ to be the *smallest truncation length* for the parameters $\theta$ with guaranteed relative bias less than $\delta$. That is

$$\kappa(\delta,\theta) = \min_K \{\Delta(K,\theta) < \delta\} \ . \tag{4.6}$$

The geometric decay in $\Delta(K,\theta)$ ensures $\kappa(\delta,\theta)$ is small.

Finally, we define the *adaptive* TBPTT gradient estimator to be $\hat{g}(\theta) = \hat{g}_{\kappa(\delta,\theta)}(\theta)$, which adaptively truncates BPTT after $\kappa(\delta,\theta)$ steps –rather than using a fixed truncation length– and therefore has relative bias less than $\delta$ whenever Theorem 7 is satisfied.

### *4.2.3 SGD with Biased Gradients*

We use stochastic gradient descent (SGD) to learn $\theta$

$$\theta_{n+1} = \theta_n - \gamma_n \cdot \hat{g}(\theta_n) \ , \tag{4.7}$$

where $\{\gamma_n\}_{n=1}^N$ are stepsizes. When using SGD for non-convex optimization, such as in training RNNs, we are interested in convergence to stationary points of $\mathcal{L}$, where $\theta$ is called a $\epsilon$-stationary point of $\mathcal{L}(\theta)$ if $\|g(\theta)\|^2 \leq \epsilon$ (Nesterov, 2013).

Usually the stochastic gradients $\hat{g}$ are assumed to be unbiased; however when training RNNs with TBPTT, the truncated gradients are biased. To leverage Chapter 4.2.2, we consider the case when $\hat{g}$ has a bounded relative bias,

$$\|\mathbb{E}\left[\hat{g}(\theta)\right] - g(\theta)\| \leq \delta \|g(\theta)\|, \ \ \forall \theta \ . \tag{4.8}$$

such as for our adaptive estimator $\hat{g}_{\kappa(\delta,\theta)}(\theta)$.

For stochastic gradients with bounded relative bias $\delta < 1$ (and the additional assumptions below), Poljak and Tsypkin (1973) and Bertsekas and Tsitsiklis (1989) prove that the averaged SGD sequence $\theta_n$ *asymptotically* converges to a stationary point when the stepsizes are $\gamma_n \propto n^{-1}$. However, *non-asymptotic* convergence rates are also of interest, as they are useful in practice to understand the non-asymptotic performance of the algorithm. Ghadimi and Lan (2013) prove non-asymptotic convergence rates for SGD with unbiased gradients. We extend these results to the case of SGD with biased gradients. Similar results were previously investigated by Chen and Luss (2018), but with weaker bounds[2].

For our SGD convergence bound we need two additional assumptions.

**Assumption (A-3).** *The gradients are L-Lipschitz*

$$\|g(\theta) - g(\theta')\| \leq L\|\theta - \theta'\|, \quad \forall \theta, \theta' \ .$$

[2]They consider the case of consistent but biased estimators, where the gradients are uniformly bounded and the relative error is controlled with high probability (rather than in expectation). See the Supplement for additional discussion.

This assumption holds for generic RNNs as long as the activation functions are smooth (e.g. $\sigma$ or tanh, but not RELU). Second, we assume that the variance of our stochastic gradient estimator is uniformly bounded.

**Assumption (A-4).** $\mathbb{E}\,\|\hat{g}(\theta) - \mathbb{E}\,\hat{g}(\theta)\|^2 \leq \sigma^2$ *for all* $\theta$.

We can now present our main theorem regarding convergence rates of SGD with biased gradients.

**Theorem 8** (SGD with Biased Gradients)**.** *If the relative bias of* $\hat{g}(\theta)$ *is bounded by* $\delta < 1$ *for all* $\theta_n$ *and (A-3) and (A-4) both hold, then SGD, Eq.* (4.7)*, with stepsizes* $\gamma_n \leq \frac{1-\delta}{L(1+\delta)^2}$ *satisfies*

$$\min_{n=1,\ldots,N+1} \|g(\theta_n)\|^2 \leq \frac{2D_{\mathcal{L}} + L\sigma^2 \sum_{n=1}^{N} \gamma_n^2}{(1-\delta)\sum_{n=1}^{N} \gamma_n} \ , \tag{4.9}$$

*where* $D_{\mathcal{L}} = (\mathcal{L}(\theta_1) - \min_{\theta^*} \mathcal{L}(\theta^*))$.

*In particular, the optimal fixed stepsize for* $N$ *fixed is* $\gamma_n = \sqrt{2D_{\mathcal{L}}/(NL\sigma^2)}$*, for which the bound is*

$$\min_{n=1,\ldots,N+1} \|g(\theta_n)\|^2 \leq \frac{1}{1-\delta} \cdot \sqrt{\frac{8D_{\mathcal{L}} L\sigma^2}{N}} \ . \tag{4.10}$$

When $\delta = 0$, Theorem 8 reduces to the smooth non-convex convergence rate bound (Ghadimi and Lan, 2013). The price of biased gradients in SGD is the factor $(1-\delta)^{-1}$.

In practice, when the constants in Theorem 8 are unknown (e.g. $D_{\mathcal{L}}$), we can use a decaying stepsize $\gamma_n = \gamma \cdot n^{-1/2}$. Once $\gamma_n \leq \frac{1-\delta}{L(1+\delta)^2}$, Theorem 8 implies

$$\min_{n=1,\ldots,N+1} \|g(\theta_n)\|^2 = \mathcal{O}\left(\frac{1}{1-\delta} \cdot \frac{\log n}{\sqrt{n}}\right) \ . \tag{4.11}$$

Theorem 8 provides bounds of the form $\min_n \|g(\theta_n)\|^2 < \epsilon$, but does not say which iterate is best. This can be accounted for by using a random-stopping time (Ghadimi and Lan, 2013) or using variance reduction methods such as SVRG (Reddi et al., 2016). We leave these extensions as future work. In our experiments, we select $\theta_n$ by evaluating performance on a validation set.

**What happens when (A-1) is violated?** In our assumptions, we do not restrict $\theta$ to $\Theta^X_{\beta,\tau}$, therefore whenever $\theta_n \notin \Theta_{\beta,\tau}$ (or in practice when $\kappa(\delta, \theta_n)$ is larger than our computational budget allows), we are not able to construct $\hat{g}(\theta_n)$ such that the relative bias is bounded. In these cases, we recommending using $\hat{g} = \hat{g}_{K_{\max}}$ for some large truncation $K_{\max}$. Although $\hat{g}_{K_{\max}}$ does not satisfy (A-1), we assume the stationary points of interest $\theta^*$ are in $\Theta^X_{\beta,\tau}$ and that during training eventually $\theta_n$ reaches a neighborhood of a $\theta^*$ that is a subset of $\Theta_{\beta,\tau}$ where our theory holds.

The advantage of an adaptive $\kappa(\delta, \theta)$ over a fixed $K$ is that it ensures convergence when possible (relative bias $\delta < 1$), while being able to get away with using a smaller $K$ during optimization for computational speed-ups.

## *4.3 Adaptive TBPTT Algorithm*

The theory in Chapter 4.2 naturally suggests an adaptive TBPTT algorithm to control relative bias.

Our method selects a truncation level by periodically estimating $\kappa(\delta, \theta)$ over the course of SGD. To estimate $\kappa(\delta, \theta)$, we first estimate the $\beta, \tau$ for Assumption (A-1) using regression (Chapter 4.3.2) and then use the relative bound derived in Theorem to estimate $\kappa$ (Chapter 4.3.3) We summarize our adaptive TBPTT scheme in Algorithm 6 [3].

### *4.3.1 Computing TBPTT Gradients*

Recall the equation for BPTT($K_1, K_2$) given by Eq. (2.37) in Chapter 2.3.3. To ensure all $K$ losses are backpropagated at least $K$ steps, in our experiments we use BPTT($2K, K$). We also scale the gradient updates $\gamma_n \hat{g}$ by $\sqrt{K_n}$ to account for the decreasing variance in BPTT($2K, K$) as $K$ increases. If we did not scale the gradient updates, then as $K$ increases, the resulting increase in the computational cost per step is not offset.

To handle the initialization of $h_{s-k}$ in Eq. (2.37), we partition $\{1, \ldots, T\}$ into $T/K$

[3] Code for our algorithm and experiments is at `https://github.com/aicherc/adaptive_tbptt`.

**Algorithm 6** Adaptive TBPTT

**Input:** initial parameters $\theta_0$, initial truncation $K_0$, stepsizes $\gamma_{1:N}$, relative bias tolerance $\delta$, batch size $S$, window size $R$
**for** $n = 0, \ldots, N-1$ **do**
  ▷ Compute adaptive truncation
  Sample random minibatch $\mathcal{S}$ of size $S$
  Calculate $\phi_k$ using BPTT$(R, 1)$ // Eq. (4.13)
  Calculate $\hat{P}_{\mathcal{S}}[\phi_k]$ for $k \in [0, R]$ // Eq. (4.15)
  Estimate $\hat{\beta}$ using regression // Eq. (4.16) or (4.17)
  Set $K_n = \hat{\kappa}(\delta, \theta)$ // Eq. (4.20)
  ▷ Update $\theta$ with streaming gradients
  **for** $m = 1, \ldots, T/K_n$ **do**
    Get minibatch $\mathcal{S}_m$ // Eq. (4.12)
    Calculate $\hat{g}(\theta_n) = \text{BPTT}(2K_n, K_n)$ on $\mathcal{S}_m$ // Eq. (2.37)
    Set $\theta_n = \theta_n - \gamma_n \cdot \sqrt{K_n} \cdot \hat{g}(\theta_n)$
  **end for**
  Set $\theta_{n+1} = \theta_n$
**end for**
Return $\theta_{1:N}$.

contiguous subsequences

$$\mathcal{S}_m = [(m-1) * K + 1, \ldots, m * K] \ . \tag{4.12}$$

By sequentially processing $\mathcal{S}_m$ in order, the final hidden state of the RNN on $\mathcal{S}_m$ can be used as the input for the RNN on $\mathcal{S}_{m+1}$.

### *4.3.2 Estimating the Geometric Decay Rate*

A prerequisite for determining our adaptive truncation level is estimating the geometric rate of decay in Eq. (4.4), $\beta$, and the lag at which it is valid, $\tau$. We consider the case where we are given a batch of gradient norms $\phi_k$ for $s$ in a random minibatch $\mathcal{S}$ of size $|\mathcal{S}| = S$ that are backpropagated over a window $k \in [0, R]$

$$\left\{ \phi_k = \left\| \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \right\| \; : \; s \in \mathcal{S}, k \in [0, R] \right\} \quad . \tag{4.13}$$

The gradient norms $\phi_k$ can be computed iteratively in parallel using the same architecture as truncated backpropagation BPTT$(R, 1)$. The window size $R$ should be set to some large value. This should be larger than the $\tau$ of the optimal $\theta^*$. The window size $R$ can be large, since we only estimate $\beta$ periodically.

We first focus on estimating $\beta$ given an estimate $\hat{\tau} > \tau$. We observe that if $\hat{\tau} \geq \tau$ and (A-1) holds, then

$$\log \beta \geq \max_{k' > k \geq \hat{\tau}} \frac{\log \mathbb{E}\left[\phi_k\right] - \log \mathbb{E}\left[\phi_{k'}\right]}{k - k'} \quad . \tag{4.14}$$

Eq. (4.14) states that $\log \beta$ bounds the slope of $\log \mathbb{E}\left[\phi_t\right]$ between any pair of points larger than $\tau$.

Using Eq. (4.14), we propose two methods for estimating $\beta$. We replace the expectation $\mathbb{E}$ with the empirical approximation based on the minibatch $\mathcal{S}$

$$\mathbb{E}\left[f(s)\right] \approx \hat{P}_{\mathcal{S}}[f(s)] := \frac{1}{\|\mathcal{S}\|} \sum_{s \in \mathcal{S}} f(s). \tag{4.15}$$

Substituting the empirical approximation into Eq. (4.14) and restricting the points to $[\hat{\tau}, R]$, we obtain

$$\hat{\beta} = \log \left[ \max_{\hat{\tau} \leq k < k' \leq R} \frac{\log \hat{P}_{\mathcal{S}}[\phi_k] - \log \hat{P}_{\mathcal{S}}[\phi_{k'}]}{k - k'} \right] \quad . \tag{4.16}$$

Because this estimate of $\beta$ is based on the maximum it is sensitive to noise: a single noisy pair of $\hat{P}_{\mathcal{S}}[\phi_k]$ completely determines $\hat{\beta}$ when using Eq. (4.16). To reduce this sensitivity, we could use a $(1 - \alpha)$ quantile instead of strict max; however, to account for the noise in

$\hat{P}_{\mathcal{S}}[\phi_k]$, we use linear regression, which is a weighted-average of the pairs of slopes

$$\tilde{\beta} = \log\left[\frac{\sum_{k,k'}(\log \hat{P}_{\mathcal{S}}[\phi_k] - \log \hat{P}_{\mathcal{S}}[\phi_{k'}](k-k'))}{\sum_{k,k'}(k-k')^2}\right] \quad . \tag{4.17}$$

This estimator is not guaranteed to be consistent for an upper bound on $\beta$ (i.e. as $|\mathcal{S}| \to T$, $\tilde{\beta} \not\geq \beta$); however, we found Eq. (4.17) performed better in practice.

The correctness and efficiency of both methods depends on the size of both the minibatch $\mathcal{S}$ and the window $[\hat{\tau}, R]$. Larger minibatches improve the approximation accuracy of $\hat{P}_{\mathcal{S}}$. Large windows $[\hat{\tau}, R]$ are necessary to check (A-1), but also lead to additional noise.

In practice, we set $\hat{\tau}$ to be a fraction of $R$; in our experiments we did not see much variability in $\hat{\beta}$ once $\hat{\tau}$ was sufficiently large, therefore we use $\hat{\tau} = \frac{9}{10}R$.

### *4.3.3 Estimating the Truncation Level*

To estimate $\kappa(\delta, \theta)$ in Eq. (4.6), we obtain empirical estimates for the absolute and relative biases of the gradient.

Given $\hat{\beta}, \hat{\tau}$, our estimated bound for the absolute bias is

$$\hat{\mathcal{E}}(K, \theta) = \begin{cases} \hat{M} \cdot \hat{P}_{\mathcal{S}}\left[\sum_{k=K+1}^{\hat{\tau}-1} \phi_k + \frac{\phi_{S,\hat{\tau}}}{1-\hat{\beta}}\right], & K < \hat{\tau} \\ \hat{M} \cdot \hat{P}_{\mathcal{S}}[\phi_{\hat{\tau}}] \cdot \frac{\hat{\beta}^{K-\hat{\tau}}}{1-\hat{\beta}}, & K \geq \hat{\tau} \end{cases} \tag{4.18}$$

where we estimate an upper-bound $\hat{M}$ for $M$ by keeping track of an upper-bound for $h_{s,t}$ and $x_{s,t}$ during training.

Similarly, our estimated bound for the relative bias is

$$\hat{\Delta}(K, \theta) = \frac{\hat{\mathcal{E}}(K, \theta)}{\| \max_{k \leq K} \hat{P}_{\mathcal{S}}[\hat{g}_k(\theta)]\| - \hat{\mathcal{E}}(k, \theta)} \tag{4.19}$$

In our implementation, we make the simplifying assumption that $\|\hat{P}_{\mathcal{S}}[\hat{g}_K / \hat{M}]\| \approx \hat{P}_{\mathcal{S}}\| \sum_{k=0}^{K} \phi_k\|$, which allows us to avoid calculating $\hat{M}$.

Our estimate for $K$ with relative error $\delta$ is

$$\hat{\kappa}(\delta, \theta) = \min_{K \in [K_{\min}, K_{\max}]} \{\hat{\Delta}(K, \theta) < \delta\} \quad , \tag{4.20}$$

where $K_{\min}$ and $K_{\max}$ are user-specified bounds.

### *4.3.4 Runtime Analysis of Algorithm 6*

Our adaptive TBPTT scheme, Algorithm 6, consists of estimating the truncation length (lines 3-7) and updating the parameters with SGD using TBPTT (lines 8-12); whereas a fixed TBPTT scheme skips lines 3-7.

Updating the parameters with BPTT$(2K, K)$ streaming over $\{1, \ldots, T\}$ (lines 8-12), takes $\mathcal{O}(T/K \cdot K) = \mathcal{O}(T)$ time and memory[4]. The additional computational cost of adaptive TBPTT (lines 3-7) is dominated by the calculation of gradient norms $\phi_k$, Eq. (4.13), using BPTT$(R, 1)$, which takes $\mathcal{O}(R)$ time and memory. If we update the truncation length $\alpha$ times each epoch, then the total cost for adaptive TBPTT is $\mathcal{O}(T + \alpha R)$. Therefore, the additional computation cost is negligible when $\alpha R << T$.

In Algorithm 6, we only update $K$ once per epoch ($\alpha = 1$); however more frequent updates ($\alpha < 1$) allow for less stale estimates at additional computational cost.

## *4.4 Experiments*

In this section we demonstrate the advantages of our adaptive TBPTT scheme (Algorithm 6) in synthetic copying tasks, language modeling, and temporal point process estimation. For each task, we compare using fixed TBPTT and our adaptive TBPTT to train RNNs with SGD and we split the data into training, validation and test sets. We evaluate performance using perplexity (PPL) on the test set, for the $\theta_n$ that achieve the best PPL on the validation set. To make a fair comparison, we measure PPL against the number of data passes (epochs) used in training (counting the data used to estimate $\hat{\kappa}(\delta, \theta_n)$). We also evaluate the relative bias of our gradient estimates $\delta$ and truncation length $K$. For our experiments with SGD, we use a fixed learning rate chosen to be the largest power of 10 such that SGD did not quickly diverge. In Chapter 4.4.3, we demonstrate that the best $\theta_n$ appear to satisfy (A-1) (e.g., $\theta \in \Theta_{\beta,\lambda}$) by presenting the gradient norms $\mathbb{E}[\phi_k]$ against lag $k$.

---

[4]As $K$ increases, BPTT$(2K, K)$ takes more time per step $\mathcal{O}(K)$, but there are less steps per epoch $\mathcal{O}(T/K)$; hence the overall computation time is $\mathcal{O}(T)$

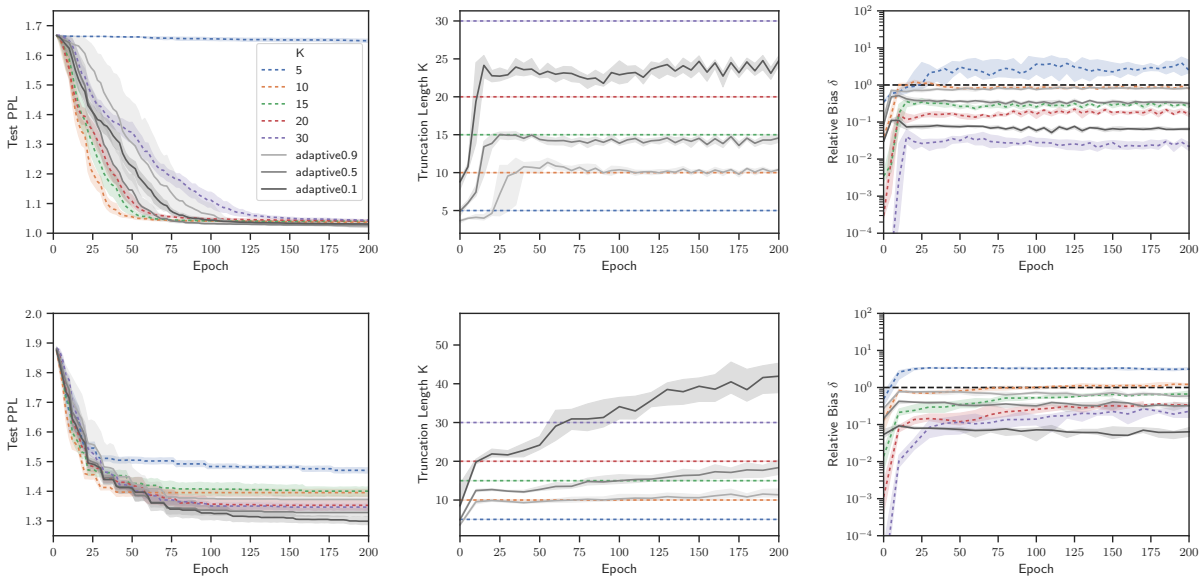


Figure 4.1: Synthetic Copy Results: (left) Test PPL vs epoch, (center) $\hat{\kappa}(\delta, \theta_n)$ vs epoch (right) $\hat{\delta}(K)$ vs epoch. (Top) fixed $m = 10$, (bottom) variable $m \in [5, 10]$. Error bars are 95 percentiles over 50 minibatch estimates. Solid dark lines are our adaptive TBPTT methods, dashed colored lines are fixed TBPTT baselines. We see that the adaptive method converges in fewer epochs (left), while maintaining a controlled relative bias $\delta(K) \leq \delta$ (right).

### *4.4.1 Synthetic Copy Tasks*

The 'copy' synthetic task is used to test the RNN's ability to remember information seen earlier (Hochreiter and Schmidhuber, 1997; Arjovsky et al., 2016; Jing et al., 2017; Vorontsov et al., 2017). We consider a special variant of this task from (Mujika et al., 2018), which we review now. Let $A = \{a_i\}$ be a set of symbols, where the first $I$ represent data and remaining two represent "blank" and "start recall". Each input consists of sequence of $m$ random data symbols followed by the "start recall" symbol and $m - 1$ more blanks. The desired output consists of $m$ blanks followed by the original sequence of $m$ data symbols. For example when $m = 6$ and $A = \{A, B, C, -, \#\}$

```
Input:   ACBBAB#-----
```

```
Output:  ------ACBBAB
```

We concatenate multiple of such inputs to construct $x_{1:T}$ and multiple outputs to construct $y_{1:T}$. We expect that TBPTT with $K > m$ will perform well, while $K < m$ will perform poorly.

In our experiments, we consider both a *fixed* $m = 10$ and a *variable* $m$ drawn uniformly over $[5, 10]$. For the variable copy length experiment, we expect TBPTT to degrade more gradually as $K$ decreases. We set $I = 6$ and use training data of length $T = 256,000$ and validation and test data of length $T = 64,000$.

**Model and Training Setup** We train separate 2-layer LSTMs with a embedding input layer and a linear output-layer to both the fixed- and variable- copy tasks by minimizing the cross-entropy loss. The embedding dimension is set to 6 and hidden and cell dimensions of the LSTM layers are set to 50. We train $\theta$ using SGD using a minibatch size of $S = 64$ and a fixed learning rate of $\gamma = 1.0$ with fixed TBPTT $K \in [5, 10, 15, 20, 30]$ and our adaptive TBPTT method $\delta \in [0.9, 0.5, 0.1]$, $W = 100$, $K_0 = 15$ and $[K_{\min}, K_{\max}] = [2, 100]$. We set $W = 100$, $K_0 = 15$ and $[K_{\min}, K_{\max}] = [2, 100]$ for Algorithm 6.

**Results** Figure 4.1 shows the results for the synthetic copy task. The left figures present the test set PPL against the number of data epochs used in training. We see that adaptive methods (black solid lines) perform as well as or better than the best fixed methods (colored dashed lines). In particular, TBPTT with $K = 5$ (blue) does not learn how to accurately predict the outputs as $K$ is too small $5 = K \leq m = 10$. On the other hand, $K = 30$ (purple) takes much longer to converge. The center figures show how $\hat{\kappa}(\delta, \theta_n)$ evolves for the adaptive TBPTT methods over training. The adaptive methods initially use small $K$ as the backpropagated gradient vanish rapidly in the early epochs; however as the adaptive TBPTT methods learn $\theta$ the necessary $K$ for a relative error of $\delta$ increases until they eventually level off at $\kappa(\delta, \theta_N)$. The right figures show the estimated relative bias $\delta$ of the gradient estimates during training. We see that the adaptive methods are able to roughly control $\delta$ to be

less than their target values, while the fixed methods initially start with low $\delta$ and before increasing and leveling off. Additional figures for the validation PPL and tables of numerical values can be found in the Supplement.

Table 4.1 is a table of the test PPL results evaluated at the 'best' validation PPL. This table provides the numeric values of the 'best' PPL values for Figures 4.1 and B.1. We see that the adaptive TBPTT perform as well as or outperform the best fixed $K$ TBPTT.

Table 4.1: Table of PPL for Synthetic Copy Experiments: (left) fixed $m = 10$, (right) variable $m \in [5, 10]$. 'Valid PPL' is the best validation set PPL. 'Test PPL' is the test set PPL at parameters of the best validation set PPL. Standard deviation over multiple initializations are in parentheses.

| Fixed Copy $m = 10$ | | | Variable Copy $m \in [5, 10]$ | | |
|---|---|---|---|---|---|
| K | Valid PPL | Test PPL | K | Valid PPL | Test PPL |
| 5 | 1.655 (0.012) | 1.646 (0.012) | 5 | 1.46 (0.01) | 1.47 (0.01) |
| 10 | 1.035 (0.007) | 1.036 (0.005) | 10 | 1.41 (0.02) | 1.39 (0.02) |
| 15 | 1.038 (0.005) | 1.039 (0.003) | 15 | 1.39 (0.03) | 1.37 (0.03) |
| 20 | 1.045 (0.009) | 1.040 (0.006) | 20 | 1.39 (0.03) | 1.35 (0.03) |
| 30 | 1.044 (0.007) | 1.043 (0.004) | 30 | **1.33 (0.02)** | **1.31 (0.01)** |
| $\delta = 0.9$ | **1.018 (0.005)** | **1.022 (0.006)** | $\delta = 0.9$ | 1.37 (0.02) | 1.35 (0.02) |
| $\delta = 0.5$ | **1.024 (0.003)** | **1.027 (0.002)** | $\delta = 0.5$ | **1.33 (0.01)** | **1.32 (0.02)** |
| $\delta = 0.1$ | **1.029 (0.004)** | **1.030 (0.005)** | $\delta = 0.1$ | **1.31 (0.01)** | **1.29 (0.01)** |

### *4.4.2 Language Modeling*

We also evaluate performance on language modeling tasks, where the goal is to predict the next word. We train and evaluate models on both the Penn Treebank (PTB) corpus (Marcus

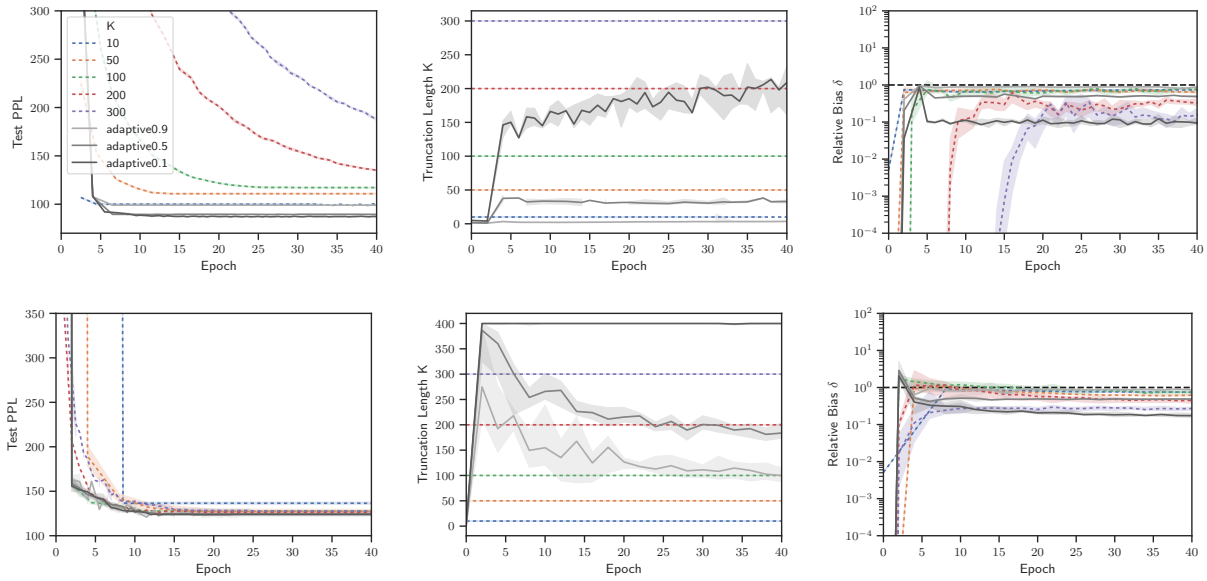


Figure 4.2: Language Modeling Results. (left) Test PPL vs epoch, (center) $\hat{\kappa}(\delta, \theta_n)$ vs Epoch, (right) $\hat{\delta}(K)$ vs epoch. (Top) PTB (bottom) Wiki2. Error bars are 95 percentiles over 50 minibatch estimates. Solid dark lines are our adaptive TBPTT methods, dashed colored lines are fixed TBPTT baselines. Our adaptive methods are competitive with the best fixed $K$ methods, while controlling the relative bias.

et al., 1993; Mikolov et al., 2010) and the Wikitext-2 (Wiki2) corpus (Merity et al., 2016). The PTB corpus contains about 1 millon tokens with a truncated vocabulary of 10k. The Wikitext-2 is twice the size of PTB and with a vocabulary of 30k.

**Model and Training Setup** For both the PTB and Wiki2 corpus, we train 1-layer LSTMs with a word embedding layer input and a linear output layer. The embedding dimension, hidden state, and cell state dimensions are all 900 for the PTB following Lei et al. (2017) and 512 for the Wiki2 corpus following Miller and Hardt (2019). We use a minibatch size of $S = 32$ and a fixed learning rate of $\gamma = 10$ for fixed TBPTT $K \in [10, 50, 100, 200, 300]$ and our adaptive TBPTT method $\delta \in [0.9, 0.5, 0.1]$. We set $W = 400$, $K_0 = 100$ and $[K_{\min}, K_{\max}] = [10, 400]$ for Algorithm 6.

**Results** Figure 4.2 (left) presents the test PPL and $K_n$ against the training epoch for both language modeling tasks. We again see that our adaptive methods are competitive with the best fixed $K$ methods, while controlling the relative bias. From the $K(\delta, \theta_n)$ vs epoch figures, our adaptive method seems to quickly converge to a constant. Therefore, on the real language data task, we have transformed the problem of selecting a fixed-$K$ to choosing a continuous parameter $\delta \in (0, 1)$. Additional figures for the validation PPL and tables of numerical values can be found in Appendix B.2.

Table 4.2 is a table of the test PPL results evaluated at the 'best' validation PPL. This table provides the numeric values of the 'best' PPL values for Figures 4.2 and B.2. We see that the adaptive TBPTT performs as well as or outperforms the best fixed $K$ TBPTT.

Table 4.2: Table of PPL for Language Modeling experiments: (left) PTB, (right) Wiki2. 'Valid PPL' is the best validation set PPL. 'Test PPL' is the test set PPL at parameters of the best validation set PPL. Standard deviation over multiple initializations are in parentheses.

| PTB | | | Wiki2 | | |
|---|---|---|---|---|---|
| K | Valid PPL | Test PPL | K | Valid PPL | Test PPL |
| 10 | 99.7 (0.6) | 99.9 (0.8) | 10 | 144.2 (0.4) | 136.5 (1.3) |
| 50 | 110.4 (0.4) | 110.8 (0.8) | 50 | 133.4 (2.9) | 127.2 (2.8) |
| 100 | 116.2 (0.5) | 116.9 (0.5) | 100 | 134.4 (0.3) | 127.8 (0.5) |
| 200 | 125.2 (1.2) | 126.1 (0.9) | 200 | 130.3 (1.1) | 124.6 (0.7) |
| 300 | 161.5 (0.5) | 161.2 (0.3) | 300 | **129.6 (1.4)** | **124.0 (2.2)** |
| $\delta = 0.9$ | 100.1 (0.5) | 99.0 (0.5) | $\delta = 0.9$ | **130.0 (1.3)** | **124.1 (2.2)** |
| $\delta = 0.5$ | 90.1 (0.4) | 89.5 (0.3) | $\delta = 0.5$ | **127.2 (0.7)** | **121.7 (0.6)** |
| $\delta = 0.1$ | **88.1 (0.2)** | **87.2 (0.2)** | $\delta = 0.1$ | **127.5 (0.6)** | **121.9 (1.2)** |

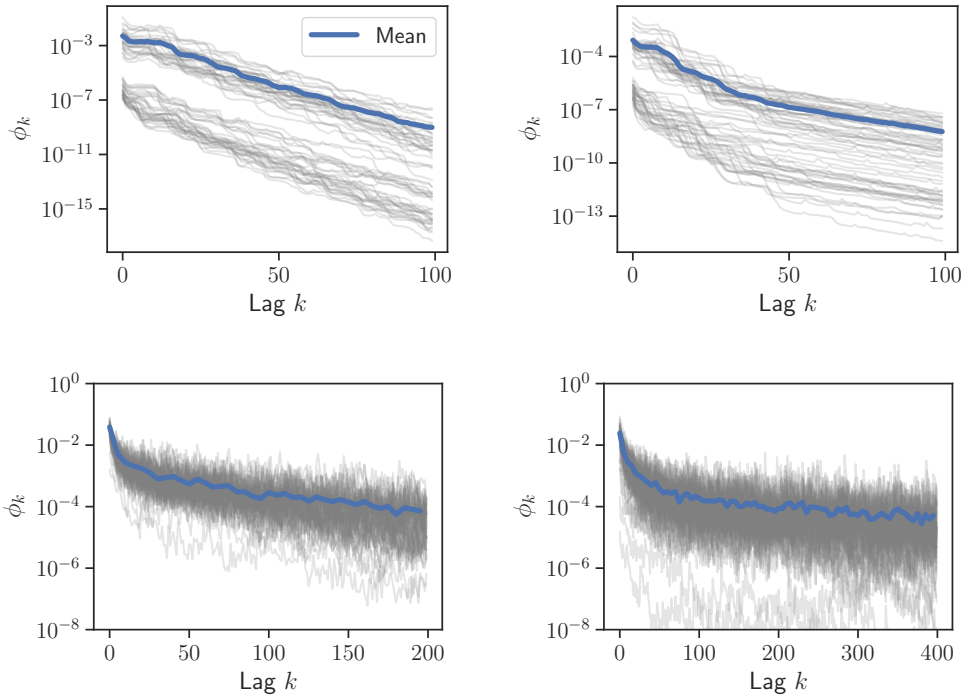


Figure 4.3: Gradient Norms $\phi_k$ vs $k$ at the best $\theta$ showing geometric decay *in expectation* (blue line) for large $k$. The gray lines are separate draws of $\phi_k$. (Top-left) fixed-length 'copy' task, (top-right) variable-length 'copy' task, (bottom-left) PTB, (bottom-right) Wiki2.

### *4.4.3 Empirically Checking (A-1)*

Figure 4.3 plots the gradient norm $\phi_k = \|\partial \mathcal{L}_s / \partial h_{s-k}\|$ vs $k$ evaluated at the best $\theta_n$ (as measured on the validation set). Note that the $y$-axis is on a log-scale We see that the expected norm $\hat{P}_{\mathcal{S}}[\phi_k]$ (blue-line) of the gradients decay geometrically for large $k$; however any individual $\phi_k$ (gray lines) are quite noisy and do not strictly decay. Therefore it appears that our RNNs satisfy (A-1), even though they are unstable, and thus the relative bias can be bounded.

### *4.4.4 Temporal Point Process Modeling*

We now consider applying our adaptive TBPTT scheme to optimizing neural networks for temporal point prediction as in (Du et al., 2016). Given a sequence $\{(y_i, t_i)_{i=1}^N\}$ of categorical observations $y_i \in \mathcal{Y}$ and observation times $t_i \in \mathbb{R}$, the task considered by Du et al. (2016) is to predict $(y_i, t_i)$ given $(y_j, t_j)_{j<i}$. Following Du et al. (2016), we model the sequence using an RNN, with input embedding layers for $y_{i-1}$ and $t_{i-1}$, and two output prediction layers: one for $y_i$ and another for $\lambda(t)$ the *conditional temporal point process intensity*. The loss now consists of two terms, which define the negative log-likelihood (NLL) for a temporal point process: (i) cross entropy loss for $y_i$ and (ii) a temporal point process loss for $\lambda(t_i)$ given by Eq.(12) in (Du et al., 2016). Du et al. (2016) also evaluate the neural network model by measuring the zero-one loss of the predicted observation $\hat{y}_i$ to $y_i$ and the root mean-squared error (RMSE) of the mean predicted observation time $\hat{t}_i = \mathbb{E}\left[t_i | \lambda(t)\right]$ to $t_i$.

**Model and Training Setup** We fit such a model using a two-layer LSTM to the 'Book Order' financial data used in (Du et al., 2016). For the input layers, we use an embedding of size 128 for the two state categorical observations $y$ and a two dimensional encoding of $t_i$ (i.e. $[t_i - t_{i-1}, t_i]$). For the two-layer LSTM, we use a hidden and cell state dimension of size 128. And the output layer dimensions follow (Du et al., 2016). For training, we use a minibatch size of $S = 64$ and a fixed learning rate of $\gamma = 0.1$ for SGD. We compare gradients from fixed TBPTT $K \in [3, 6, 9, 15, 21]$ and our adaptive TBPTT method $\delta \in [0.9, 0.5, 0.1]$. We set $W = 200$, $K_0 = 6$ and $[K_{\min}, K_{\max}] = [1, 100]$ for Algorithm 6.

The 'Book Order' dataset consists of the high-frequency financial transactions from the NYSE for a stock in one day. It consists of 0.7 million transactions records (in milliseconds) and the possible actions $\mathcal{Y}$ are 'to buy' or 'to sell'. We use the train-test split of Du et al. (2016) and split their test set in half to form a validation set.

**Results** The results of our experiment in Figures 4.4 and Table 4.3. From Figure 4.4 (top center and top right) we see that the adaptive methods control for bias, by slowly increasing

$K$. From Figure 4.4 (top left) and Table 4.3, we find that adaptive TBPTT methods achieve the best test set NLL. We also see from Figure 4.4 (bottom) and Table 4.3 that fixed TBPTT $K = 3$ performs better at predicting $y_i$ at the cost of increased error in predicting $t_i$. Similarly, fixed TBPTT $K = 15$ and $K = 21$ are better at predicting $t_i$, but poorer at predicting $y_i$. Figure 4.4 (bottom) is noisy for the individual tasks for predicting $y_i$ or $t_i$ as the algorithms trade-off between the two when optimizing the NLL.

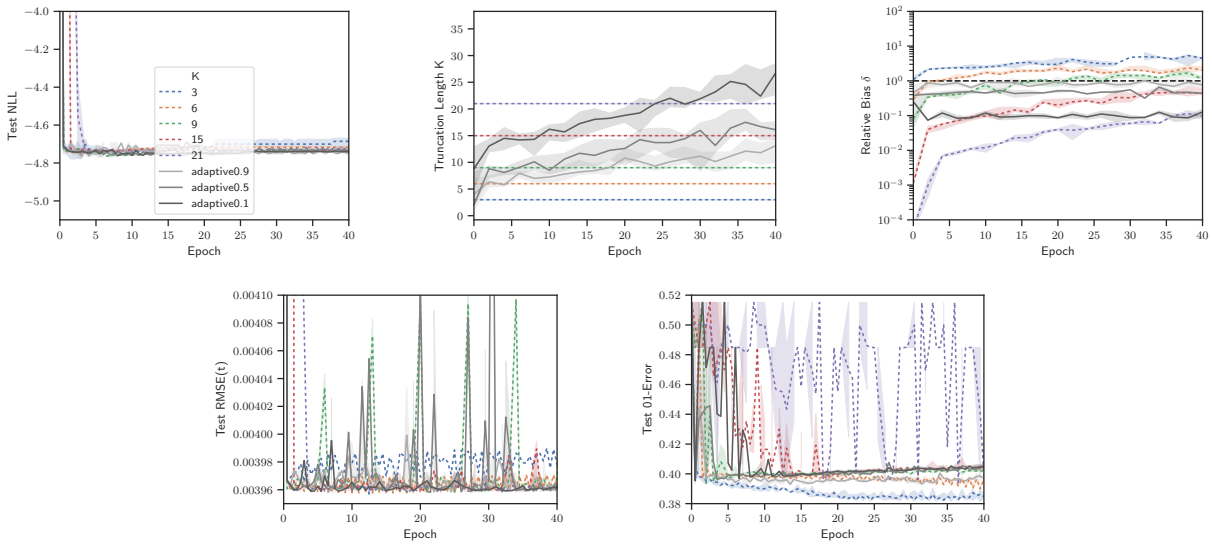


Figure 4.4: Book Order Experiment. Top row: (left) Test NLL. (center) truncation length $\hat{\kappa}(\delta, \theta_n)$, (right) relative bias $\hat{\delta}(K)$. Bottom row: (left) Test RMSE for $t$, (right) Test 01-Error for $y$. Solid dark lines are our adaptive TBPTT methods, dashed colored lines are fixed TBPTT baselines.

### *4.4.5 Challenges in Higher Dimensions*

During our experiments, we found that when training RNNs with high-dimensional $h$, but without introducing regularization on $\theta$ (in the form of dropout or weight decay), our estimates $\hat{\beta}$ were often close to or greater than 1; therefore our conservative relative error bound

Table 4.3: Table of metrics for Book Order experiment. Test metrics are evaluated at the parameters of the best valdiation set NLL. Standard deviation over multiple initializations are in parentheses.

| K | Valid NLL | Test NLL | RMSE($t$) $10^{-3}$ | 01-Loss($y$) |
|---|---|---|---|---|
| 3 | -4.983 (0.013) | -4.694 (0.015) | 3.9705 (0.0016) | **0.3827 (0.0003)** |
| 6 | -4.905 (0.006) | -4.716 (0.006) | 3.9691 (0.0007) | 0.3959 (0.0009) |
| 9 | -4.898 (0.005) | **-4.732 (0.005)** | 3.9634 (0.0003) | 0.3944 (0.0011) |
| 15 | -4.875 (0.007) | **-4.734 (0.004)** | **3.9619 (0.0011)** | 0.3971 (0.0010) |
| 21 | **-4.831 (0.026)** | -4.719 (0.019) | **3.9622 (0.0001)** | 0.4316 (0.0336) |
| $\delta = 0.9$ | -4.930 (0.016) | **-4.745 (0.009)** | 3.9641 (0.0007) | 0.3932 (0.0006) |
| $\delta = 0.5$ | -4.890 (0.003) | **-4.733 (0.013)** | **3.9662 (0.0043)** | 0.3953 (0.0002) |
| $\delta = 0.1$ | **-4.867 (0.001)** | **-4.739 (0.002)** | 3.9634 (0.0003) | 0.3954 (0.0001) |

lead to extremely large (impractical) truncation estimates $K$. During inspection, we found that although most dimensions of $\frac{\partial \mathcal{L}_s}{\partial h_{s-k}}$ decay rapidly with $k$, a few dimensions did not and these dimensions cause the overall norm $\|\frac{\partial \mathcal{L}_s}{\partial h_{s-k}}\|$ to decay slowly, thus $\hat{\beta} \approx 1$. However if these dimensions do not influence $\partial \mathcal{L}/\partial \theta$ (i.e. if $\frac{\partial h_t}{\partial \theta}$ is close to zero), then these dimensions should be ignored. Therefore, to better apply our results to higher-dimensional $h$, we suspect one should replace the Euclidean norm with a norm that weights dimensions of $\frac{\partial \mathcal{L}_s}{\partial h_{s-t}}$ by $\frac{\partial h_t}{\partial \theta}$ (such as the Mahalanobis norm $\|x\|_\Sigma = x^T \Sigma^{-1} x$ for some positive definite matrix $\Sigma$), but we leave this for future work.

## *4.5 Conclusions*

In this chapter, we developed an adaptively truncating BPTT scheme for RNNs that satistify a generalized vanishing gradient property. We showed that if the gradient decays geometrically *in expectation* (A-1), then we can control the relative bias of TBPTT (Theorem 7)

and guarantee non-asymptotic convergence bounds for SGD (Theorem 8). We additionally showed how to take advantage of these ideas in practice in Algorithm 6, by developing estimators for the relative bias based on backpropagated gradients. We evaluated our proposed method on synthetic copy and language modeling tasks finding it performs similarly to the best fixed-$K$ TBPTT schemes, while still controlling the relative bias of the gradient estimates.

There are many directions for future work. One could develop restrictions on the parameters to $\Theta_{\beta,\tau}$ during training similar to existing work on unitary and stable RNNs (see Chapter 4.1.1), but with the relaxation to only assuming gradient decay in expectation. Another direction is to extend the theoretical results of Theorem 8 to stochastic gradient optimization using momentum or acceleration such as ADAGRAD (Duchi et al., 2011), ADAM (Kingma and Ba, 2014), or RMSProp (Tieleman and Hinton, 2012) that are popular in practice. Finally, one could extend our error bounds from Chapter 4.2.2 beyond using the Euclidean norm to avoid the challenges in higher dimensions we observed in Chapter 4.4.5.

# Chapter 5

# APPROXIMATE COLLAPSED GIBBS CLUSTERING

## *5.1 Introduction*

In this chapter, we focus on Bayesian inference for clustering data using complex latent variable models.

A common task in unsupervised learning is to cluster observed data into groups that are similar. One principled approach is to infer latent cluster assignments in a hierarchical probabilistic model. Hierarchical latent variable models have the benefit of allowing for both (i) more flexible and complex models to be built from simpler distributions and (ii) statistical strength to be shared within clusters for inference.

As a motivating example, we consider clustering time series data. Specifically, we are interested in finding clusters of time series such that series within a cluster are correlated and series between clusters are independent. We take motivation from a housing application previously analyzed by Ren et al. (2017), though the methods are much more broadly applicable.

In the housing application, the goal is to estimate house values at very fine spatial resolutions, such as the census tract level. Housing trends vary over time and space, and the spatial structure is very heterogeneous where neighboring census tracts can behave quite differently: sharing information using spatial structure can be detrimental. As a result, Ren et al. (2017) simply treat the house price processes within the set of census tracts as a collection of time series, ignoring spatial structure. However, due to the scarcity of spatiotemporally localized house sales observations, the census tracts cannot be analyzed independently while still providing reasonable estimates of house value. To handle this data scarcity, Ren et al. (2017) proposed a method for discovering groups of *correlated* census tracts extending the correlated

clustering model of Palla et al. (2012) to time series data using SSMs. Through discovering a clustering of time series, one can improve estimates of local trends by sharing information via a form of multiple shrinkage.

The goal of learning the clustering is to improve predictive performance of house values. This type of clustering-of-time-series approach also proved useful in crime forecasting (Aldor-Noiman et al., 2016). In other scenarios, the goal of clustering may be to produce an interpretable structure for understanding the relationships between time series. There is widespread demand for such time series clustering approaches.

Although the Bayesian model provides performance gains over alternative approaches, a significant limitation of the method is the complexity of the Bayesian inference procedure. In particular, inference of the cluster assignments of the individual time series presents a huge computational bottleneck: each single assignment update requires a likelihood computation with runtime scaling cubicly with the number of time series per cluster. This costly step has to be repeated *for each time series and each possible cluster assignment at each iteration* of the Bayesian inference algorithm. Unfortunately, due to the structure of the problem, there are no opportunities for sharing computations between steps.

More specifically, to perform Bayesian inference of the cluster assignments, a Gibbs sampler is used which iteratively draws individual cluster assignments and model parameters from the posterior conditioned on all other variables (see Chapter 2.1.1). While such *naive* Gibbs sampling is theoretically guaranteed to converge, in practice, it is known to mix slowly in high dimensions (Van Dyk and Park, 2008). A popular modification is *collapsed* Gibbs sampling, which iteratively draws from marginals of the posterior by integrating out variables. Integrating out variables reduces the dimension of the posterior and often eliminates local modes arising from tightly coupled variables (Liu, 2008).

Unfortunately, for complex models, sampling from the marginal posterior can be analytically intractable or computationally prohibitive. For example, in the time series clustering model of Ren et al. (2017), collapsed Gibbs sampling requires running a computationally intensive Kalman smoother per iteration that scales cubically in the number of series per

cluster. Another common example is mixture modeling with non-conjugate emissions. One example is mixture modeling with Student-$t$ emissions, which is popular in robust modeling due to its ability to model heavy-tails. In such cases, the emission parameters cannot be directly integrated out due to non-conjugacy. In these two cases, which we use as illustrative of the challenges faced in many models appropriate for real-data analysis, collapsed Gibbs sampling is either infeasible or impractical.

A recently popular alternative MCMC technique to Gibbs sampling is the class of Hamiltonian Monte Carlo (HMC)-like algorithms (Neal et al., 2011) and their scalable stochastic gradient variants (see Chapter 2.1.2). These algorithms utilize (stochastic) gradient information about the target posterior and simulate continuous dynamics to efficiently explore the distribution. However, these methods only apply to fixed-sized continuous parameter spaces. Therefore, in our setting, the discrete latent variables must be marginalized out, which (i) requires handling *label switching*, (ii) does not apply to nonparametric mixtures, and (iii) are slow for large clusters with complex likelihoods. As such, this class of MCMC techniques does not maintain the spirit of collapsed Gibbs. One such approximately collapsed method is *griddy Gibbs*; however it is limited to univariate variables (Ritter and Tanner, 1992).

We instead stay within the collapsed Gibbs framework and aim to address how to handle the challenging required integrals in many scenarios. We draw inspiration from expectation propagation (EP) (Minka, 2001; Seeger, 2005) and approximate the intractable integrals in cases where moments can be matched (See Chapter 2.1.3). Traditionally, EP is a method of approximating a target distribution with a distribution from a fixed simpler family of distributions, usually an exponential family. In our case, instead of using EP to directly approximate the posterior of cluster assignments (Fan and Bouguila, 2014), we use EP to approximate the conditional posterior of the nuisance parameters we wish to collapse out. By selecting an appropriate family of distributions for our EP approximation, we can efficiently integrate out parameters, leading to quicker mixing. Importantly, through the use of EP, we still integrate over an approximation of our uncertainty when collapsing the nuisance variables.

Our experiments for the time series clustering model and mixture of Student-$t$ model demonstrate the effectiveness of our proposed approach finding that our approximate EP samplers mix as well as collapsed Gibbs in terms of performance versus number of iterations, while significantly outperforming collapsed Gibbs in terms of runtime.

The remainder of this chapter is organized as follows. First we propose our general framework for approximate collapsed Gibbs sampling using EP in Chapter 5.2. Then, we present two applications of the framework – time series clustering and robust clustering – in Chapter 5.3. Finally, we present experiments on synthetic and real data in Chapter 5.4.

## *5.2 Approximate Collapsed Gibbs Sampling*

Our goal is to develop efficient approximate collapsed Gibbs samplers for mixture models, when the required integrals for collapsed Gibbs sampling are intractable. Recall from Chapter 2.1.1 the required integrals are Eqs. (2.5) and (2.6) are

$$p(z_i|z_{-i}) = \int p(z_i|\pi)p(\pi|z_{-i})\ d\pi \tag{2.5}$$

$$p(y_i|y_{-i}, z) = \int p(y_i|z_i, \phi)p(\phi|y_{-i}, z_{-i})\ d\phi\,. \tag{2.6}$$

Generically, we can write the intractable integrals of interest as

$$F(\zeta_i) = \int f_i(\zeta_i, \theta)p(\theta \mid \zeta_{-i})\ d\theta\ . \tag{5.1}$$

where for Eqs. (2.5) and (2.6)

$$\zeta_i = z_i\ ,\ \ f_i(z_i, \theta) := p_\pi(z_i|\pi)\ ,\ \ p(\theta|z_{-i}) := p(\pi|z_{-i})\ ,$$
$$\zeta_i = z_i, y_i\ ,\ \ f_i(z_i, \theta) := p_\pi(y_i|z_i, \phi)\ ,\ \ p(\theta|z_{-i}) := p(\phi|y_{-i}, z_{-i})\ .$$

We assume that Eq. (5.1) is intractable as $p(\theta|\zeta_{-i})$ either does not have an analytic form or is computationally intractable to calculate. As a result, integral approximation methods, such as the Laplace approximation, cannot be directly applied to Eq. (5.1).

Our key idea is to replace $p(\theta|\zeta_{-i})$ with an approximate distribution $q(\theta|\zeta_{-i})$ such that

$$\tilde{F}(\zeta_i) = \int f_i(\zeta_i, \theta)q(\theta \mid \zeta_{-i})\ d\theta \tag{5.2}$$

is a good approximation to $F(\zeta_i)$ in Eq. (5.1). To do this, we borrow ideas from EP, an iterative method for minimizing the Kullback-Leibler (KL) divergence between an approximation $q$ and a posterior $p$ (See Chapter 2.1.3).

### *5.2.1 EP for Approximate Collapsed Gibbs*

There are a couple of necessary leaps to see how we apply EP to approximate the integrals of Eqs. (2.5) and (2.6). First, instead of approximating the posterior $p(\theta|y)$ with $q(\theta|y)$, we are interested in approximating $p(\theta|\zeta_{-i})$ with the cavity distribution $q(\theta|\zeta_{-i})$ for each $i$.

Note that our target distribution $p(\theta|\zeta_i)$ is conditioned on the *sampled* latent variables $z_{-i}$. In contrast to a fixed target distribution $p(\theta|y)$, our target distribution $p(\theta|\zeta_{-i}^{(s)})$ changes as we sample $z^{(s)}$ at every iteration $s$. Therefore, the fixed points of our update scheme, the best EP approximation $q^*(\theta|\zeta^{(s)})$, change at each iteration. To ensure stable performance, one approach would be to run EP to convergence at every iteration

- Sample $z_i^{(s+1)} \sim \tilde{\Pr}(z_i \,|\, y, z_{-i}^{(s)})$ approximately integrating out $\pi, \phi$ using $q^*(\theta \,|\, \zeta_{-i}^{(s)})$.
- Calculate $q^*(\theta \,|\, \zeta^{(s+1)})$ by updating *all site* approximations $\tilde{f}_j(\theta)$ until convergence.

At each iteration, at most one latent variable $z_i$ changes; therefore we only need to update the site approximations $\tilde{f}_j(\theta)$ belonging in $z_i^{(s)}$ and $z_i^{(s+1)}$. However, this is computationally costly as updating all sites in both clusters would take $\mathrm{O}(N)$ time.

Instead, we choose a second approach that leverages our existing approximation $q^{(s)}(\theta|\zeta^{(s)})$ and only updates site approximation $\tilde{f}_i$ after sampling $z_i$. That is,

- Sample $z_i^{(s+1)} \sim \tilde{\Pr}(z_i \,|\, y, z_{-i}^{(s)})$ approximately integrating out $\pi, \phi$ using $q^{(s)}(\theta \,|\, \zeta_{-i}^{(s)})$.
- Calculate $q^{(s+1)}(\theta \,|\, \zeta^{(s+1)})$ by updating *only* site approximation $\tilde{f}_i(\theta)$.
- Periodically (e.g. after a full pass), update all site approximations until convergence.

By not updating all site approximations to convergence, we introduce some error between our new approximation $q^{(s)}(\theta|\zeta^{(s)})$ and the best EP approximation $q^*(\theta|\zeta^{(s)})$. This error arises due to using 'stale' site approximations. The idea is similar in spirit to "Parameter Server" methods that infer global parameters by passing 'stale' sufficient statistics between machines (Li et al., 2014; Ahmed et al., 2012). Intuitively, we expect this error to be small when our approximating family closely resembles the likelihood and when the latent variables $z$ change slowly. By periodically including full EP update passes, we can bound the convergence error between this sparse update scheme and full EP (up to model specific constants). A more precise description and analysis of this convergence error, including illustrative synthetic experiments, can be found in Appendix C.3. In our experiments (Chapter 5.4), we found that it was even sufficient to omit the full pass and only update the local site approximation at each iteration.

## *5.3 Case Studies*

We consider two motivating examples for the use of our EP-based approximate collapsed Gibbs algorithm. The first is a mixture of Student-$t$ distributions, which can capture heavy-tailed emissions crucial in robust modeling (i.e., reducing sensitivity to outliers). The second example is the correlated time series clustering model of Ren et al. (2017).

### *5.3.1 Mixture of Multivariate Student-t*

The multivariate Student-$t$ (MVT) distribution, is a popular method for handling robustness (Portilla et al., 2003; Peel and McLachlan, 2000; Svensén and Bishop, 2005; Andrews and Mallows, 1974). To perform robust Bayesian clustering of data $y = y_{1:N}$ in $\mathbb{R}^d$, we use MVT as the emission distributions:

$$p(y_i \,|\, z_i = k, \mu, \Sigma, \nu) := t_{\nu_k}(y_i \,|\, \mu_k, \Sigma_k) \ , \tag{5.3}$$

$$t_\nu(y \,|\, \mu, \Sigma) \propto |\Sigma|^{-1/2} \left(1 + \frac{(y_i - \mu)^T \Sigma (y_i - \mu)}{\nu}\right)^{-\frac{\nu+d}{2}} ,$$

where $\mu_k, \Sigma_k, \nu_k$ are the mean, covariance matrix and degrees of freedom parameter for cluster $k$, respectively. A common construction of the MVT arises from a scale mixture of Gaussians

$$t_\nu(y \mid \mu, \Sigma) = \int \mathcal{N}(y \mid \mu, \Sigma/u)\Gamma(u \mid \nu/2, \nu/2)\, du \ . \tag{5.4}$$

For this paper, we focus on the case where $\nu$ is known and learn $z, \mu, \Sigma$ by Gibbs sampling. However, all of the following sampling strategies can be extended to learn $\nu$ by adding a Metropolis-Hasting step.

**'Naive' and Collapsed Gibbs:** Because the MVT likelihood is non-conjugate to standard exponential family priors, the posterior conditional distribution for $\mu, \Sigma$ does not have a closed analytic form. However, by exploiting the representation of a MVT as a scale mixture of Gaussians, Eq. (5.4), we can use data augmentation to construct a Gibbs sampler with analytic steps (Liu, 1996; Damlen et al., 1999). By introducing auxiliary variables $u = \{u_{i,k}\}$ for each observation-cluster pair $i, k$, we can replace the MVT likelihood with a Gaussian conditioned on $u$.

The `naive Gibbs` sampler can be straightforwardly derived on the expanded space of $z, \mu, \Sigma, u$ as

$$\begin{aligned}
z_i \mid z_{-i}, \mu, \Sigma, u, y &\sim \mathcal{N}(y_i \mid \mu_{z_i}, \Sigma_{z_i}/u_{i,z_i})p(z_i \mid z_{-i}) \\
\mu_k, \Sigma_k \mid z, u, y &\sim \mathcal{NIW}(\mu, \Sigma \mid \tau_{\mu_k,\Sigma_k}(z, u, y)) \\
u_{i,k} \mid z, \mu, \Sigma, y &\sim \Gamma(u \mid \tau_{u_{i,k}}(z, \mu, \Sigma, y)) \ ,
\end{aligned}$$

where the specific form of the parameters $\tau$ is given in the Appendix C.1.

Conditioned on $u$, the posterior for $\mu, \Sigma$ is conjugate to the likelihood of $y$. Therefore, we can integrate out $\mu, \Sigma$ when sampling $z$. We cannot completely collapse out $\mu, \Sigma$ as they are required for sampling $u$ (which is required for sampling $z$). The result is a (partially

collapsed) `blocked Gibbs` sampler that samples

$$\left.\begin{aligned} &z_i \,|\, z_{-i}, u, y \sim t(y_i \,|\, \tau(z,u)) p(z_i \,|\, z_{-i}) \\ &\mu_k, \Sigma_k \,|\, z, u, y \sim \mathcal{NIW}(\mu, \Sigma \,|\, \tau_{\mu_k,\Sigma_k}(z,u,y)) \end{aligned}\right\} \text{ block}$$
$$u_{i,k} \,|\, z, \mu, \Sigma, y \sim \Gamma(u \,|\, \tau_{u_{i,k}}(z, \mu, \Sigma, y)) \ .$$

For further details, see Appendix C.1.

Although the data-augmentation method allows us to construct analytic Gibbs samplers for a mixture of MVTs, this approach has serious drawbacks. By expanding the representation of the MVT with $u$, we (i) increase the number local modes and (ii) increase computation by sampling $NK$ auxiliary parameters. For these reasons, the data-augmentation approach is not commonly used beyond small $K$.

**Approximate Collapsed Gibbs:** We can handle the non-conjugacy of the MVT likelihood (Eq. (5.3)) using our framework by approximately collapsing out $\mu, \Sigma$ without introducing auxiliary variables $u$.

Using the Gaussian scale mixture representation of the MVT, the collapsed likelihood term for $z_i$ is

$$F(\zeta_i) = \int \underbrace{p(\mu_{z_i}, \Sigma_{z_i} \,|\, y_{-i}, z_{-i})}_{p(\theta|\zeta_{-i})} \times \underbrace{\int \mathcal{N}(y_i \,|\, \mu_{z_i}, \Sigma_{z_i}/u) \Gamma(u|\nu/2, \nu/2)\ du}_{f_i(\zeta_i,\theta)}\ d\mu\, d\Sigma \tag{5.5}$$

By selecting our approximation family $q(\mu_k, \Sigma_k|\zeta)$ to be a $D$-dimensional normal inverse-Wishart, and by swapping the order of integration, our approximation to Eq. (5.5) becomes a 1-dimensional integral over $u$

$$\tilde{F}(\zeta_i) = \int \Gamma(u|\nu/2, \nu/2) \times \underbrace{\int q(\mu_{z_i}, \Sigma_{z_i}|\zeta_{-i}) \mathcal{N}(y_i \,|\, \mu_{z_i}, \Sigma_{z_i}/u)\, d\mu\, d\Sigma}_{\propto \text{ normal inverse-Wishart}(u)}\ du \tag{5.6}$$

Because the integrand is a ratio of normal inverse-Wishart normalizing constants, which are analytically known, this 1-dimensional integral can be calculated numerically. Similarly, the

moments required for EP are also calculated as a 1-dimensional integral of weighted normal inverse-Wishart sufficient statistics. For complete details, see the Appendix C.1.

For the MVT, the key innovation is using EP to keep track of an approximation $q(\theta \mid \zeta_{-i})$ (here a normal inverse-Wishart) for $p(\theta \mid \zeta_{-i})$, thus allowing Eq. (5.6) to be numerically tractable. This approach allows us to approximately collapse out $\mu, \Sigma$, which in turn enables us to avoid sampling the auxiliary variables $u$ introduced in the naive and blocked samplers.

### *5.3.2 Time Series Clustering*

Given a collection of time series, we are interested in finding clusters of series such that series within a cluster are *correlated* and between clusters are *independent*. Recall, we take motivation from the housing application analyzed by Ren et al. (2017). The goal is to estimate housing price trends at fine spatial resolutions. The series cannot be analyzed independently while providing reasonable value estimates due to the scarcity of spatiotemporally localized house sales observations. The time series clustering helps handle this data scarcity by sharing information across regions discovered to be related.

Let $y = \{y_i \in \mathbb{R}^T\}_{i=1}^N$ be a collection of $N$ observed time series of length $T$ (different lengths and missing data can also be accommodated). The individual series follow a state space model:

$$\begin{aligned} x_{i,t} &= a_i x_{i,t-1} + \epsilon_{i,t} & \epsilon_{i,t} &\sim \mathcal{N}(0, \sigma_{i,t}^2) \\ y_{i,t} &= x_{i,t} + \nu_{i,t} & \nu_{i,t} &\sim \mathcal{N}(0, \sigma_{y_i}^2) \ . \end{aligned} \tag{5.7}$$

Here, $x_{i,t} \in \mathbb{R}$. Clusters of correlated time series are induced by introducing latent cluster assignments $z$ and modifying the latent noise process $\epsilon_{i,t}$ to follow a cluster-specific latent factor process $\eta_{z_i,t}$ with factor loading $\lambda_i$:

$$\epsilon_{i,t} = \lambda_i \eta_{z_i,t} + \tilde{\epsilon}_{i,t} \qquad \eta_{k,t} \sim \mathcal{N}(0,1) \ , \ \tilde{\epsilon}_{i,t} \sim \mathcal{N}(0, \sigma_x^2) \ . \tag{5.8}$$

Marginalizing over $\eta_{k,t}$, the covariance between the latent noise processes $\mathrm{Cov}(\epsilon_{i,t}, \epsilon_{j,t} | z) = \lambda_i \lambda_j + \sigma_x^2 \mathbf{1}_{i=j}$ are correlated if $z_i = z_j$ and 0 otherwise. Combining Eq. (5.7) and (5.8), an

equivalent representation for the individual latent series dynamics is

$$x_{i,t} = a_i x_{i,t-1} + \lambda_i \eta_{z_i,t} + \tilde{\epsilon}_{i,t} \ . \tag{5.9}$$

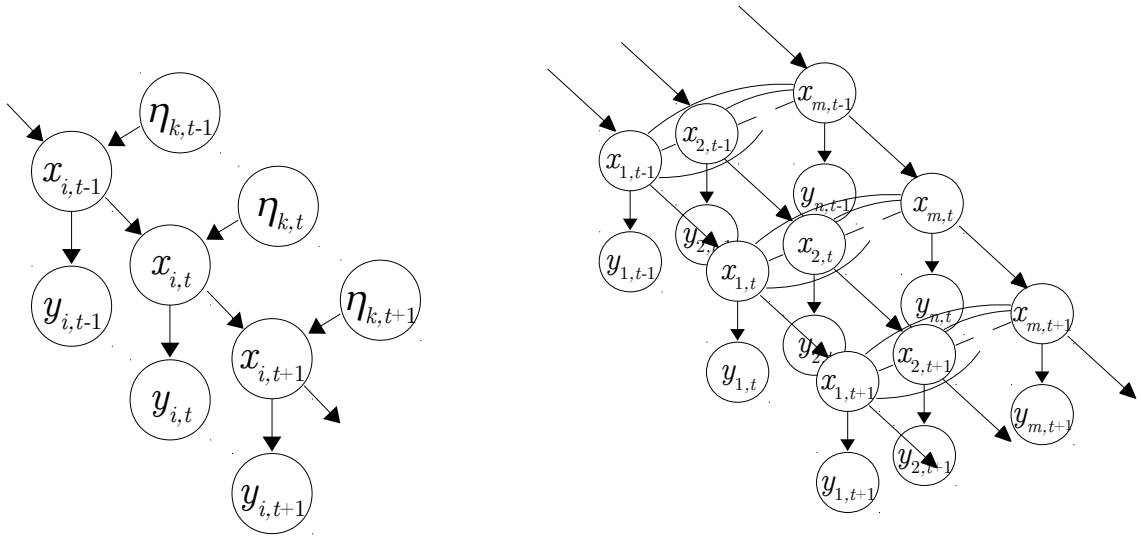


Figure 5.1: Individual time series likelihoods (left) without and (right) with collapsing the latent factor processes $\eta_{1:K,1:T}$.

**'Naive' and Collapsed Gibbs:** The simplest Gibbs sampler is to iteratively sample all variables, including the latent states $x$. Instead, as in Ren et al. (2017), we exploit the time series structure of the state space model and always integrate out $x$ using a Kalman smoother (Barber et al., 2011; Bishop, 2006) with a slight modification to account for the time-varying mean term $\lambda_i \eta_{z_i}$:

$$p(y_i|z_i, \phi) := p(y_i|z_i, \eta) = \int \prod_{t=1}^{T} p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) \ dx_t \ . \tag{5.10}$$

See Figure 5.1(left) for a visualization of this partially collapsed likelihood. In this model, the Dirichlet prior $p_\pi$ over cluster assignments $z$ is conjugate and can be analytically marginal-

ized. We refer to this partially collapsed scenario that conditions on the latent factor processes $\eta_{1:K,1:T}$ as `naive Gibbs`. Note that running the Kalman smoother on one series has a runtime complexity of $\mathcal{O}(T)$ (Barber et al., 2011). By evaluating this for each potential cluster assignment, sampling $z_i$ has a total runtime complexity of $\mathcal{O}(TK)$. Unfortunately, this naive sampler is sensitive to initialization and exhibits poor performance.

To overcome this, Ren et al. (2017) constructed a collapsed Gibbs sampler that additionally integrates out $\eta$. From the state space model of Eq. (5.9), collapsing out $\eta$ induces dependencies between the latent states $x$ assigned to the same cluster (see Figure 5.1(right)). The conditional covariance structure is specified under Eq. (5.8). As a result, calculating the collapsed likelihood term requires running the Kalman smoother on all series $y_j$ assigned to the same cluster. Although analytically tractable, the computational complexity of the Kalman smoother scales cubically in the number of series (Barber et al., 2011). Therefore, the collapsed likelihood is computationally prohibitive for large cluster sizes. We refer to this sampler as `collapsed Gibbs`.

For complete details of the Kalman smoother for both naive and collapsed Gibbs samplers, see Appendix C.2.

**Approximate Collapsed Gibbs:** We apply our framework of Chapter 5.2 to reduce the computational overhead by calculating an approximation to the collapsed likelihood term

$$F(\zeta_i) = \int \underbrace{p(\eta \mid y_{-i}, z_{-i})}_{p(\theta|\zeta_{-i})} \times \underbrace{\prod_{t=1}^{T} p(y_{i,t} \mid x_{i,t}) p(x_{i,t} \mid x_{i,t-1}, \eta_{z_i,t}) \, dx_i}_{f_i(\zeta_i,\theta)} \, d\eta \ . \tag{5.11}$$

By selecting $q(\eta_k \mid z) \in \{\mathcal{N}_T(\eta_k \mid \mu_k, \mathrm{diag}(\sigma_k))\}$, i.e., as a $T$-dimensional diagonal Gaussian, we can factorize $q$ over $t$ and approximate Eq. (5.11) with

$$\tilde{F}(\zeta_i) = \int \prod_{t=1}^{T} q(\eta_{z_i,t} \mid \zeta_{-i}) \times p(y_{i,t} \mid x_{i,t}) p(x_{i,t} \mid x_{i,t-1}, \eta_{z_i,t}) \, dx_i \, d\eta \ . \tag{5.12}$$

This integrand has the same graphical model form as in the naive Gibbs case (Figure 5.1(left)) and can be calculated in $\mathcal{O}(T)$ time using the Kalman smoother modified to account for $\eta$.

To update the site approximations $\tilde{f}_i(\eta) \in \{C_i \mathcal{N}_T(\eta \mid \mu_i, \text{diag}(\sigma_i))\}$, we must calculate the marginal mean and variance of $\eta_{k,t}$. Fortunately, the moments of $\eta_{k,t}$ can be calculated given the pairwise distribution of $(x_t, x_{t+1})$ extracted from the Kalman smoother. For further details, see Appendix C.2.

## *5.4 Experiments*

To assess the computational complexity and cluster assignment mixing of our sampling methods, we perform experiments on both synthetic and real data from the considered models of Chapter 5.3.1 and 5.3.2 [1].

To evaluate our sampling methods, we measure the normalized mutual information (NMI) of the inferred cluster assignment to the ground truth when known. When the clustering is not known, we compare the inferred cluster assignment to the clustering associated with the MAP of the exact collapsed sampler run for a long time. NMI is an information theoretic measure of similarity between cluster assignments (Vinh et al., 2010), with NMI maximized at 1 when the assignments are equal up to a permutation and NMI minimized at 0 when the assignments share no information.

### *5.4.1 Mixture of Multivariate Student-t*

We consider fitting mixtures of MVTs to synthetic data and a low-dimensional variational auto-encoder embedding of the MNIST dataset. We compare the `naive Gibbs`, `blocked Gibbs` and approximate collapsed `EP Gibbs` samplers of Chapter 5.3.1. For this model, the exact collapsed sampler is not available.

For our synthetic experiments, we generated data from a mixture of MVTs with $\nu = 5$, $K = 20$, and $N = 600$. The cluster mean and variance parameters $\mu, \Sigma$ were drawn from the normal inverse-Wishart. We considered three different signal-to-noise (SNR) settings by increasing the variance of $\mu$ ranging from hard to easy. Figure 5.2 shows the performance

[1] Python code is available at `https://github.com/aicherc/EP_Collapsed_Gibbs`

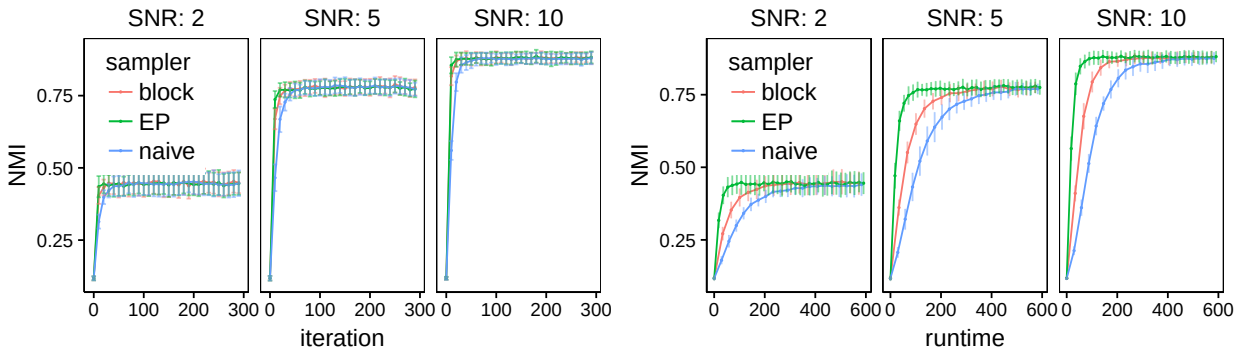


Figure 5.2: Comparing three Gibbs samplers on synthetic data: blocked Gibbs, our EP approximate collapsed Gibbs, and naive Gibbs. (Top) NMI vs iteration, (bottom) NMI vs runtime. The mean value and error bars are over 25 trials.

of each sampler. From the iteration plot (left), we see that all methods have similar performance. From the runtime plot (right), we see that `EP Gibbs` > `blocked Gibbs` > `naive Gibbs` in terms of performance.

For our real dataset example, we consider clustering a $\mathbb{R}^3$ embedding of MNIST handwritten digit images (LeCun and Cortes, 2010), where the ground truth cluster assignments are taken to be the true digit-labels. A simple past approach to clustering MNIST consists of running PCA to learn a low dimensional embedding followed by clustering. Instead of PCA, we use variational autoencoders (VAEs), an increasingly popular and flexible method for unsupervised learning of complex distributions (Kingma and Welling, 2013). VAEs learn a probabilistic encoder to infer a latent embedding such that the latent embedding comes from a simple distribution (usually an isotropic Gaussian). In practice, when we the data come from different classes, the VAE warps the clusters apart making them non-Gaussian.

We trained a simple VAE on the MNIST dataset with latent embedding dimension 3 using the same architecture as in (Kingma and Welling, 2013). The scatter plot, Figure 5.3 (left), visualizes the VAE embedding, with separate colors for each digit.

We fit the MVT samplers from Chapter 5.3.1 using $\nu = 5$ and $K = 10$ on a stratified subset of MNIST ($N = 10000$) using the VAE embedding. In addition, we also fit a `Gaussian`

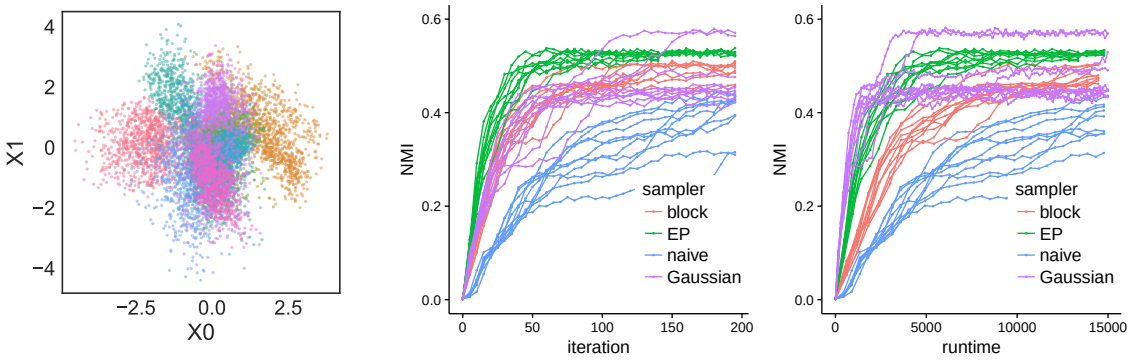


Figure 5.3: Comparing four Gibbs samplers on the VAE embedding of *MNIST*: blocked Gibbs, our EP Gibbs, naive Gibbs, and Gaussian mixture model. (Left) VAE embedding, (middle) NMI vs iteration, (right) NMI vs runtime.

mixture model using a collapsed Gibbs sampler to illustrate the potential advantage of the more robust MVT likelihood. In Figure 5.3, we present the results comparing each sampler's clustering assignment with the ground truth labels. Figure 5.3 (middle) plots NMI vs iteration. We see that the MVT `EP Gibbs` and `blocked Gibbs` methods out perform the `Gaussian` mixture model per iteration (on average). Figure 5.3 (right) is NMI vs runtime. We see that `EP Gibbs` is much faster than the alternative data-augmentation MVT samplers (due to sampling $u_{i,k}$). We expect the runtime improvements of EP over data-augmentation Gibbs to be greater for larger $K$.

### *5.4.2 Time Series Clustering*

For synthetic data drawn from the model of Chapter 5.3.2, we first demonstrate that our approximate collapsed sampler `EP Gibbs` is competitive with `naive Gibbs`' running time and with `collapsed Gibbs`' mixing rate. We simulate data using $T = 200$, $K = 20$, $\sigma_x^2 = 0.01$, $\sigma_y^2 = 1$, $a_i = 0.95$, and $\lambda = 1$. Aside from $K$, we treat all parameters as unknown in our sampling.

For our first experiment, in Figure 5.4 (left) we compare the runtime per iteration as the number of series $N$, and thus number of series per cluster, varies. We clearly see that

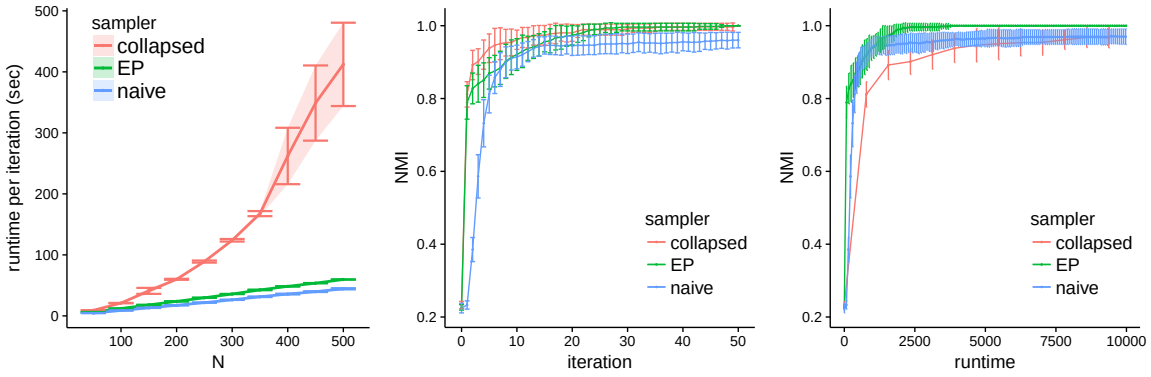


Figure 5.4: Comparing three Gibbs samplers on synthetic time series: collapsed Gibbs, our EP Gibbs, and naive Gibbs. (Left) runtime per iteration vs. number of series $N$, (center) NMI vs. iteration, (right) NMI vs. average runtime. The mean value and error bars (st.dev.) are over 20 trials.

`collapsed Gibbs` scales super-linearly, while the other two methods have linear scaling. This validates that `collapsed Gibbs` is intractable for large datasets and motivates considering faster approximate samplers.

For our second experiment, we fix $N = 300$ and measure the performance of all three samplers in terms of log-likelihood versus Gibbs iteration. From Figure 5.4 (center), we see that on average, `collapsed Gibbs` and our `EP Gibbs` samplers both mix quickly to a higher log-likelihood than `naive Gibbs`, which slowly explores its high dimensional parameter space and is sensitive to local modes. Importantly, when scaling the $x$-axis by the average runtime per iteration of each method, we clearly see in Figure 5.4 (right) that our `EP Gibbs` sampler handily outperforms both competitors. `Collapsed Gibbs` is particularly poor on these axes because of the high per-iteration runtime. Trace plots and box plots of model parameters, rather than resulting log-likelihood, are provided in Appendix C.4 and show that the approximate Gibbs sampler produces similar results to Gibbs in terms of sampled mean and variance of parameters.

To demonstrate the accuracy of our approximate sampler on real time series data, we replicate the experiment of Ren et al. (2017) to predict house prices in the city of Seattle. The

data consists of 124,480 housing transactions in 140 census tracts (series) of Seattle from 1997 to 2013, partitioned into a 75-25 train test split stratified by series. Each transaction consists of a sales price, our prediction target, and house-specific covariates such as 'lot square-feet' or 'number of bathrooms'. We first remove a global trend and jointly fit the time series clustering model with series-specific regressions on individual transaction covariates. Full details can be found in the Appendix C.5.

We compare fitting this model using our approximate sampler to the collapsed Gibbs sampler using the same error metrics as in (Ren et al., 2017): root mean-squared error (RMSE) in price, and mean / median / 90th percentile of absolute percent error (APE).

The performance of our approximate sampler `EP Gibbs` and the `collapsed Gibbs` sampler are presented in Table 5.1; we present NMI comparisons to the MAP of the `collapsed Gibbs` in Figure 5.5. We see that both algorithms for time series clustering produce similar results on all metrics (within a standard deviation). However, `EP Gibbs` achieves superior performance much more rapidly (in half the time). As such, we view our algorithm as an attractive alternative in this case. Furthermore, note that our gains would only increase with the size of the dataset, e.g., number of regions $N$, a limitation of the original approach (Ren et al., 2017).

Table 5.1: Test metrics on housing data averaged over 10 trials. Parenthetical values are one standard deviation.

| metric | collapsed | EP |
|---|---|---|
| RMSE | 125280 (50) | 125280 (80) |
| Mean APE | 16.20 (0.01) | 16.20 (0.01) |
| Median APE | 12.07 (0.01) | 12.07 (0.01) |
| 90th APE | 34.17 (0.07) | 34.22 (0.05) |
| Runtime | 121.6 (8.1) | 62.8 (3.7) |

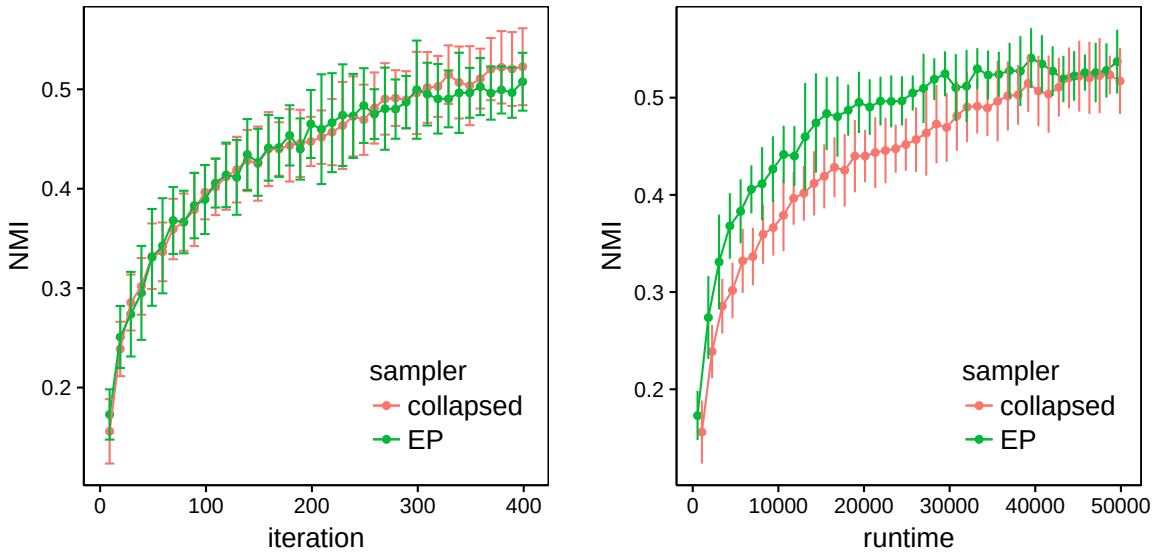


Figure 5.5: Comparisons to the collapsed sampler MAP assignment on the housing data: collapsed Gibbs and our EP Gibbs. (Left) NMI vs iteration, (right) NMI vs runtime.

Comparisons with alternative baseline models are in Appendix C.5. These comparisons illustrate the importance of the time series clustering approach in improving predictive performance.

## *5.5 Conclusions*

In this chapter, we presented a framework for constructing approximate collapsed Gibbs samplers for efficient inference in complex clustering models. The key idea is to approximately marginalize the nuisance latent variables by using EP to approximate the conditional distributions of the latent variables with an individual observation removed. By approximating this conditional, the required integral becomes tractable in a much wider range of scenarios than that of conjugate models. Our use of this EP approximation takes two steps from its traditional use: (1) we approximate a (nearly) full conditional rather than directly targeting the posterior, and (2) our targeted conditional changes as we sample the cluster assignment variables. For the latter, we provided a brief analysis and demonstrated the impact of the changing target, drawing parallels to previously proposed samplers that use stale sufficient

statistics.

We demonstrated how to apply our EP-based approximate sampling approach in two applications: mixtures of Student-$t$ distributions and time series clustering. Our experiments demonstrate that our EP approximate collapsed samplers mix more rapidly than naive Gibbs, while being computationally scalable and analytically tractable. We expect this method to provide the greatest benefit when approximately collapsing large parameter spaces.

There are many interesting directions for future work, including deriving bounds on the asymptotic convergence of our approximate sampler (Pillai and Smith, 2014; Dinh et al., 2017), considering different likelihood approximation update rules such as *power EP* (Minka, 2004), and extending our idea of approximately integrating out variables to other samplers. For the analysis, Dehaene and Barthelmé (2018) showed that EP with Gaussian approximations is exact in the large data limit; one could extend these results to consider the case of data being allocated amongst *multiple* clusters. Another interesting direction is to explore our EP-based approximate collapsing within the context of variational inference, possibly extending the set of models for which collapsed variational Bayes (Teh et al., 2007) is possible. Finally, there are many ways in which our algorithm could be made even more scalable through distributed, asynchronous implementations, such as in (Ahmed et al., 2012).

# Chapter 6
# CONCLUSIONS AND FUTURE WORK

In this dissertation, we developed scalable methods for learning latent state sequence models. To do so, we developed methods that trade accuracy for scalability:

- We developed buffered stochastic gradient estimators for learning long state space models (Chapter 3).

- We developed an adaptive truncated backpropagation algorithm for learning long recurrent neural networks (Chapter 4).

- We developed an approximate collapsed Gibbs sampler using expectation propagation for clustering high-dimensional state space models and other mixture models (Chapter 5).

To analyze the effect of the error on learning, we developed new theory for stochastic gradient methods with biased gradients:

- We developed error bounds for the finite sample error for stochastic gradient MCMC with biased gradients (Theorem 1).

- We developed bounds on the convergence rate of smooth non-convex optimization for stochastic gradient descent with biased gradient (Theorem 8).

And we showed both theoretically and experimentally that our proposed methods were much more scalable than existing approaches, while still maintaining competitive accuracy.

**Future Work** There are many interesting directions for future work.

- One could generalize the error analysis and convergence guarantees of vanilla stochastic gradient methods with biased stochastic gradients to stochastic gradient methods with acceleration or augmented dynamics.

- One could generalize the applications of our buffered stochastic gradient estimators to learning other latent state models such as variational auto-encoders or spatial processes.

- One could relax the log-concave restriction of the contraction bound (Theorem 5 for exponential decay of bias in buffer size), generalizing the class of nonlinear SSMs with guarantees.

- One could apply the error analysis of biased stochastic gradient methods to other examples of biased stochastic gradients such as in stochastic gradients for continuous-time process and non-contiguous (downsampled) subsequences.

- The buffered stochastic gradient estimators could be improved by adaptively adjusting the buffer size during training similar to the adaptive truncated backpropagation algorithm.

- The adaptive truncated backpropagation estimators could be improved by better estimating the decay rate using specialized regression techniques and adjusting the norm for higher dimensional hidden states.

- The approximate collapsed Gibbs sampler could be improved using stochastic EP for memory savings and adapting its error analysis.

# BIBLIOGRAPHY

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

Amr Ahmed, Mohamed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander Smola. Scalable inference in latent variable models. In *International conference on Web Search and Data Mining*, volume 51, pages 1257–1264, 2012.

Christopher Aicher and Emily B Fox. Scalable clustering of correlated time series using expectation propagation. *SIGKDD Workshop on MiLeTS*, 2016.

Christopher Aicher and Emily B Fox. Approximate collapsed Gibbs clustering with expectation propagation. *arXiv preprint arXiv:1807.07621*, 7 2018.

Christopher Aicher, Nicholas J Foti, and Emily B Fox. Adaptively truncating backpropagation through time to control gradient bias. *Uncertainty in Artificial Intelligence*, 5 2019a.

Christopher Aicher, Yi-An Ma, Nicholas J Foti, and Emily B Fox. Stochastic gradient MCMC for state space models. *SIAM Journal on Mathematics of Data Science*, 1(3): 555–587, 2019b.

Christopher Aicher, Srshti Putcha, Christopher Nemeth, Paul Fearnhead, and Emily B Fox. Stochastic gradient MCMC for nonlinear state space models. *arXiv preprint arXiv:1901.10568*, 1 2019c.

Sivan Aldor-Noiman, Lawrence D Brown, Emily B Fox, and Robert A Stine. Spatio-temporal low count processes with application to violent crime events. *Statistica Sinica*, pages 1587–1610, 2016.

David F Andrews and Colin L Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society: Series B*, pages 99–102, 1974.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Control variates for stochastic gradient MCMC. *Statistics and Computing*, pages 1–17, 2017.

Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Large-scale stochastic sampling from the probability simplex. In *Advances in Neural Information Processing Systems*, pages 6722–6732, 2018.

David Barber, Ali Taylan Cemgil, and Silvia Chiappa. Inference and estimation in probabilistic time series models. *Bayesian Time Series Models*, 2011.

Matthew James Beal et al. *Variational Algorithms for Approximate Bayesian Inference*. university of London London, 2003.

Francois Belletti, Alex Beutel, Sagar Jain, and Ed Chi. Factorized recurrent neural architectures for longer range dependence. In *International Conference on Artificial Intelligence and Statistics*, pages 1522–1530, 2018.

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

David Beniaguev. Historical hourly weather data 2012-2017. `https://www.kaggle.com/selfishgene/historical-hourly-weather-data`, 2017.

Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307 – 327, 1986. ISSN 0304-4076.

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61, 2010.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

Chris K Carter and Robert Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

Hock Peng Chan, Chiang-Wee Heng, and Ajay Jasra. Theory of segmented particle filters. *Advances in Applied Probability*, 48(1):69–87, 2016.

Niladri S Chatterji, Nicolas Flammarion, Yi-An Ma, Peter L Bartlett, and Michael I Jordan. On the theory of variance reduction for stochastic gradient Monte Carlo. In *Proceedings of the 35th International Conference on Machine Learning*, pages 764–773, 2018.

Changyou Chen, Nan Ding, and Lawrence Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pages 2278–2286, 2015a.

Jie Chen and Ronny Luss. Stochastic gradient descent with biased but consistent gradient estimators. *arXiv preprint arXiv:1807.11880*, 2018.

Tianqi Chen, Emily B Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pages 1683–1691, 2014.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015b.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Bertrand Cloez, Martin Hairer, et al. Exponential ergodicity for Markov processes with random switching. *Bernoulli*, 21(1):505–536, 2015.

Maria Colombo, Alessio Figalli, and Yash Jhaveri. Lipschitz changes of variables between perturbations of log-concave measures. *arXiv preprint arXiv:1510.03687*, 2015.

Johan Dahlin, Fredrik Lindsten, and Thomas B Schön. Particle Metropolis–Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015.

Arnak S Dalalyan and Avetik G Karagulyan. User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 2019.

Paul Damlen, John Wakefield, and Stephen Walker. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B*, 61(2):331–344, 1999.

Kathryn A Davis, Hoameng Ung, Drausin Wulsin, Joost Wagenaar, Emily B Fox, Ned Patterson, Charles Vite, Gregory Worrell, and Brian Litt. Mining continuous intracranial EEG in focal canine epilepsy: Relating interictal bursts to seizure onsets. *Epilepsia*, 57 (1):89–98, 2016.

Guillaume Dehaene and Simon Barthelmé. Expectation propagation in the large data limit. *Journal of the Royal Statistical Society: Series B*, 80(1):199–217, 2018.

Pierre Del Moral, Arnaud Doucet, and Sumeetpal Singh. Forward smoothing using sequential Monte Carlo. *arXiv preprint arXiv:1012.5390*, 2010.

Pierre Del Moral, Ajay Jasra, and Yan Zhou. Biased online parameter inference for state-space models. *Methodology and Computing in Applied Probability*, 19(3):727–749, 2017.

Persi Diaconis and David Freedman. Iterated random functions. *SIAM Review*, 41(1):45–76, 1999.

Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems*, pages 3203–3211, 2014.

Vu Dinh, Ann E Rundell, and Gregery T Buzzard. Convergence of Griddy Gibbs sampling and other perturbed Markov chains. *Journal of Statistical Computation and Simulation*, 87(7):1379–1400, 2017.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12, 2011.

Alain Durmus and Éric Moulines. Quantitative bounds of convergence for geometrically ergodic Markov chain in the Wasserstein distance with application to the Metropolis adjusted Langevin algorithm. *Statistics and Computing*, 25(1):5–19, 2015.

Sean R Eddy. Profile hidden Markov models. *Bioinformatics (Oxford, England)*, 14(9): 755–763, 1998.

Wentao Fan and Nizar Bouguila. Non-Gaussian data clustering via expectation propagation learning of finite Dirichlet mixture models and applications. *Neural Processing Letters*, 39 (2):115–135, 2014.

Paul Fearnhead and Hans R Künsch. Particle filters and data assimilation. *Annual Review of Statistics and Its Application*, 5(1):421–449, 2018.

Nick Foti, Jason Xu, Dillon Laird, and Emily B Fox. Stochastic variational inference for hidden Markov models. In *Advances in Neural Information Processing Systems*, pages 3599–3607, 2014.

Emily B Fox. *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*. PhD thesis, Massachusetts Institute of Technology, 2009.

Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. Bayesian nonparametric inference of switching dynamic linear models. *IEEE Transactions on Signal Processing*, 59, 2011.

Andrew Gelman, John B Carlin, Donald B Rubin, Aki Vehtari, David B Dunson, and Hal S Stern. *Bayesian Data Analysis*. CRC Press, 2013.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 73(2):123–214, 2011.

Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Residual splash for optimally parallelizing belief propagation. In *Artificial Intelligence and Statistics*, pages 177–184, 2009.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

Charles AE Goodhart and Maureen O'Hara. High frequency data in financial markets: Issues and applications. *Journal of Empirical Finance*, 4(2-3):73–114, 1997.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113, 1993.

Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1292–1301. JMLR, 2017.

James Douglas Hamilton. *Time Series Analysis*, volume 2. Princeton university press Princeton, NJ, 1994.

Leonard Hasenclever, Stefan Webb, Thibaut Lienart, Sebastian Vollmer, Balaji Lakshminarayanan, Charles Blundell, and Yee Whye Teh. Distributed Bayesian learning with stochastic natural gradient expectation propagation and the posterior server. *The Journal of Machine Learning Research*, 18(1):3744–3780, 2017.

Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 216–223. Morgan Kaufmann Publishers Inc, 2002.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In *International Conference on Machine Learning*, pages 1733–1741, 2017.

Li Jing, Caglar Gulcehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljacic, and Yoshua Bengio. Gated orthogonal recurrent units: On learning to forget. *Neural computation*, 31(4):765–783, 2019.

Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

Gregor Kastner. Dealing with stochastic volatility in time series using the R package stochvol. *Journal of Statistical Software*, 69(5):1–30, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Genshiro Kitagawa and Seisho Sato. Monte Carlo smoothing and self-organising state-space model. In *Sequential Monte Carlo Methods in Practice*, pages 177–195. Springer, 2001.

Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *AAAI*, pages 2101–2109, 2017.

Thomas Laurent and James von Brecht. A recurrent neural network without chaos. In *International Conference on Learning Representations*, 2017.

Francçois Le Gland and Laurent Mevel. Exponential forgetting and geometric ergodicity in hidden Markov models. *Mathematics of Control, Signals and Systems*, 13(1):63–93, 2000.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *AT&T Labs*, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

Tao Lei, Yu Zhang, and Yoav Artzi. Training RNNs as fast as CNNs. *arXiv preprint arXiv:1709.02755*, 2017.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th Symposium on Operating Systems Design and Implementation*, 2014.

Chuanhai Liu. Bayesian robust multivariate linear regression with incomplete data. *Journal of the American Statistical Association*, 91(435):1219–1227, 1996.

Jun S Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Science & Business Media, 2008.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pages 276–284, 2016.

Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer Science & Business Media, 2005.

Yi-An Ma, Tianqi Chen, and Emily B Fox. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

Yi-An Ma, Nicholas J Foti, and Emily B Fox. Stochastic gradient MCMC methods for hidden Markov models. In *International Conference on Machine Learning*, pages 2265–2274, 2017.

Neal Madras, Deniz Sezer, et al. Quantitative bounds for Markov chain convergence: Wasserstein and total variation distances. *Bernoulli*, 16(3):882–908, 2010.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 1993.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *International Speech Communication Association*, 2010.

John Miller and Moritz Hardt. Stable recurrent models. In *International Conference on Learning Representations*, 2019.

Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc, 2001.

Thomas P Minka. Power EP. Technical report, Technical Report, Microsoft Research, Cambridge, 2004.

Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random Kronecker factors. In *Advances in Neural Information Processing Systems*, pages 6594–6603, 2018.

Tigran Nagapetyan, Andrew B Duncan, Leonard Hasenclever, Sebastian J Vollmer, Lukasz Szpruch, and Konstantinos Zygalakis. The true cost of stochastic gradient Langevin dynamics. *arXiv preprint arXiv:1706.02692*, 2017.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

Christopher Nemeth and Paul Fearnhead. Stochastic gradient Markov chain Monte Carlo. *arXiv preprint arXiv:1907.06986*, 2019.

Christopher Nemeth, Paul Fearnhead, and Lyudmila Mihaylova. Particle approximations of the score and observed information matrix for parameter estimation in state space models with linear computational cost. *Journal of Computational and Graphical Statistics*, 25(4): 1138–1157, 2016.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2013.

Jimmy Olsson and Johan Westerborn. Efficient particle-based online smoothing in general hidden Markov models: the PaRIS algorithm. *Bernoulli*, 23(3):1951–1996, 2017.

Jimmy Olsson, Olivier Cappé, Randal Douc, Eric Moulines, et al. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179, 2008.

Konstantina Palla, Zoubin Ghahramani, and David A Knowles. A nonparametric variable clustering model. In *Advances in Neural Information Processing Systems*, 2012.

Konstantina Palla, David A Knowles, and Zoubin Ghahramani. A reversible infinite HMM using normalised random measures. In *International Conference on Machine Learning*, 2014.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

Sam Patterson and Yee Whye Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*, pages 3102–3110, 2013.

David Peel and Geoffrey J McLachlan. Robust mixture modelling using the t-distribution. *Statistics and Computing*, 10(4):339–348, 2000.

Natesh S Pillai and Aaron Smith. Ergodicity of approximate MCMC chains with applications to large data sets. *arXiv preprint arXiv:1405.0182*, 2014.

B T Poljak and Ya Z Tsypkin. Pseudogradient adaptation and training algorithms. *Automation and Remote Control*, 34:45–67, 1973.

Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image processing*, 12(11):1338–1351, 2003.

George Poyiadjis, Arnaud Doucet, and Sumeetpal S Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pages 314–323, 2016.

You Ren, Emily B Fox, Andrew Bruce, et al. Clustering correlated, sparse data streams to estimate a localized housing price index. *The Annals of Applied Statistics*, 11(2):808–839, 2017.

Christian Ritter and Martin A Tanner. Facilitating the Gibbs sampler: the Gibbs stopper and the Griddy-Gibbs sampler. *Journal of the American Statistical Association*, 87(419): 861–868, 1992.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B*, 60(1):255–268, 1998.

Gareth O Roberts, Richard L Tweedie, et al. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

Jacob K Rosenstein, Siddharth Ramakrishnan, Jared Roseman, and Kenneth L Shepard. Single ion channel recordings with CMOS-anchored lipid membranes. *Nano Letters*, 13 (6):2682–2686, 2013.

Daniel Rudolf and Nikolaus Schweizer. Perturbation theory for Markov chains via Wasserstein distance. *Bernoulli*, 24(4A):2610–2639, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Adrien Saumard and Jon A Wellner. Log-concavity and strong log-concavity: a review. *Statistics surveys*, 8:45, 2014.

Steven L Scott. Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351, 2002.

Matthias Seeger. Expectation propagation for exponential families. Technical report, 2005.

Neil Shephard. *Stochastic Volatility: Selected Readings*. Oxford University Press, 2005.

Erik B Sudderth, Alexander T Ihler, Michael Isard, William T Freeman, and Alan S Willsky. Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103, 2010.

Ilya Sutskever. *Training Recurrent Neural Networks*. University of Toronto Toronto, Ontario, Canada, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

Markus Svensén and Christopher M Bishop. Robust Bayesian mixture modelling. *Neurocomputing*, 64, 2005.

Yee W Teh, David Newman, and Max Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1353–1360, 2007.

Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17(1): 193–225, 2016.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 RMSProp: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural networks for machine learning*, 4(2):26–31, 2012.

Xin Thomson Tong and Ramon Van Handel. Ergodicity and stability of the conditional distributions of nondegenerate Markov chains. *The Annals of Applied Probability*, pages 1495–1540, 2012.

Nilesh Tripuraneni, Shixiang Shane Gu, Hong Ge, and Zoubin Ghahramani. Particle Gibbs for infinite hidden Markov models. In *Advances in Neural Information Processing Systems*, pages 2386–2394, 2015.

David A Van Dyk and Taeyoung Park. Partially collapsed Gibbs samplers: Theory and methods. *Journal of the American Statistical Association*, 103(482), 2008.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

Sebastian J Vollmer, Konstantinos C Zygalakis, and Yee Whye Teh. Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics. *The Journal of Machine Learning Research*, 17(1):5504–5548, 2016.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pages 3570–3578, 2017.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning*, pages 681–688, 2011.

Paul J Werbos et al. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Paul John Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons, 1994.

Nick Whiteley. Stability properties of some particle filters. *The Annals of Applied Probability*, 23(6):2500–2537, 2013.

Ronald J Williams and David Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, Architectures, and Applications*, 433, 1995.

Drausin F Wulsin. *Bayesian Nonparametric Modeling of Epileptic Events*. University of Pennsylvania, 2013.

Tatiana Xifara, Chris Sherlock, Samuel Livingstone, Simon Byrne, and Mark Girolami. Langevin diffusions and the Metropolis-adjusted Langevin algorithm. *Statistics & Probability Letters*, 91:14–19, 2014.

Felix X-F Ye, Yi-an Ma, and Hong Qian. Estimate exponential memory decay in hidden Markov model and its applications. *arXiv preprint arXiv:1710.06078*, 2017.

# Appendix A

# STOCHASTIC GRADIENT MCMC FOR STATE SPACE MODELS

This chapter of the Appendix provides supplementary details for Chapter 3. It is organized as follows. We first present proofs for the theorems of Chapter 3.3. We then present additional details and figures for the experiments of Chapter 3.5.

## *A.1 Error Analysis Proofs*

### *A.1.1 Proof of Theorem 1*

We now prove the error bounds for biased SGLD's finite sample average found in Chapter 3.3.1. The proof is a modification of the proof of Theorem 3 found in Appendix E of Chen et al. (2015a).

We first review the condition the test statistic $\phi$ must satisfy. Let $\mathcal{L}$ be the generator of the Langevin diffusion

$$\mathcal{L}[\psi(\theta_t)] = \nabla U(\theta_t) \cdot \nabla \psi(\theta_t) + \frac{\epsilon^2}{2} \operatorname{tr}(\nabla^2 \psi(\theta_t)) \ .$$

Then, we define $\psi$ to solve the *Poisson equation*

$$\frac{1}{K} \sum_{k=1}^{K} \mathcal{L}[\psi(\theta^{(k)})] = \hat{\phi}_{K,\epsilon} - \bar{\phi} \ , \tag{A.1}$$

where we recall

$$\hat{\phi}_{K,\epsilon} = K^{-1} \sum_{k=1}^{K} \phi(\theta^{(k)}) \quad \text{and} \quad \bar{\phi} = \mathbb{E}_{\,p(\theta|y)}[\phi(\theta)] \ .$$

Our assumption on $\phi$ is that $\psi(\theta)$ and its derivatives are bounded by some finite constant $M$. This is the implicit moment condition for $\phi$, which is also assumed by Vollmer et al. (2016) and Chen et al. (2015a).

The proof of Theorem 1 then proceeds as in Theorem 3 of (Chen et al., 2015a), except that we allow for a $\delta > 0$ such that $\mathbb{E}\,\|\hat{g}(\theta) - g(\theta)\| \leq \delta$ for all $\theta$ rather than restrict $\delta = 0$.

For compactness of notation, we will use $g_k$ to denote $g(\theta^{(k)})$, $\hat{g}_k$ to denote $\hat{g}(\theta^{(k)})$, and $\psi_k$ to denote $\psi(\theta^{(k)})$.

*Proof of Theorem 1.* Following Chen et al. (2015a), from the definition of the functional $\psi$ and generator $\mathcal{L}$, we have

$$\hat{\phi}_{K,\epsilon} - \bar{\phi} = \frac{\mathbb{E}\,\psi_K - \psi_1}{K\epsilon} - \frac{\sum_{k=1}^{K}(\mathbb{E}\,\psi_k - \psi_k)}{K\epsilon} + \frac{\sum_{k=1}^{K}(\hat{g}_k - g_k)\cdot\nabla\psi_k}{K} + \mathcal{O}(\epsilon)\ , \tag{A.2}$$

and we also have $\mathbb{E}\,\psi_K - \mathbb{E}\,\psi_1$ is $\mathcal{O}(1)$, $\mathbb{E}\,(\mathbb{E}\,\psi_K - \psi_1)^2$ is $\mathcal{O}(1)$, and $\mathbb{E}\,(\mathbb{E}\,\psi_k - \psi_k)^2$ is $\mathcal{O}(\epsilon)$.

Let $\xi_k = (\hat{g}_k - g_k)\cdot\nabla\psi_k$. From our assumptions on the bias and MSE of $\hat{g}$ and as $\nabla\psi$ is bounded, we have $|\mathbb{E}\,\xi_k| \leq M\delta$ and $\mathbb{E}\,[\xi_k^2] \leq M^2\sigma^2$ for all $k$. In addition, we have for all $k \neq k'$

$$|\mathbb{E}\,[\xi_k\xi_{k'}]| \leq M^2\|\mathbb{E}\,[\hat{g}_k - g_k]\|\|\mathbb{E}\,[\hat{g}_{k'} - g_{k'}]\| \leq M^2\delta^2\ , \tag{A.3}$$

where the expectations are over *independent* stochastic subsequences $\mathcal{S}$ chosen at steps $k$ and $k'$.

To prove the bias bound, we take the expectation of Eq. (A.2) and bound each term

$$|\mathbb{E}\,\hat{\phi}_{K,\epsilon} - \bar{\phi}| = \mathcal{O}\left(\frac{1}{K\epsilon} + \delta + \epsilon\right) \tag{A.4}$$

To prove the MSE bound, we take the square and expectation of both sides of Eq. (A.2),

$$\begin{aligned}
&\mathbb{E}\,(\hat{\phi}_{K,\epsilon} - \bar{\phi}\,)^2 \leq \\
&\quad\mathbb{E}\left[\frac{(\mathbb{E}\,\psi_K - \psi_0))^2}{K^2\epsilon^2} + \frac{\sum_{k=1}^{K}(\mathbb{E}\,\phi_k - \psi_k)^2}{K^2\epsilon^2}\right. \\
&\qquad + \frac{\sum_{k=1}^{K}\xi_k^2 + \sum_{k\neq k'=1}^{K}\xi_k\xi_{k'}}{K^2} + \mathcal{O}(\epsilon^2) \\
&\qquad \left. + \frac{\sum_{k=1}^{K}\xi_k}{K}\cdot\left(\frac{\mathbb{E}\,\psi_K - \psi_1}{K\epsilon} - \frac{\sum_{k=1}^{K}(\mathbb{E}\,\psi_k - \psi_k)}{K\epsilon} + \mathcal{O}\,(\epsilon)\right)\right]
\end{aligned} \tag{A.5}$$

The first two lines are the squared terms and the last two lines are the cross terms that do not go to zero. In particular, we do not assume $\hat{g}(\theta)$ is unbiased for $g(\theta)$, therefore we keep the cross-terms involving $\xi_k$. Bounding each term of Eq. (A.5) gives the MSE bound

$$\begin{aligned}
&\mathbb{E}\,(\hat{\phi}_{K,\epsilon} - \bar{\phi}\,)^2 \\
&\quad \leq \frac{\mathcal{O}(1)}{K^2\epsilon^2} + \frac{\mathcal{O}(K\epsilon)}{K^2\epsilon^2} + \frac{\mathcal{O}(K\sigma^2) + \mathcal{O}(K^2\delta^2)}{K^2} + \mathcal{O}(\epsilon^2) \\
&\qquad\quad + \mathcal{O}(\delta) \cdot \left( \frac{\mathcal{O}(1)}{K\epsilon} + \frac{\mathcal{O}(K)}{K\epsilon} + \mathcal{O}\left(\epsilon\right) \right) \\
&\quad \leq \mathcal{O}\left( \frac{1}{K\epsilon} + \frac{\sigma^2}{K} + \epsilon^2 + \delta^2 + \frac{\delta}{\epsilon} + \delta\epsilon \right) \ . \qquad \text{(A.6)}
\end{aligned}$$

□

### *A.1.2 Proof of Theorems 2 and 6*

We now prove Theorems 2 and 6, which bounds the bias and MSE of our stochastic gradients $g_{S,B}$ and $g^{\mathrm{PF}}_{S,B,N}$ respectively. In our proof, we use Lemma 4 to bound the subsequence error which is proved in Chapter A.1.2.

*Proof of Theorems 2 and 6.* For the bias bound, (3.18), we apply the triangle inequality to decompose the error into three terms

$$\|g^{\mathrm{PF}}_{S,B,N}(\theta) - g(\theta)\| \leq \underbrace{\|g^{\mathrm{PF}}_{S,B,N}(\theta) - \hat{g}_{S,B}(\theta)\|}_{\text{particle error (I)}} + \underbrace{\|\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\|}_{\text{buffering error (II)}} + \underbrace{\|\hat{g}_{S,T}(\theta) - g(\theta)\|}_{\text{subsequence error (III)}}, \quad \text{(A.7)}$$

where expectations are over the random subsequence $\mathcal{S}$ and particles. Each term is bounded separately (recalling that $\gamma = \max_t \Pr(t \in \mathcal{S})^{-1}$)

(I) *Particle bias*: from Eq. 3.15 of (Kantas et al., 2015), the bias of our particle smoother is $\mathcal{O}(\gamma\frac{S+2B}{N})$.

(II) *Buffering bias*: from (Aicher et al., 2019b), we know there exists a finite constant $C_1 < \infty$ that is independent of $T, S, B, N$, such that

$$\mathbb{E}\,\|\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\| \leq \gamma \cdot C_1 \cdot (L_\theta)^B \ . \qquad \text{(A.8)}$$

Thus, the buffering bias can be upper bounded using Jensen's inequality

$$\|\mathbb{E}\left(\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\right)\| \leq \mathbb{E}\,\|\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\| \leq \gamma \cdot C_1 \cdot (L_\theta)^B \ . \tag{A.9}$$

(III) *Subsequence bias*: the subsequence bias is zero as $\mathbb{E}\,\hat{g}_{S,T}(\theta) = g(\theta)$.

Applying these bounds gives us the bias bound

$$\|\mathbb{E}\,g^{\mathrm{PF}}_{S,B,N}(\theta) - g(\theta)\| \leq \gamma \cdot \left[C_1(L_\theta)^B + \mathcal{O}\left(\frac{S+2B}{N}\right)\right] . \tag{A.10}$$

For the MSE bound, (3.19), we again apply the triangle inequality and recall that $2XY \leq X^2 + Y^2$ implies $(X+Y+Z)^2 \leq 3(X^2+Y^2+Z^2)$ to decompose the error into three terms

$$\mathbb{E}\,\|g^{\mathrm{PF}}_{S,B,N} - g\|^2 \leq 3\underbrace{\mathbb{E}\,\|g^{\mathrm{PF}}_{S,B,N} - \hat{g}_{S,B}\|^2}_{\text{particle MSE (I)}} + 3\underbrace{\mathbb{E}\,\|\hat{g}_{S,B} - \hat{g}_{S,T}\|^2}_{\text{buffering MSE (II)}} + 3\underbrace{\mathbb{E}\,\|\hat{g}_{S,T} - g\|^2}_{\text{subsequence MSE (III)}} \ ,$$

where expectations are over the random subsequence $\mathcal{S}$ and particles. Again, each term is bounded separately,

(I) *Particle MSE*: from Eq. 3.15 of (Kantas et al., 2015), the MSE of our particle smoother is bounded by $\mathcal{O}(\gamma^2 \frac{(S+2B)^2}{N})$.

(II) *Buffering MSE*: from (Aicher et al., 2019b), the buffering MSE is bounded

$$\mathbb{E}\,\|\hat{g}_{S,B}(\theta) - \hat{g}_{S,T}(\theta)\|^2 \leq \gamma^2 \cdot C_1^2 \cdot (L_\theta)^{2B} \ . \tag{A.11}$$

(III) *Subsequence MSE*: from Lemma 4, there exists a constant $C_2 < \infty$ independent of $T, S, B, N$ such that

$$\mathbb{E}\,\|\hat{g}_{S,T}(\theta) - g(\theta)\|^2 \leq \gamma^2 \cdot C_2 \cdot S \ . \tag{A.12}$$

Combining these bounds gives us the MSE bound

$$\mathbb{E}\,\|g^{\mathrm{PF}}_\theta(S,B,N) - g_\theta\|^2 \leq 3\gamma^2 \cdot \left[C_1^2(L_\theta)^{2B} + C_2 S + \mathcal{O}\left(\frac{(S+2B)^2}{N}\right)\right] \ . \tag{A.13}$$

□

*Stochastic Subsequence MSE*

For the proof of Theorem 2, we bound the MSE between the full gradient $g(\theta)$ and the unbiased stochastic gradient estimate $\hat{g}_{S,T}(\theta)$, specifically for the case of randomly sampling a *contiguous* subsequence $\mathcal{S}$. Because $\hat{g}_{S,T}(\theta)$ is unbiased for $g(\theta)$, this reduces to calculating the variance of $\hat{g}_{S,T}(\theta)$ with respect to the sampling distribution of the subsequence $\mathcal{S}$.

Let $f_t$ denote the $t$-th gradient term in Fisher's identity

$$f_t = \mathbb{E}_{x_{1:T}|y_{1:T},\theta}[\nabla \log p(y_t, x_t \,|\, x_{t-1}, \theta)] \tag{A.14}$$

Therefore

$$g(\theta) = \sum_{t=1}^{T} f_t \quad \text{and} \quad \hat{g}_{S,T}(\theta) = \sum_{t \in \mathcal{S}} \Pr(t \in \mathcal{S})^{-1} \cdot f_t$$

We now present the lemma that bounds the variance of $\hat{g}_\theta(S, T)$, under the assumption that the autocorrelation of $f_t$ decays geometrically $|\operatorname{Corr}(f_t, f_{t+s}| \leq \rho^s$.

**Lemma 4.** *If for all $t$, the variance of $f_t$ is bounded and the autocorrelation of $f_t$ is geometrically bounded, then there exists a constant $C_2 < \infty$ (not dependent on $T, S, B, N$) such that*

$$\operatorname{Var}(\hat{g}_{S,T}(\theta)) \leq \gamma^2 \cdot C_2 \cdot S \ . \tag{A.15}$$

The assumption that the autocorrelation of $f_t$ decays geometrically is reasonable when both the observations $Y_{1:T}$ and the posterior latent states $X_{1:T}|Y_{1:T}$ are *ergodic* (i.e. exhibit an exponential forgetting property) (Cappé et al., 2005).

We now present the proof.

*Proof of Lemma 4.* Let $V < \infty$ be a bound on the variance of $f_t$ for all $t$ (i.e. $\operatorname{Var}(f_t) \leq V$). Let $\rho \in [0, 1)$ be a bound on the geometric decay of the autocorrelation of $f_t$. Then we have $|\operatorname{Corr}(f_t, f_{t+s})| \leq \rho^s$ for all $t$ and $s \in \mathbb{N}$, Together these bounds imply a bound on the covariance between any $f_t$ and $f_{t+s}$

$$\operatorname{Cov}(f_t, f_{t+s}) \leq |\operatorname{Corr}(f_t, f_{t+s})| \cdot \sqrt{\operatorname{Var}(f_t) \cdot \operatorname{Var}(f_{t+s})} \leq V\rho^s \ . \tag{A.16}$$

Then we have

$$\begin{aligned}
\mathrm{Var}(\hat{g}_{S,T}(\theta)) &\leq \gamma^2 \cdot \mathrm{Var}\left[\sum_{t\in\mathcal{S}} f_t\right] \\
&= \gamma^2 \cdot \left[\sum_{t\in\mathcal{S}} \mathrm{Var}(f_t) + \sum_{t\neq t'\in\mathcal{S}} \mathrm{Cov}(f_t, f_{t'})\right], \\
&\leq \gamma^2 \cdot \left[S \cdot V + \sum_{s=1}^{S-1} 2(S-s)\cdot V\rho^s\right] \\
&= \gamma^2 \cdot S \cdot \left[V + 2V\sum_{s=1}^{S-1}(1-s/S)\cdot\rho^s\right] \\
&\leq \gamma^2 \cdot S \cdot \left[2V\sum_{s=0}^{S-1}\rho^s\right] \\
&\leq \gamma^2 \cdot S \cdot 2V/(1-\rho) \qquad \text{(A.17)}
\end{aligned}$$

As $S \geq 1$, if $C_2 = 2V/(1-\rho)$, we have

$$\mathbb{E}\,\|\hat{g}_{S,T}(\theta) - g(\theta)\|^2 = \mathrm{Var}(\hat{g}_{S,T}(\theta)) \leq \gamma^2 \cdot S \cdot C_2 \ . \qquad \text{(A.18)}$$

□

### *A.1.3 Proof of Theorems 3 and 4*

First, we show how to bound the error in $\bar{g} = \hat{g}_{S,T}, \tilde{g} = \hat{g}_{S,B}$ in terms of Wasserstein distances between $\gamma, \widetilde{\gamma}$. Second, we show these Wasserstein distances decay geometrically in $B$. Finally, we prove Theorems 3 and 4. To keep the presentation clean, we leave proofs of Lemmas until the end.

*Functional Bound in terms of Wasserstein*

We connect the error $\|\bar{g} - \tilde{g}\|_2$ to the Wasserstein distances between $\gamma, \widetilde{\gamma}$, by applying this duality formula Eq. (2.30) to the difference of Eqs. (3.3) and (3.5)

$$\bar{g}(\theta) - \tilde{g}(\theta) = \frac{T}{S}\sum_{t\in\mathcal{S}} \mathbb{E}_{\gamma_{t-1:t}}\left[\nabla \log p(y_t, u_t|u_{t-1}, \theta)\right] - \mathbb{E}_{\widetilde{\gamma}_{t-1:t}}\left[\nabla \log p(y_t, u_t|u_{t-1}, \theta)\right]. \qquad \text{(A.19)}$$

Applying the triangle inequality gives Lemma 5.

**Lemma 5.** *If $\nabla \log p(y_t, u_t|u_{t-1}, \theta)$ are Lipschitz in $u_{t-1:t}$ with constant $C_\theta$,*

$$\|\bar{g}(\theta) - \tilde{g}(\theta)\|_2 \leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t \in \mathcal{S}} \mathcal{W}_1(\gamma_{t-1:t}, \widetilde{\gamma}_{t-1:t}). \tag{A.20}$$

If $\nabla \log p(y_t, u_t \,|\, u_{t-1}, \theta)$ is not Lipschitz in $u_{t-1:t}$, but is Lipschitz in $u_{t-1:t}u_{t-1:t}^T$ (as in LGSSMs), then the following Lemma lets us bound the 1-Wasserstein distance of $uu^T$ in terms of the 2-Wasserstein distance of $u$.

**Lemma 6.** *Let $\gamma'$ be the distribution of $uu^T$. Let $\widetilde{\gamma}'$ be the distribution of $\widetilde{u}\widetilde{u}^T$. Let $M = \mathbb{E}_\gamma[\|u\|_2^2] < \infty$. (Note $\mathcal{W}_2(\gamma, \widetilde{\gamma}) < \infty$ implies $\mathbb{E}_\gamma[\|u\|_2^2] < \infty$.) Then,*

$$\mathcal{W}_1(\gamma', \widetilde{\gamma}') \leq (2\sqrt{M} + 1) \cdot \max\left\{\mathcal{W}_2(\gamma, \widetilde{\gamma})^{1/2}, \mathcal{W}_2(\gamma, \widetilde{\gamma})\right\} \quad .$$

*Geometric Wasserstein Decay*

We first review why contractive random maps induce Wasserstein bounds. If two distributions $\gamma_t, \gamma_t'$ have identically distributed random maps $\psi_t, \psi_t'$, that is there exists a random function $\psi_t$ satisfying

$$u \sim \gamma_t \text{ and } u' \sim \gamma_t' \;\; \Rightarrow \;\; \psi_t(u) \sim \gamma_{t+1} \text{ and } \psi_t(u') \sim \gamma_{t+1}' \;\; , \tag{A.21}$$

then we can bound the Wasserstein distance of $\gamma_{t+1}, \gamma_{t+1}'$ in terms of the Wasserstein distance of $\gamma_t, \gamma_t'$ given a bound on the random map's Lipschitz constant $\|\psi_t\|_{Lip} < L$

$$\begin{aligned} \mathcal{W}_p(\gamma_{t+1}, \gamma_{t+1}')^p &= \inf_{\xi_{t+1}} \int \|u_{t+1} - u_{t+1}'\|_2^p \, d\xi_{t+1}(u_{t+1}, u_{t+1}') \\ &\leq \inf_{\xi_t} \int \|\psi_t(u_t) - \psi_t(u_t')\|_2^p \, d\xi_t(u_t, u_t') d\psi_t \\ &\leq \inf_{\xi_t} \int L^p \cdot \|u_t - u_t'\|_2^p \, d\xi_t(u_t, u_t') = L^p \cdot \mathcal{W}_p(\gamma_t, \gamma_t')^p \quad . \end{aligned} \tag{2.32}$$

Unfortunately for SSMs, Eq. (2.32) does not apply as the random maps $\vec{\psi}_t, \overleftarrow{\psi}_t$ of $\gamma$ and $\widetilde{f}_t, \widetilde{b}_t$ of $\widetilde{\gamma}$ are *not identically* distributed. To see this, we first review the conditional

probability distributions used to define $\vec{\psi}_t, \overleftarrow{\psi}_t$. The forward random map $\vec{\psi}_t$ draws $u_{t+1} \,|\, u_t$ from the *forward smoothing kernel*

$$\mathcal{F}_t(u_{t+1} \,|\, u_t) := p(u_{t+1} \,|\, u_t, y_{>t}) = p(u_{t+1} \,|\, u_t) p(y_{t+1} \,|\, u_{t+1}) \beta_{t+1}(u_{t+1}) / \beta_t(u_t) \tag{A.22}$$

and the backward random map $\overleftarrow{\psi}_t$ draws $u_{t-1} \,|\, u_t$ from the *backward smoothing kernel*

$$\mathcal{B}_t(u_{t-1} \,|\, u_t) := p(u_{t-1} \,|\, u_t, y_{\geq t}) = p(u_t \,|\, u_{t-1}) p(y_t \,|\, u_t) \alpha_{t-1}(u_{t-1}) / \alpha_t(u_t) \ . \tag{A.23}$$

Because $\widetilde{\gamma}$ uses different forward and backward messages $\widetilde{\alpha}$, $\widetilde{\beta}$ in Eqs. (A.22) and (A.23), the kernels $\widetilde{\mathcal{F}}_t, \widetilde{\mathcal{B}}_t$ are not identical to $\mathcal{F}_t, \mathcal{B}_t$ (and the random maps are *not* identically distributed). This is unlike homogeneous Markov chains, where the kernels are identical at each time $t$ (and the random maps are identically distributed).

Instead of connecting $\gamma$ to $\widetilde{\gamma}$ directly, we use the triangle inequality to connect them through an intermediate distribution $\widehat{\gamma} := p(u \,|\, y_{t \geq t_{-B}}, \theta)$

$$\mathcal{W}_p(\gamma, \widetilde{\gamma}) \leq \mathcal{W}_p(\gamma, \widehat{\gamma}) + \mathcal{W}_p(\widehat{\gamma}, \widetilde{\gamma}) \ . \tag{A.24}$$

Introducing this particular intermediate distribution $\widehat{\gamma}$ is the key step for our Wasserstein bounds between $\gamma$ and $\widetilde{\gamma}$. Because $\widehat{\gamma}$ conditions on all $y_t$ after $y_{\mathcal{S}^*}$, $\widehat{\gamma}$ and $\gamma$ have identical backward messages $\beta_t$ and therefore identically distributed forward random maps $\vec{\psi}_t$. Similarly, because $\widehat{\gamma}$ does not condition on $y_t$ before $y_{\mathcal{S}^*}$, $\widehat{\gamma}$ and $\widetilde{\gamma}$ have identical forward messages $\widetilde{\alpha}_t$ and identically distributed backward random maps $\widetilde{b}_t$.

Therefore, we can bound $\mathcal{W}_p(\gamma, \widehat{\gamma})$ using $\vec{\psi}_t$ and bound $\mathcal{W}_p(\widehat{\gamma}, \widetilde{\gamma})$ using $\widetilde{b}_t$ with the contraction trick Eq. (2.32) giving us Lemma 7.

**Lemma 7.** *If there exists $L_\theta < 1$ such that for all $t \in \mathcal{S}^*$, $\|\vec{\psi}_t\|_{Lip} < L_\theta$ and $\|\widetilde{b}_t\|_{Lip} < L_\theta$, then for all $t \in \mathcal{S}$ we have*

$$\begin{aligned} \mathcal{W}_p(\gamma_{t-1:t}, \widehat{\gamma}_{t-1:t}) &\leq (1 + L_\theta^p)^{1/p} \cdot \mathcal{W}_p(\gamma_{t-1}, \widehat{\gamma}_{t-1}) \qquad \text{(A.25)} \\ &\leq (1 + L_\theta^p)^{1/p} \cdot L_\theta^{t-1-t_{-B}} \cdot \mathcal{W}_p(\gamma_{t_{-B}}, \widehat{\gamma}_{t_{-B}}) \\ \mathcal{W}_p(\widehat{\gamma}_{t-1:t}, \widetilde{\gamma}_{t-1:t}) &\leq (1 + L_\theta^p)^{1/p} \cdot \mathcal{W}_p(\widehat{\gamma}_t, \widetilde{\gamma}_t) \qquad \text{(A.26)} \\ &\leq (1 + L_\theta^p)^{1/p} \cdot L_\theta^{t_{S+B}-t} \cdot \mathcal{W}_p(\widehat{\gamma}_{t_{S+B}}, \widetilde{\gamma}_{t_{S+B}}) \end{aligned}$$

*Proof of Theorems*

Putting together the results of the previous two subsections gives us our geometric error bounds: Theorem 3 when the gradient terms are Lipschitz in $u$ and Theorem 4 when the gradient terms are Lipschitz in $uu^T$. Both theorems require the random maps of the forward and backward smoothing kernels are contractions. We first prove Theorem 3.

*Proof of Theorem 3.* Combining Lemmas 5 and 7 with some algebra

$$\begin{aligned}
\|\bar{g}(\theta) - \tilde{g}(\theta)\|_2 &\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t \in \mathcal{S}} \mathcal{W}_1(\gamma_{t-1:t}, \widetilde{\gamma}_{t-1:t}) \\
&\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t \in \mathcal{S}} \mathcal{W}_1(\gamma_{t-1:t}, \widehat{\gamma}_{t-1:t}) + \mathcal{W}_1(\widehat{\gamma}_{t-1:t}, \widetilde{\gamma}_{t-1:t}) \\
&\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t=1}^{S} (1 + L_\theta) L_\theta^{B+t-1} \epsilon_1 + (1 + L_\theta) L_\theta^{B+S-t} \epsilon_1 \\
&\leq T \cdot C_\theta \cdot \frac{1+L}{1-L} \cdot \frac{1 - L^S}{S} \cdot L^B \cdot 2\epsilon_1 \ ,
\end{aligned}$$

where $\max_{\mathcal{S}^* \subset 1:T} \{\mathcal{W}_1(\gamma_{t_-B}, \widehat{\gamma}_{t_-B}), \mathcal{W}_1(\widehat{\gamma}_{t_{S+B}}, \widetilde{\gamma}_{t_{S+B}})\} = \max_{\mathcal{S}^* \subset 1:T} \{\epsilon_\rightarrow, \epsilon_\leftarrow\} = \epsilon_1$. □

We now prove Theorem 4 for when $\nabla \log p(y, u_t \,|\, u_{t-1}\theta)$ is Lipschitz in $uu^T$.

*Proof of Theorem 4.* Applying Lemmas 6 and 7, we have

$$\begin{aligned}
\|\bar{g}(\theta) - \tilde{g}(\theta)\|_2 &\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t \in \mathcal{S}} \max_{r \in \{1/2, 1\}} [\mathcal{W}_2(\gamma_{t-1:t}, \widehat{\gamma}_{t-1:t}) + \mathcal{W}_2(\widehat{\gamma}_{t-1:t}, \widetilde{\gamma}_{t-1:t})]^r \\
&\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t=1}^{S} \max_{r \in \{1/2, 1\}} \left[ (L^{B+t-1} + L^{B+S-t}) \sqrt{1 + L^2} \epsilon_2 \right]^r \\
&\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t=1}^{S} L^{(B + \min\{t-1, S-t\})/2} \cdot \sqrt{1 + L^2} \cdot \max_{r \in \{1/2, 1\}} (2\epsilon_2)^r \\
&\leq \frac{T}{S} \cdot C_\theta \cdot 2 \cdot \frac{1 - L^{S/4}}{1 - L^{1/2}} \cdot L^{B/2} \cdot \sqrt{1 + L^2} \cdot \max_{r \in \{1/2, 1\}} (2\epsilon_2)^r
\end{aligned}$$

□

*Proof of Lemmas in Chapter A.1.3*

We now provide proofs to the Lemmas in section A.1.3.

We first present a proof of Lemma 5 that relates the error in the difference of expectations in Eq. (A.19) to Wasserstein distance.

*Proof of Lemma 5.* Let $g_t(u_{t-1:t}) = \nabla \log p(y_t, u_t \,|\, u_{t-1}, \theta)$.

Recall $\|g_t(u_{t-1:t})\|_{Lip} \leq C_\theta$ for all $t$ by assumption. Then, by the Kantorovich-Rubinstein duality formula Eq. (2.30), we have

$$\left\|\mathbb{E}_{\gamma_{t-1:t}}\left[g_t(u_{t-1:t})\right] - \mathbb{E}_{\widetilde{\gamma}_{t-1:t}}\left[g_t(u_{t-1:t})\right]\right\|_2 \leq C_\theta \cdot \mathcal{W}_1(\gamma_{t-1:t}, \widetilde{\gamma}_{t-1:t}) \ . \tag{A.27}$$

Therefore,

$$\begin{aligned}
\|\bar{g}(\theta) - \tilde{g}(\theta)\|_2 &\leq \left\| \frac{T}{S} \sum_{t \in \mathcal{S}} \mathbb{E}_{\gamma_{t-1:t}}\left[g_t(u_{t-1:t})\right] - \mathbb{E}_{\widetilde{\gamma}_{t-1:t}}\left[\nabla U_t(u_{t-1:t})\right] \right\|_2 && \text{(A.28)} \\
&\leq \frac{T}{S} \sum_{t \in \mathcal{S}} \left\| \mathbb{E}_{\gamma_{t-1:t}}\left[g_t(u_{t-1:t})\right] - \mathbb{E}_{\widetilde{\gamma}_{t-1:t}}\left[\nabla U_t(u_{t-1:t})\right] \right\|_2 && \text{(A.29)} \\
&\leq \frac{T}{S} \cdot C_\theta \cdot \sum_{t \in \mathcal{S}} \mathcal{W}_1(\gamma_{t-1:t}, \widetilde{\gamma}_{t-1:t}) \ . && \text{(A.30)}
\end{aligned}$$

□

We now present the proof of Lemma 6 that relates the 1-Wasserstein distance between distributons $(\gamma', \widetilde{\gamma}')$ of $uu^T$ to the 2-Wasserstein distance between $(\gamma, \widetilde{\gamma})$ over $u$.

*Proof of Lemma 6.* Let $\xi$ be a joint distribution over $u$ and $\widetilde{u}$ with marginals $\gamma$ and $\widetilde{\gamma}$. Let $w := \widetilde{u} - u$, which implies $\widetilde{u} = u + w$.

Then we have

$$\mathbb{E}\,\|\widetilde{u}\widetilde{u}^T - uu^T\|_F = \mathbb{E}\,\|uw^T + wu^T + ww^T\|_F \tag{A.31}$$

$$\leq \mathbb{E}\,\|uw^T\|_F + \mathbb{E}\,\|wu^T\|_F + \mathbb{E}\,\|ww^T\|_F \tag{A.32}$$

$$= 2\mathbb{E}\,|u^T w| + \mathbb{E}\,[\|w\|^2] \tag{A.33}$$

$$\leq 2\sqrt{\mathbb{E}\,[\|u\|^2]\mathbb{E}\,[\|w\|^2]} + \mathbb{E}\,[\|w\|^2] \tag{A.34}$$

$$\leq (2\sqrt{M}+1)\max\{\mathbb{E}\,[\|w\|^2]^{1/2}, \mathbb{E}\,[\|w\|^2]\} \tag{A.35}$$

$$= (2\sqrt{M}+1)\max\{\mathbb{E}\,[\|\widetilde{u}-u\|^2]^{1/2}, \mathbb{E}\,[\|\widetilde{u}-u\|^2]\} \tag{A.36}$$

where we observe $\|xx^T\|_F = \|xx^T\|_2 = \|x\|_2^2 = x^T x$ and we use Cauchy-Schwartz.

Taking the infimum over all $\xi$ gives the result

$$\mathcal{W}_1(\gamma', \widetilde{\gamma}') = \inf_{\xi} \mathbb{E}\,\|\widetilde{u}\widetilde{u}^T - uu^T\|_F \tag{A.37}$$

$$\leq \inf_{\xi} \left[(2\sqrt{M}+1)\max\{\mathbb{E}\,[\|\widetilde{u}-u\|^2]^{1/2}, \mathbb{E}\,[\|\widetilde{u}-u\|^2]\}\right] \tag{A.38}$$

$$= (2\sqrt{M}+1)\max\{\inf_{\xi}\mathbb{E}\,[\|\widetilde{u}-u\|^2]^{1/2}, \inf_{\xi}\mathbb{E}\,[\|\widetilde{u}-u\|^2]\} \tag{A.39}$$

$$= (2\sqrt{M}+1)\cdot \max_{r\in 1,1/2} \mathcal{W}_2(\gamma,\widetilde{\gamma})^r \ . \tag{A.40}$$

□

We now prove Lemma 7 that bounds $\mathcal{W}_p(\gamma_{t-1:t}, \widetilde{\gamma}_{t-1:t})$ in terms of buffer size, if the forward and backward random maps $\vec{\psi}_t, \overleftarrow{\psi}_t$ are Lipschitz.

*Proof of Lemma 7.* We will first prove Eq. (A.25). Recall $\vec{\psi}_t$ is Lipschitz with constant $L_\theta < 1$ for all $t \in \mathcal{S}^*$.

Let $\xi_{t:t+1}$ be a joint distribution over $u_{t:t+1}$ and $\widehat{u}_{t:t+1}$ with marginals $\gamma_{t:t+1}$ and $\widehat{\gamma}_{t:t+1}$. Let $\xi_t$ be a joint distribution over $u_t$ and $\widehat{u}_t$ with marginals $\gamma_t$ and $\widehat{\gamma}_t$. Then for all $t \in \mathcal{S}$, we

have

$$\mathcal{W}_p(\gamma_{t:t+1}, \widehat{\gamma}_{t:t+1})^p \leq \inf_{\xi_{t:t+1}} \int \|u_t - \hat{u}_t\|_2^p + \|u_{t+1} - \hat{u}_{t+1}\|_2^p \, d\xi_{t:t+1}(u_{t:t+1}, \hat{u}_{t:t+1}) \tag{A.41}$$

$$\leq \inf_{\xi_t} \int \|u_t - \hat{u}_t\|_2^p + \|\vec{\psi}_t(u_t) - \vec{\psi}_t(\hat{u}_t)\|_2^p \, d\xi_t(u_t, \hat{u}_t) d\vec{\psi}_t \tag{A.42}$$

$$\leq \inf_{\xi_t} \int \|u_t - \hat{u}_t\|_2^p + L_\theta^p \cdot \|u_t - \hat{u}_t\|_2^p \, d\xi_t(u_t, \hat{u}_t) \tag{A.43}$$

$$\leq (1 + L_\theta^p) \cdot \mathcal{W}_p(\gamma_t, \widehat{\gamma}_t)^p \tag{A.44}$$

Repeatedly applying Eq. (2.32) completes the proof for Eq. (A.25)

$$\mathcal{W}_p(\gamma_{t-1:t}, \widehat{\gamma}_{t-1:t}) \leq (1 + L_\theta^p)^{1/p} \cdot \mathcal{W}_p(\gamma_{t-1}, \widehat{\gamma}_{t-1}) \tag{A.45}$$

$$\leq (1 + L_\theta^p)^{1/p} \cdot L_\theta \cdot \mathcal{W}_p(\gamma_{t-2}, \widehat{\gamma}_{t-2}) \tag{A.46}$$

$$\leq (1 + L_\theta^p)^{1/p} \cdot L_\theta^2 \cdot \mathcal{W}_p(\gamma_{t-3}, \widehat{\gamma}_{t-3}) \tag{A.47}$$

$$\leq \ldots \tag{A.48}$$

$$\leq (1 + L_\theta^p)^{1/p} \cdot L_\theta^{B+t-1} \cdot \mathcal{W}_p(\gamma_{-B}, \widehat{\gamma}_{-B}) \ . \tag{A.49}$$

The proof of Eq. (A.26) is identical. □

### *A.1.4 Proof of Theorem 5*

Theorem 5 states that if the prior distribution for $x_0$, the transition distribution $p(x_t \,|\, x_{t-1}, \theta)$ and the emission distribution $p(y_t \,|\, x_t)$ are log-concave, then we can bound the Lipschitz constant of $\vec{\Psi}_t$ in terms of $\vec{\Psi}_t^{(0)}$ and $\vec{\Psi}_t^{(1)}$.

We first we review *Caffarelli's log-concave perturbation theorem*, the main tool we use in our proof. Then, we present the proof of Theorem 5.

#### *Caffarelli's Perturbation Theorem*

Caffarelli's log-concave perturbation theorem allows us to connect Lipschitz constants between kernels that are log-concave perturbations of one another.

**Theorem 9** (Caffarelli's)**.** *Suppose $\gamma(x)$ is a log-concave measure for $x$ and $\ell(x)$ is a log-concave function such that $\gamma'(x) = \ell(x)\gamma(x)$ is a probability measure over $x$. Then there exists a 1-Lipschitz mapping $T : \mathcal{X} \to \mathcal{X}$ such that if $x \sim \gamma(x)$ then $T(x) \sim \gamma'(x)$.*

We can think of $\gamma(x)$ as a prior distribution $p(x)$, $\ell(x)$ as a normalized conditional likelihood $p(y|x)/p(y)$ and $\gamma'(x)$ as the posterior $p(x|y)$. As $\ell(x)$ is log-concave, we call $\gamma'(x)$ a *log-concave perturbation* of $\gamma$.

The original version of Caffarelli's log-concave perturbation theorem (Colombo et al., 2015; Saumard and Wellner, 2014) requires the prior $\gamma(x)$ to be *strongly* log-concave (e.g. a Gaussian) to show that the mapping $T$ is a *strict* contraction $\|T\|_{Lip} < 1$; however this weaker version, Theorem 9 in (Villani, 2008), is sufficient for our purposes.

*Proof of Theorem 5*

Using Theorem 9, we can now prove Theorem 5.

*Proof of Theorem 5.* Let $\vec{\psi}_t, \vec{\psi}_t^{(0)}, \vec{\psi}_t^{(1)}$ be random mappings associated respectively with forward kernels $\vec{\Psi}_t, \vec{\Psi}_t^{(0)}, \vec{\Psi}_t^{(1)}$. Because the transition and emission distributions are log-concave and log-concavity is preserved under product and marginalization (Saumard and Wellner, 2014), $\vec{\Psi}_t$, $\vec{\Psi}_t^{(0)}$, $\vec{\Psi}^{(1)}$ are log-concave and $p(y_{t\geq} \,|\, x_t)$ and $p(y_{>t} \,|\, x_t)$ are also log-concave.

Since $p(y_{\geq t} \,|\, x_t)$ is log-concave, we can write $\vec{\Psi}_t$ as a log-concave perturbation of $\vec{\Psi}_t^{(0)}$,

$$\vec{\Psi}_t = p(x_t \,|\, x_{t-1}, y_{t:T}, \theta) \propto p(y_{\geq t} \,|\, x_t) p(x_t \,|\, x_{t-1}, \theta) = p(y_{\geq t} \,|\, x_t) \cdot \vec{\Psi}_t^{(0)} \ . \tag{A.50}$$

Therefore, there exists $T_t^{(0)}$ with $\|T_t^{(0)}\|_{Lip} \leq 1$ such that $\vec{\psi}_t = (T_t^{(0)} \circ \vec{\psi}_t^{(0)})$. Thus,

$$\|\vec{\Psi}_t\|_{Lip} = \|T_t^{(0)}\|_{Lip} \cdot \|\vec{\Psi}_t^{(0)}\|_{Lip} \leq \|\vec{\Psi}_t^{(0)}\|_{Lip}. \tag{A.51}$$

Similarly, we can write $\vec{\Psi}_t$ as a log-concave perturbation of $\vec{\Psi}_t^{(1)}$ using $p(y_{>t} \,|\, x_t)$, thus $\|\vec{\Psi}_t\|_{Lip} \leq \|\vec{\Psi}_t^{(1)}\|_{Lip}$.

$$\vec{\Psi}_t = p(x_t \,|\, x_{t-1}, y_{t:T}, \theta) \propto p(y_{>t} \,|\, x_t) p(x_t \,|\, y_t, x_{t-1}, \theta) = p(y_{>t} \,|\, x_t) \cdot \vec{\Psi}_t^{(1)} \ . \tag{A.52}$$

□

Note the assumptions for equivalent results in the backward smoothers $\overleftarrow{\Psi}_t$ are identical. Log-concavity in $p(x_t \,|\, x_{t+1}, \theta)$ is implied from both $p(x_t \,|\, x_{t-1}, \theta)$ and the prior $p(x_t)$ being log-concave.

### *A.1.5 Bounds for Specific Models*

We now provide specific bounds for the buffering error for models we consider in Chapter 3.4 (LGSSM and SVM) using Theorem 5.

For both the LGSSM and SVM, we assume the prior $\nu(x_0|\theta) = \mathcal{N}(0, \sigma^2/(1-\phi^2))$. Then the latent state transitions are

$$\begin{aligned} p(x_t \,|\, x_{t-1}, \theta) &= \mathcal{N}(x_t \,|\, \phi x_{t-1}, \sigma^2) \\ p(x_t \,|\, x_{t+1}, \theta) &= \mathcal{N}(x_t \,|\, \phi x_{t+1}, \sigma^2) \ , \end{aligned}$$

which are both Gaussian and therefore log-concave in $x$.

Similarly, the emissions for the LGSSM and SVM are also log-concave in $x$:
For the LGSSM,

$$p(y_t \,|\, x_t, \theta) \propto \exp\left(-\tfrac{(y_t - x_t)^2}{2\sigma^2}\right) \ ,$$

which is log-concave.
For the SVM,

$$p(y_t \,|\, x_t, \theta) \propto \exp\left(-\tfrac{y_t^2}{2\sigma^2} \cdot e^{-x_t} - \tfrac{x_t}{2}\right) \ ,$$

which is log-concave as $e^{-x}$ is convex.

*Contraction Bound for LGSSM*

We assume the prior $\nu(x_0|\theta) = \mathcal{N}(0, \sigma^2/(1-\phi^2))$. For the LGSSM, the filtered kernels are

$$\begin{aligned} \vec{\Psi}_t^{(1)}(x_t \,|\, x_{t-1}) &= p(x_t \,|\, x_{t-1}, y_t, \theta) \propto \mathcal{N}(x_t|\phi x_{t-1}, \sigma^2) \cdot \mathcal{N}(y_t|x_t, \tau^2), \\ \overleftarrow{\Psi}_t^{(1)}(x_t \,|\, x_{t+1}) &= p(x_t \,|\, x_{t+1}, y_t, \theta) \propto \mathcal{N}(x_t|\phi x_{t+1}, \sigma^2) \cdot \mathcal{N}(y_t|x_t, \tau^2). \end{aligned} \tag{A.53}$$

Therefore,

$$\vec{\Psi}_t^{(1)}(x_t \,|\, x_{t-1}) = \mathcal{N}\left(x_t \,\middle|\, \frac{\sigma^2 y_t + \phi\tau^2 x_{t-1}}{\sigma^2+\tau^2}\,,\, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right),$$
$$\overleftarrow{\Psi}_t^{(1)}(x_t \,|\, x_{t+1}) = \mathcal{N}\left(x_t \,\middle|\, \frac{\sigma^2 y_t + \phi\tau^2 x_{t+1}}{\sigma^2+\tau^2}\,,\, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right). \tag{A.54}$$

The associated random mapping are,

$$\vec{\psi}_t^{(1)}(x_t \,|\, x_{t-1}) = \frac{\sigma^2 y_t}{\sigma^2+\tau^2} + \frac{\phi\tau^2}{\sigma^2+\tau^2}\cdot x_{t-1} + \vec{z}_t\,,$$
$$\overleftarrow{\psi}_t^{(1)}(x_t \,|\, x_{t+1}) = \frac{\sigma^2 y_t}{\sigma^2+\tau^2} + \frac{\phi\tau^2}{\sigma^2+\tau^2}\cdot x_{t+1} + \overleftarrow{z}_t\,, \tag{A.55}$$

where $\vec{z}_t$ and $\overleftarrow{z}_t$ are $\mathcal{N}\left(0\,,\, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right)$ random variables.

Since these maps are linear, we have $\|\vec{\Psi}_t^{(1)}\|_{Lip} = \|\overleftarrow{\Psi}_t^{(1)}\|_{Lip} = |\phi| \cdot \frac{\tau^2}{\sigma^2+\tau^2}$. Applying Theorem 5, we obtain

$$L_\theta \le \max_t\{\|\vec{\Psi}_t^{(1)}\|, \|\overleftarrow{\Psi}_t^{(1)}\|\} = |\phi| \cdot (1+\sigma^2/\tau^2)^{-1}. \tag{A.56}$$

Therefore $L_\theta < 1$ whenever $|\phi| < 1 + \sigma^2/\tau^2$.

*Contraction Bound for SVM*

We assume the prior $\nu(x_0\theta) = \mathcal{N}(0, \sigma^2/(1-\phi^2))$. For the SVM, the prior kernels are,

$$\vec{\Psi}_t^{(0)}(x_t \,|\, x_{t-1}) = p(x_t \,|\, x_{t-1}, \theta) \propto \mathcal{N}(x_t|\phi x_{t-1}, \sigma^2),$$
$$\overleftarrow{\Psi}_t^{(0)}(x_t \,|\, x_{t+1}) = p(x_t \,|\, x_{t+1}, \theta) \propto \mathcal{N}(x_t|\phi x_{t+1}, \sigma^2). \tag{A.57}$$

The associated random mapping are

$$\vec{\psi}_t^{(0)}(x_t \,|\, x_{t-1}) = \phi \cdot x_{t-1} + \mathcal{N}\left(0\,,\, \sigma^2\right),$$
$$\overleftarrow{\psi}_t^{(0)}(x_t \,|\, x_{t+1}) = \phi \cdot x_{t+1} + \mathcal{N}\left(0\,,\, \sigma^2\right). \tag{A.58}$$

Applying Theorem 5, we obtain $L_\theta \le |\phi|$.

## *A.2 Additional Experiment Details and Results*

### *A.2.1 Experiment Hyperparameters*

*Priors*

In our experiments, we use the following (conjugate) priors for $\theta$.

For the discrete latent state sequence transition matrix $\Pi$, we use a flat-Dirichlet prior

$$\Pr(\Pi_k) \propto \prod_{k'} \Pi_{k,k'}^{\alpha_{k,k'}-1} \ , \text{ where } \alpha_{k,k'} = 1 \ . \tag{A.59}$$

For the continuous transition matrix $A$, we use a matrix normal prior

$$\Pr(A) \propto \exp\left(-\operatorname{tr}\left[V^{-1}(A-M)^T U^{-1}(A-M)\right]/2\right) \ , \tag{A.60}$$

with mean $M = 0$, diagonal column covariance $V = 10^2 \cdot I_n$, and row variance $U = Q$.

For the noise covariances $Q$ and $R$, we use flat Wishart priors over $Q^{-1}$ and $R^{-1}$

$$\Pr(Q^{-1}) \propto |Q|^{(n+1-\nu)/2} e^{-\operatorname{tr}(\Psi Q^{-1})/2} \ , \ \Pr(R^{-1}) \propto |R|^{(m+1-\nu)/2} e^{-\operatorname{tr}(\Psi R^{-1})/2} \ , \tag{A.61}$$

where $\Psi = \nu \cdot I$ and $\nu = n+1$ or $m+1$.

*List of Hyperparameters*

- Ion Channel (Full) Gaussian HMM
  - Prior: $\Pi_k$ are Dirichlet, $\mu$ is Normal, and $Q^{-1}$ are Wishart.
  - Initialization: using K-means on $y_t$
  - Stepsizes:

| SGLD | | SGRLD | |
|---|---|---|---|
| No-Buffer | Buffer | No-Buffer | Buffer |
| 0.0001 | 0.0001 | 0.01 | 0.01 |

- Ion Channel (Subset) Gaussian HMM

  - Prior: $\Pi_k$ are Dirichlet, $\mu$ is Normal, and $Q^{-1}$ are Wishart.
  - Initialization: using K-means on $y_t$
  - Stepsizes:

| SGLD | | SGRLD | |
|---|---|---|---|
| No-Buffer | Buffer | No-Buffer | Buffer |
| 0.001 | 0.001 | 0.001 | 0.001 |

- Synthetic ARHMM $T = 10^4$
  - Prior: $\Pi_k$ are Dirichlet, $A$ is matrix Normal, and $Q^{-1}$ are Wishart.
  - Initialization: using K-means on $[y_t, y_{t-1}]$
  - Stepsizes:

| SGLD | | | SGRLD | | |
|---|---|---|---|---|---|
| No-Buffer | Buffer | Full | No-Buffer | Buffer | Full |
| 0.0001 | 0.0001 | 0.01 | 0.001 | 0.001 | 0.1 |

- Synthetic ARHMM $T = 10^6$
  - Prior: $\Pi_k$ are Dirichlet, $A$ is matrix Normal, and $Q^{-1}$ are Wishart.
  - Initialization: using K-means on $[y_t, y_{t-1}]$
  - Stepsizes:

| SGLD | | | SGRLD | | |
|---|---|---|---|---|---|
| No-Buffer | Buffer | Full | No-Buffer | Buffer | Full |
| 0.0001 | 0.0001 | 0.1 | 0.0001 | 0.0001 | 0.1 |

- Canine Seizure ARHMM
  - Prior: $\Pi_k$ are Dirichlet, $A$ is matrix Normal, and $Q^{-1}$ are Wishart.
  - Initialization: using K-means on $[y_t, y_{t-1}]$

  - Stepsizes: SGLD = 0.01, SGRLD = 0.1.

- Synthetic LGSSM $T = 10^4$
  - Prior: $A$ is matrix Normal and $Q^{-1}, R^{-1}$ are Wishart.
  - Initialization: From prior with $\nu = 4, \Psi = 4 \cdot I_2$ for the Wishart priors.
  - Stepsizes:

| SGLD | | | SGRLD | | |
|---|---|---|---|---|---|
| No-Buffer | Buffer | Full | No-Buffer | Buffer | Full |
| 0.01 | 0.01 | 0.1 | 0.01 | 0.01 | 0.1 |

- Synthetic LGSSM $T = 10^6$
  - Prior: $A$ is matrix Normal and $Q^{-1}, R^{-1}$ are Wishart.
  - Initialization: From prior with $\nu = 4, \Psi = 4 \cdot I_2$ for the Wishart priors.
  - Stepsizes:

| SGLD | | | SGRLD | | |
|---|---|---|---|---|---|
| No-Buffer | Buffer | Full | No-Buffer | Buffer | Full |
| 0.01 | 0.01 | 1.0 | 0.01 | 0.01 | 1.0 |

- Synthetic SLDS $T = 10^4$
  - Prior: $\Pi_k$ is Dirichlet, $A_k$ is matrix Normal and $Q_k^{-1}, R^{-1}$ are Wishart.
  - Initialization: $R$ from Wishart Prior, $\Pi, A, Q$ from $K$-means as in ARHMM.
  - Stepsizes: SGRLD X = 0.5, SGRLD Z = 0.1, SGRLD XZ = 0.1.

- Synthetic SLDS $T = 10^6$
  - Prior: $\Pi_k$ is Dirichlet, $A_k$ is matrix Normal and $Q_k^{-1}, R^{-1}$ are Wishart.
  - Initialization: $R$ from Wishart Prior, $\Pi, A, Q$ from $K$-means as in ARHMM.

  - Stepsizes: SGRLD X = 0.5, SGRLD Z = 0.1, SGRLD XZ = 0.1.

- Canine Seizure SLDS

  - Prior: $\Pi_k$ is Dirichlet, $A_k$ is matrix Normal and $Q_k^{-1}, R^{-1}$ are Wishart.
  - Initialization: $R$ from Wishart Prior, $\Pi, A, Q$ from $K$-means as in ARHMM.
  - Stepsizes: SGRLD = 0.1, SGLD = 0.1.

- Daily Weather SLDS

  - Prior: $\Pi_k$ is Dirichlet, $A_k$ is matrix Normal and $Q_k^{-1}, R^{-1}$ are Wishart.
  - Initialization: $R$ from Wishart Prior, $\Pi, A, Q$ from $K$-means as in ARHMM.
  - Stepsizes: SGRLD = 0.1, SGLD = 0.1.

- Hourly Weather SLDS

  - Prior: $\Pi_k$ is Dirichlet, $A_k$ is matrix Normal and $Q_k^{-1}, R^{-1}$ are Wishart.
  - Initialization: $R$ from Wishart Prior, $\Pi, A, Q$ from $K$-means as in ARHMM.
  - Stepsizes: SGRLD = 0.1, SGLD = 0.01.

.

### *A.2.2 Additional Synthetic Experiment Plots*

We now present additional plots for the synthetic data experiments. These plots show the MSE for 'other' components of $\theta$ to the true parameters of $\theta^*$ as well as other measures of fit such as predictive loglikelihood or recovery of the latent state sequence (NMI or RMSE).

*ARHMM*

Figure A.1 are plots of additional metrics for the ARHMM.

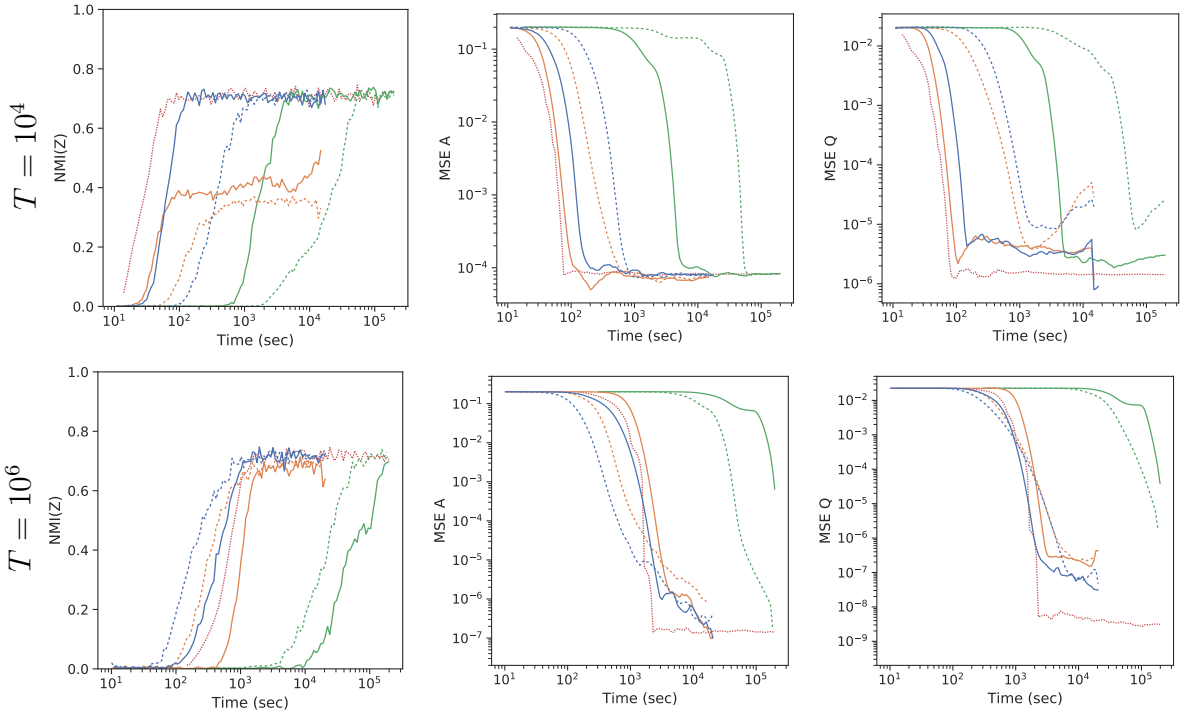


Figure A.1: Additional Metrics vs Runtime on ARHMM data with $T = 10^4$ (top), $T = 10^6$ (bottom), for different methods: (Gibbs), (Full), (No Buffer) and (Buffer) SGMCMC. For SGMCMC methods, solid (—) and dashed (- -) lines indicate SGRLD and SGLD respectively. The different metrics are: (left) NMI, (center) $MSE(\hat{A}^{(s)}, A^*)$ (right) $MSE(\hat{Q}^{(s)}, Q^*)$.

*LGSSM*

Figure A.2 are plots of additional metrics for the LGSSM synthetic data.

*SLDS*

Figure A.3 are plots of additional metrics for the SLDS data.

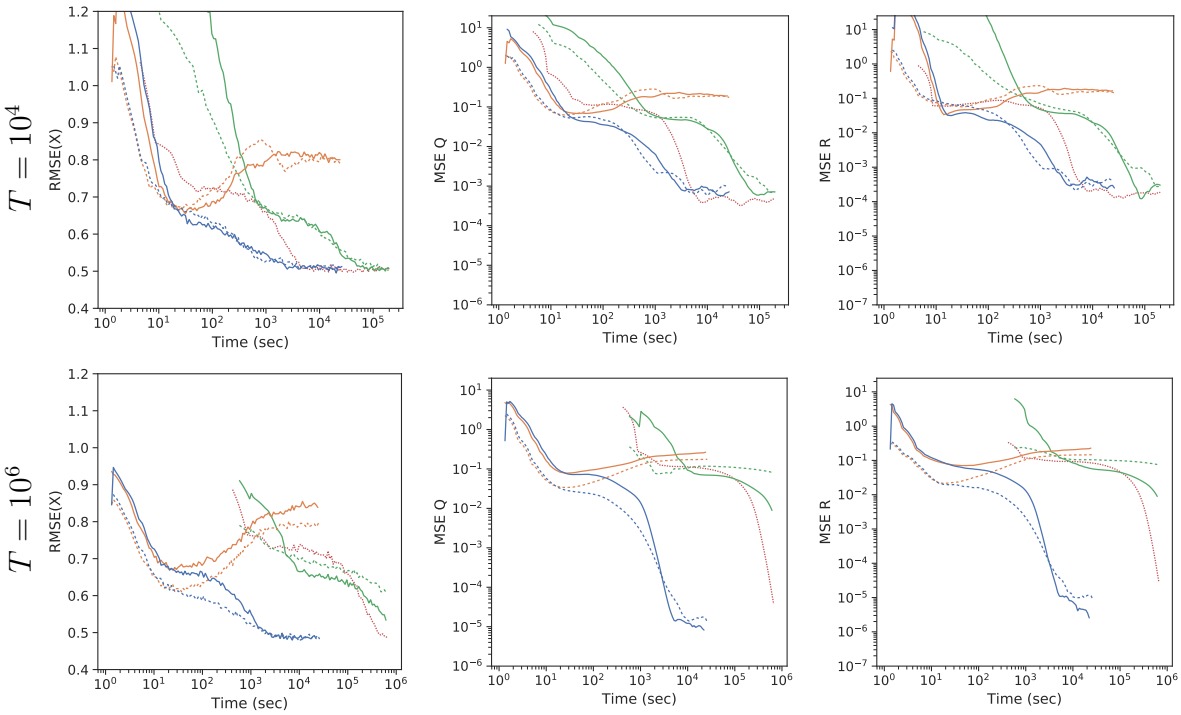


Figure A.2: Additional Metrics vs Runtime on LGSSM data with $T = 10^4$ (top), $T = 10^6$ (bottom), for different methods: (Gibbs), (Full), (No Buffer) and (Buffer) SGMCMC. For SGMCMC methods, solid (—) and dashed (- -) lines indicate SGRLD and SGLD respectively. The different metrics are: (left) root-mean squared error (RMSE) between $\hat{x}$ and $x^*$, (center) estimation error $MSE(\hat{Q}^{(s)}, Q^*)$ (right) estimation error $MSE(\hat{R}^{(s)}, R^*)$ .

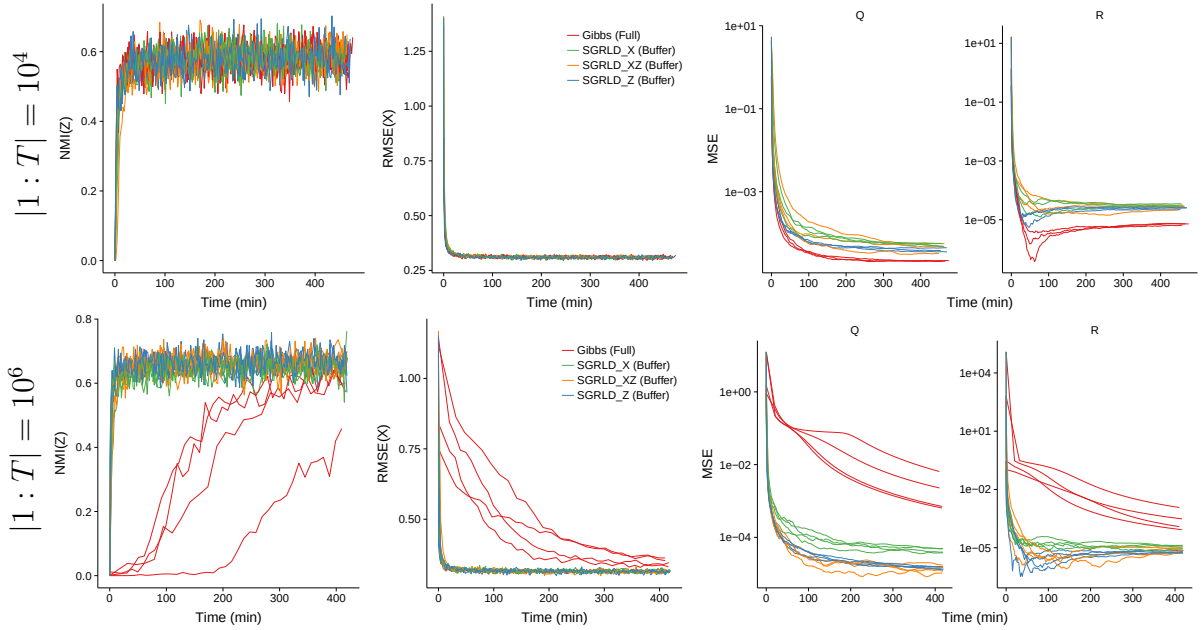


Figure A.3: Additional Metrics vs Runtime on SLDS data: (Top) $|1:T| = 10^4$, (Bottom) $|1:T| = 10^6$. (Left) NMI between $\hat{z}$ and $z^*$. (Center) root-mean square error (RMSE) between $\hat{x}$ and $x^*$, (Right) estimation error $\|\theta^{(s)} - \theta^*\|$. Methods: Gibbs, SGRLD X, SGRLD XZ, and SGLRD Z.

*LGSSM Particle Smoother*

Figure A.4 presents extra MSE plots for the parameters not presented in Chapter 3.5.3.

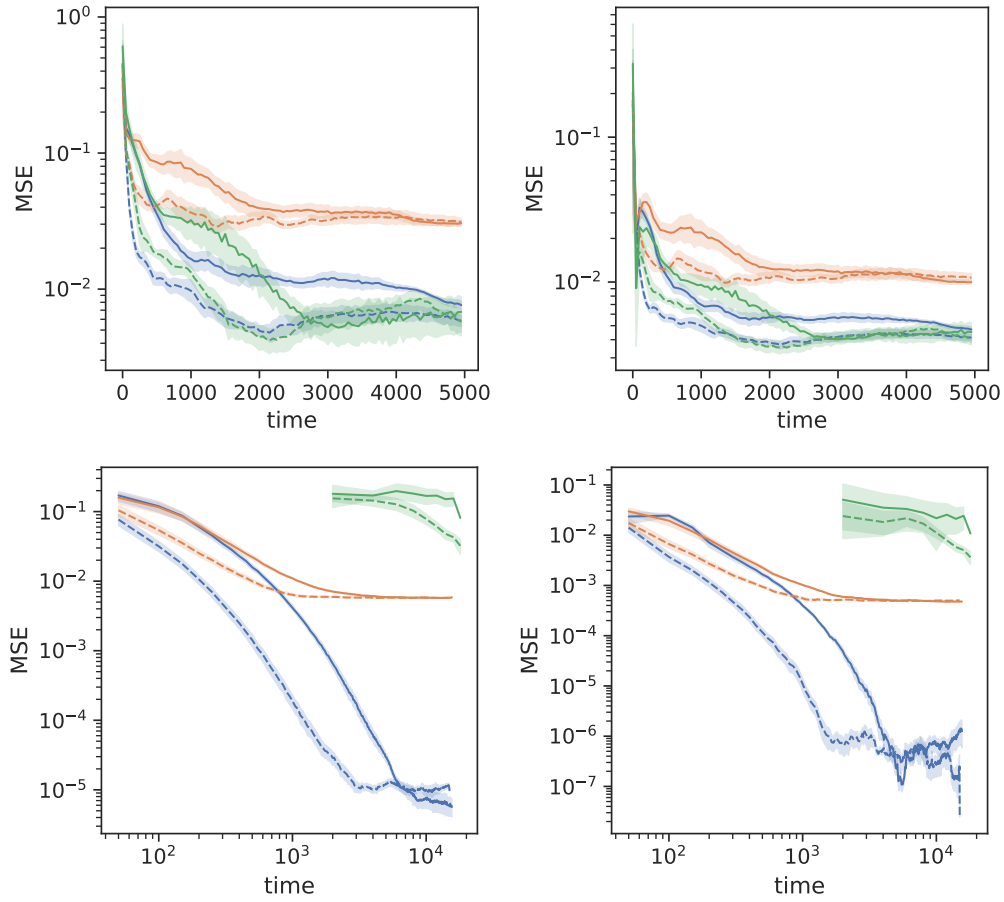


Figure A.4: Additional metrics for SGLD on LGSSM: (left) MSE of $\sigma$, (right) MSE of $\tau$, (top) $T = 10^3$, (bottom) $T = 10^6$

*Higher Dimensional LGSSM with Particle Smoothing*

We generate synthetic LGSSM data for $X_t, Y_t \in \mathbb{R}^d$ using $\phi = 0.9 \cdot \mathbb{I}_d$, $\sigma = 0.7 \cdot \mathbb{I}_d$, and $\tau = \mathbb{I}_d$ for dimensions $d \in \{5, 10\}$. Figure A.5 presents the trace plot metrics for $d = 5$ and for $d = 10$. Table A.1 presents the KSD tables for both.

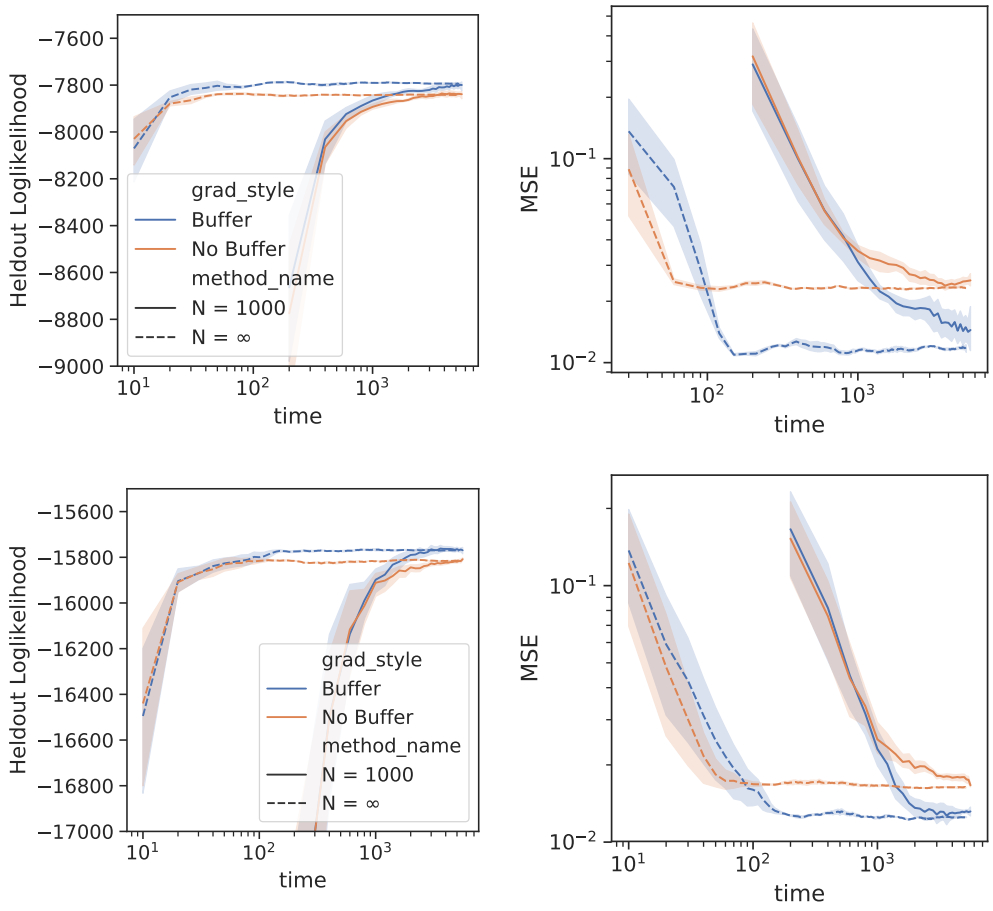


Figure A.5: SGLD Results for LGSSM (left) heldout loglikelihood, (right) MSE of $A$ for $x_t \in \mathbb{R}^5$ (top), $x_t \in \mathbb{R}^{10}$ (bottom).

We find that the Kalman filter $N = \infty$ is able to much more rapidly mix compared to the particle filter with $N = 1000$. This is both due to the increased particle filter variance in higher dimensions and the longer computation required for sampling particles in higher dimensions. However in both cases, we again see that buffering is necessary to avoid bias.

Table A.1: KSD results for Synthetic LGSSM in higher dimensions

| | | | $\log_{10}$KSD | | | |
|---|---|---|---|---|---|---|
| Dim | Grad Est. | N | $\phi$ | $\sigma$ | $\tau$ | Total |
| 5 | No Buffer | 1000 | 1.78 (0.04) | 1.97 (0.26) | 1.44 (0.45) | 2.28 (0.20) |
| | | $\infty$ | 1.74 (0.01) | 2.09 (0.02) | 1.64 (0.02) | 2.35 (0.01) |
| | Buffer | 1000 | 1.18 (0.17) | 1.74 (0.25) | 1.44 (0.03) | **2.01 (0.13)** |
| | | $\infty$ | 0.84 (0.03) | 1.97 (0.03) | 1.40 (0.05) | **2.10 (0.03)** |
| 10 | No Buffer | 1000 | 1.84 (0.01) | 2.40 (0.06) | 2.26 (0.13) | 2.71 (0.06) |
| | | $\infty$ | 1.79 (0.01) | 2.13 (0.04) | 2.12 (0.01) | 2.52 (0.02) |
| | Buffer | 1000 | 1.60 (0.13) | 2.37 (0.04) | 2.20 (0.04) | 2.64 (0.04) |
| | | $\infty$ | 1.04 (0.06) | 2.08 (0.04) | 2.07 (0.01) | **2.39 (0.02)** |

*SVM with Particle Smoothing*

Figure A.6 presents the trace plot metrics for SGLD on the synthetic SVM data $T = 1000$ and Table A.2 presents the KSD for each sampled chain.

We find that buffering performs best (as measured by KSD). From Figure A.6 we see that not buffering leads to bias, while the full sequence method is nosier (fewer larger steps) compared to the buffer method.

*GARCH with Particle Smoothing*

Figure A.7 presents the trace plot metrics for SGLD on the synthetic GARCH data $T = 1000$ and Table A.3 presents the KSD for each sampled chain.

We again find that buffering performs best (as measured by KSD). From Figure A.7 we see that not buffering leads to bias in sampling $\mu$ and $\lambda$. The full sequence method encounters high particle error and therefore requires a much longer runtime with a much smaller stepsize

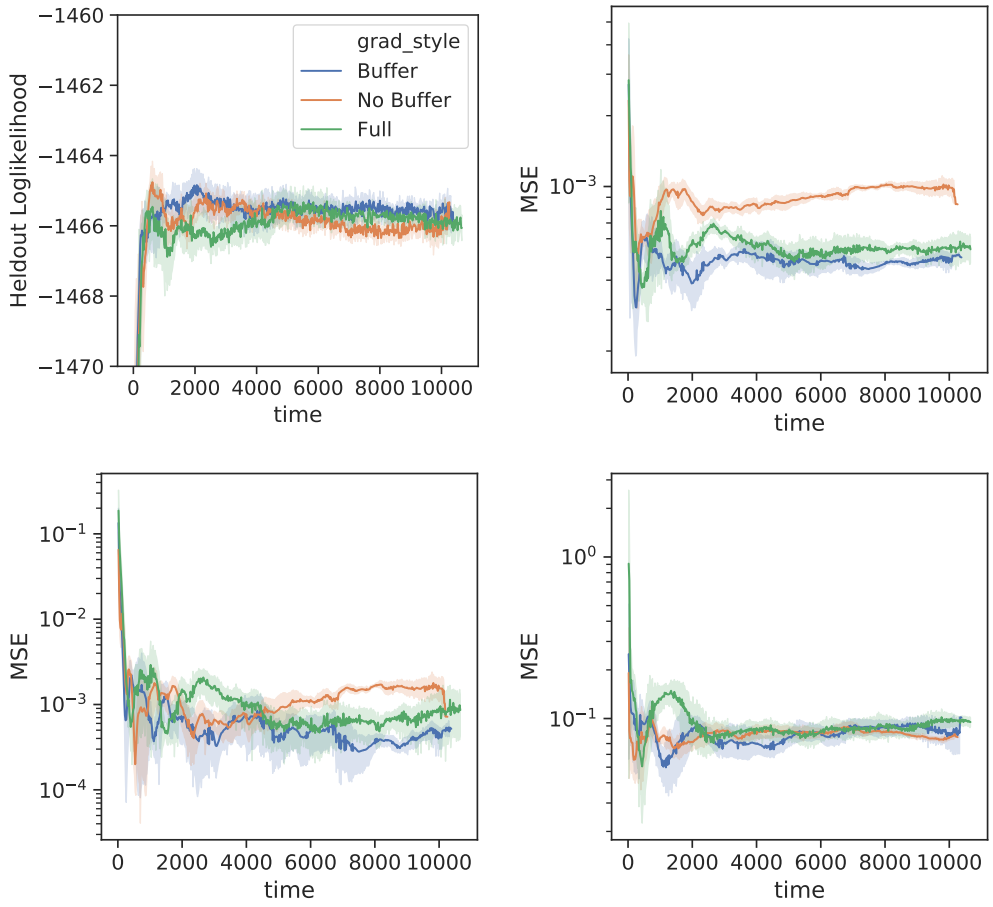


Figure A.6: SGLD results for synthetic SVM data: (top-left) heldout loglikelihood, (top-right) MSE of $\phi$, (bottom-left) MSE of $\sigma$, (bottom-right) MSE of $\tau$.

to reduce bias.

Table A.2: KSD results for Synthetic SVM

| | $\log_{10}$KSD | | | |
|---|---|---|---|---|
| Grad Est. | $\phi$ | $\sigma$ | $\tau$ | Total |
| Full | 0.68 (0.28) | 0.38 (0.40) | 0.44 (0.54) | 1.12 (0.22) |
| No Buffer | 1.49 (0.05) | -0.01 (0.23) | 0.09 (0.35) | 1.53 (0.05) |
| Buffer | 0.35 (0.33) | 0.23 (0.29) | 0.21 (0.40) | **0.81 (0.22)** |

Table A.3: KSD results for Synthetic GARCH

| | $\log_{10}$KSD | | | | |
|---|---|---|---|---|---|
| Grad Est. | $\log \mu$ | $\text{logit}\, \lambda$ | $\text{logit}\, \phi$ | $\tau$ | Total |
| Full | 0.29 (0.59) | 0.04 (0.03) | 0.18 (0.34) | 0.55 (0.11) | 0.97 (0.05) |
| No Buffer | 0.07 (0.08) | -0.38 (0.09) | -0.15 (0.10) | 0.56 (0.10) | 0.77 (0.08) |
| Buffer | -0.27 (0.24) | -0.72 (0.19) | -0.69 (0.17) | 0.12 (0.19) | **0.39 (0.09)** |

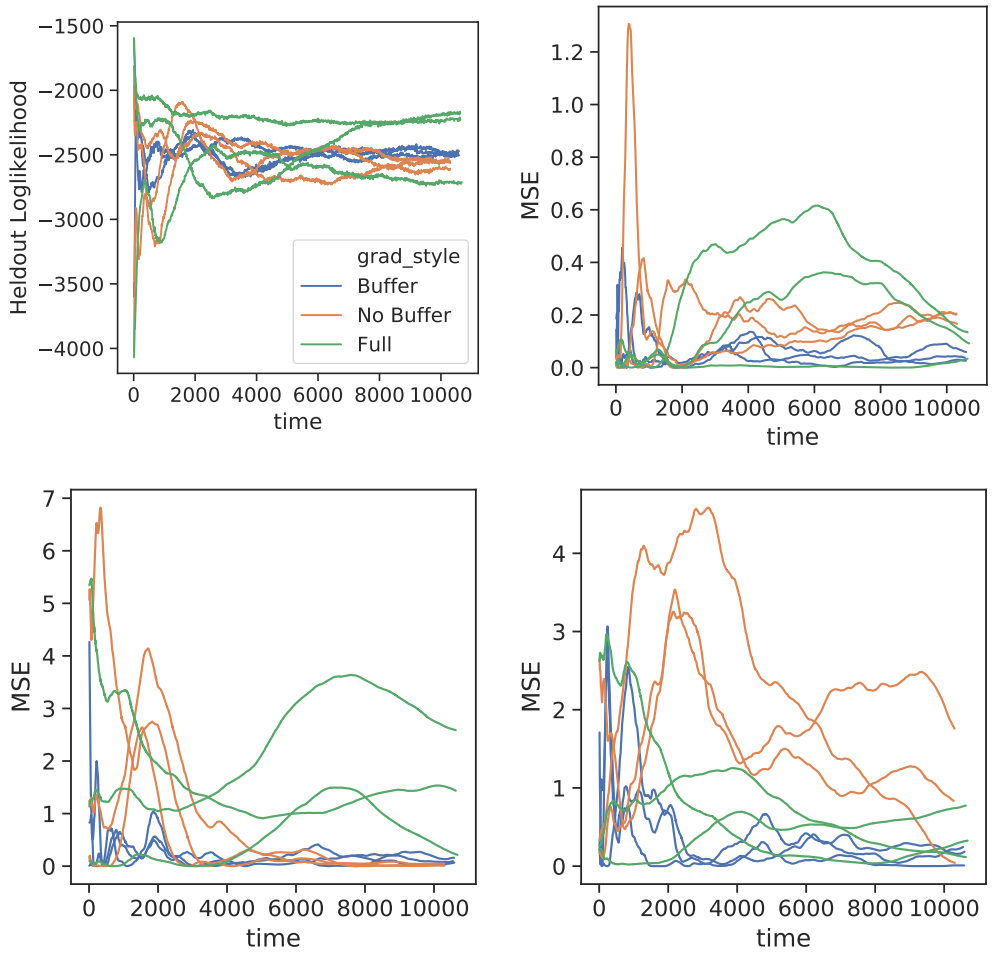


Figure A.7: SGLD results for synthetic SVM data: (top-left) heldout loglikelihood, (top-right) MSE of $\log(\mu)$, (bottom-left) MSE of logit $\phi$, (bottom-right) MSE of logit $\lambda$.

### *A.2.3 Downsampled Ion Channel Recordings*

We now consider a downsampled version of the ion channel recording data presented in Chapter 3.5.4. In particular, we consider downsampling the data by a factor of 50 (as in (Ma et al., 2017)), resulting in $T = 209,634$ observations. We again train on the first 90% and evaluate on the last 10% after applying a log-transform and normalizing the observations to use Gaussian emissions. For our SGMCMC methods we again use a subsequence size of $S = 10$ and a buffer size of $B = 0$ (no-buffer) or $B = 10$ (buffer). Figure A.8 presents our results including comparisons to Gibbs sampling (red). For this (shorter) downsampled data, Gibbs sampling outperforms the SGMCMC methods. We see that the performance of the SGMCMC method is similar to the full sample case (compare to Figure 3.17) and that SGRLD with buffering quickly reaches the same mode as Gibbs.

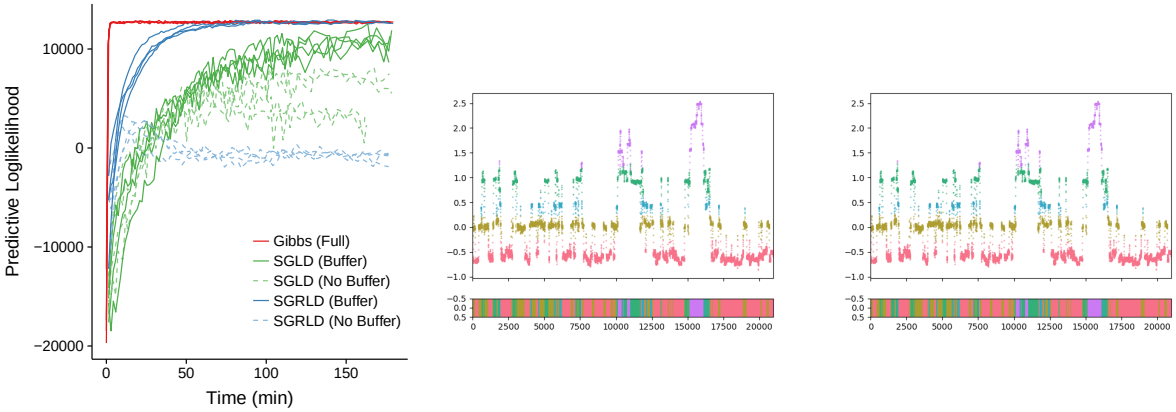


Figure A.8: Ion Channel Recordings: (Left) predictive loglikelihood vs runtime. (Center) example segmentation using Gibbs (Right) example segmentation using SGRLD.

### *A.2.4 SGLD on Exchange Rate Additional Details*

The EUR-US exchange rate data was fulled from the `https://www.finam.ru` website for the time period of November 2017 to October 2018 at the minute resolution. The *demeaned log-returns* are calculated by taking the difference of the log-closing price (at each minute) and removing the mean, as done in the `stochvol` package in R (Kastner, 2016)

$$\tilde{y}_t = \log(y_t/y_{t-1}) - \frac{1}{T}\sum_{t'} \log(y_{t'}/y_{t'-1}) \ . \tag{A.62}$$

The data is plotted in Figure A.9.

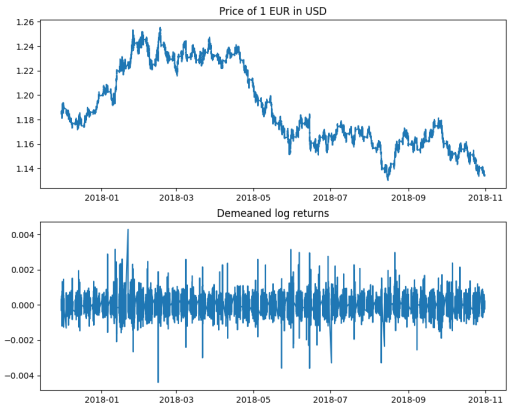


Figure A.9: EUR-US Exchange Rate Data (top) raw data (bottom) demeaned log-returns

*SVM*

For the SVM, we initialized each chain at $\phi = 0.9$, $\sigma = 1.73$ and $\tau = 0.1$ for all SGLD methods. The full KSD results are presented in Table A.4.

*GARCH*

For the GARCH model, we initialized each chain at $\log \mu = -0.4$, $\text{logit}\, \phi = 1.7$, $\text{logit}\, \lambda = 2.7$ and $\tau = 0.1$ for all SGLD methods. The full KSD results are presented in Table A.5.

Table A.4: KSD results for SVM on exchange rate data.

| | $\log_{10}$KSD | | | |
|---|---|---|---|---|
| Grad Est. | $\phi$ | $\sigma$ | $\tau$ | Total |
| Full | 3.63 (0.30) | 3.76 (0.07) | 1.46 (0.38) | 4.03 (0.14) |
| Weekly | 3.86 (0.08) | 2.18 (0.28) | 0.67 (0.39) | 3.87 (0.08) |
| No Buffer | 4.48 (0.01) | 1.84 (0.15) | 1.21 (0.14) | 4.48 (0.01) |
| Buffer | 3.53 (0.11) | 2.32 (0.13) | 1.23 (0.05) | **3.56 (0.10)** |

Table A.5: KSD results for GARCH on exchange rate data.

| | $\log_{10}$KSD | | | | |
|---|---|---|---|---|---|
| Grad Est. | $\log \mu$ | $\text{logit}\, \lambda$ | $\text{logit}\, \phi$ | $\tau$ | Total |
| Full | 2.18 (0.67) | 2.18 (0.07) | 2.19 (0.61) | 2.07 (0.06) | 2.84 (0.30) |
| Weekly | 2.17 (0.51) | 2.21 (0.03) | 2.31 (0.29) | 1.85 (0.19) | 2.81 (0.21) |
| No Buffer | 1.76 (0.06) | 1.43 (0.46) | 1.31 (0.09) | 1.58 (0.08) | **2.09 (0.09)** |
| Buffer | 1.76 (0.03) | 2.01 (0.08) | 1.11 (0.07) | 1.87 (0.07) | **2.19 (0.05)** |

# Appendix B

# ADAPTIVELY TRUNCATING BACKPROPAGATION THROUGH TIME

This chapter of the Appendix provides supplementary details for Chapter 4. We first present proofs for Chapter 4.2 and then provide additional tables and figures for the experiments.

## *B.1 Proofs for Chapter 4.2*

### *B.1.1 Proof of Theorem 7*

The proof of Theorem 7 consists of two part. First we bound the absolute bias by $\mathcal{E}(K, \theta)$ using assumptions (A-1) and (A-2). Then we bound the relative bias using the triangle inequality

*Proof of Theorem 7.* The bias of $\hat{g}_K$ is bounded by the expected error between $\hat{g}_K$ and $\hat{g}_T$

$$\|\mathbb{E}\left[\hat{g}_K(\theta)\right] - g(\theta)\| = \|\mathbb{E}\left[\hat{g}_K(\theta) - \hat{g}_T(\theta)\right]\| \leq \mathbb{E}\left[\|\hat{g}_K(\theta) - \hat{g}_T(\theta)\|\right] \quad . \tag{B.1}$$

Applying the triangle-inequality to the difference between $\hat{g}_K$ and $\hat{g}_T$ gives

$$\|\hat{g}_K(\theta) - \hat{g}_T(\theta)\| = \left\| \sum_{k=K+1}^{s} \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \cdot \frac{\partial h_{s-k}}{\partial \theta} \right\| \leq \sum_{k=K+1}^{s} \left\| \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \right\| \cdot \left\| \frac{\partial h_{s-k}}{\partial \theta} \right\| \leq \sum_{k=K+1}^{s} \left\| \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \right\| \cdot M \quad , \tag{B.2}$$

where in the last inequality we apply the assumption (A-2), $\|\partial h_t / \partial \theta\| < M$ for all $t$. Taking the expectation with respect to $s$ of both sides of Eq. (B.2) gives

$$\mathbb{E}\left[\|\hat{g}_K(\theta) - \hat{g}_T(\theta)\|\right] \leq \sum_{k=K+1}^{s} \mathbb{E}\left\| \frac{\partial \mathcal{L}_s}{\partial h_{s-k}} \right\| \cdot M = \sum_{k=K+1}^{s} \mathbb{E}\left[\phi_k\right] \cdot M \quad , \tag{B.3}$$

where we recall that $\phi_k = \|\partial \mathcal{L}_s / \partial h_{s-k}\|$.

Recursively applying assumption (A-2) to $\mathbb{E}[\phi_{\tau+t}]$ gives

$$\mathbb{E}[\phi_{\tau+t}] \leq \beta \cdot \mathbb{E}[\phi_{\tau+t-1}] \leq \ldots \leq \beta^t \cdot \mathbb{E}[\phi_\tau] \ . \tag{B.4}$$

Combining Eqs. (B.1), (B.3), and (B.4) gives us the first half of the result

$$\|\mathbb{E}[\hat{g}_K(\theta)] - g(\theta)\| \leq \mathbb{E}\left[\|\hat{g}_K(\theta) - \hat{g}_T(\theta)\|\right] \leq \sum_{k=K+1}^{s} \mathbb{E}_{[}\phi_k] \cdot M = \mathcal{E}(K,\theta) \ . \tag{B.5}$$

To bound the relative error, we apply the reverse triangle inequality to $\|g(\theta)\|$

$$\|g(\theta)\| \geq \|\mathbb{E}[\hat{g}_K(\theta)]\| - \|\mathbb{E}[\hat{g}_K(\theta)] - g(\theta)\| \geq \|\mathbb{E}[\hat{g}_K(\theta)]\| - \mathcal{E}(K,\theta) \ , \tag{B.6}$$

when $\|\mathbb{E}[\hat{g}_K(\theta)]\| - \mathcal{E}(K,\theta) > 0$.

Since $\mathcal{E}(K,\theta)$ is an upper bound for the numerator and $\|\mathbb{E}[\hat{g}_K(\theta)]\| - \mathcal{E}(K,\theta)$ is a lower bound for the denominator, we obtain the result

$$\frac{\|\mathbb{E}[\hat{g}_K(\theta)] - g(\theta)\|}{\|g(\theta)\|} \leq \frac{\mathcal{E}(K,\theta)}{\|\mathbb{E}[\hat{g}_K(\theta)]\| - \mathcal{E}(K,\theta)} = \delta(K,\theta) \ . \tag{B.7}$$

□

### *B.1.2 Proof of Theorem 8*

Let $\langle x_1, x_2 \rangle$ denote the inner-product between two vectors.

We first presents some Lemmas involving $\hat{g}(\theta)$ and $g(\theta)$ when the gradient has bounded relative bias $\delta$.

**Lemma 8.** *If $\hat{g}(\theta)$ has bounded relative bias of $\delta$ then*

$$\mathbb{E}\langle g(\theta), \hat{g}(\theta) - g(\theta)\rangle \leq \delta\|g(\theta)\|^2 \quad \textit{and} \quad \mathbb{E}\langle g(\theta), \hat{g}(\theta)\rangle \geq (1-\delta)\|g(\theta)\|^2 \tag{B.8}$$

*Proof of Lemma 8.* The first inequality follows from the Cauchy-Schwartz inequality and bound on relative bias

$$\mathbb{E}\langle g(\theta), \hat{g}(\theta) - g(\theta)\rangle = \langle g(\theta), \mathbb{E}[\hat{g}(\theta)] - g(\theta)\rangle \leq \|g(\theta)\|\|\mathbb{E}[\hat{g}(\theta) - g(\theta)\| \leq \delta\|g(\theta)\|^2 \ . \tag{B.9}$$

The second inequality follows immediately from the first

$$\langle g(\theta), \hat{g}(\theta) \langle = \langle g(\theta), g(\theta) \rangle + \langle g(\theta), \hat{g}(\theta) - g(\theta) \rangle \leq \|g(\theta)\|^2 - \delta \|g(\theta)\|^2 = (1-\delta)\|g(\theta)\|^2 \ . \quad \text{(B.10)}$$

□

The next lemma bounds the second moment of $\|\hat{g}(\theta)\|$.

**Lemma 9.** *If $\hat{g}$ has bounded relative bias $\delta$ and bounded variance $\sigma^2$ for all $\theta$ (assumption (A-4)), then*

$$\mathbb{E}\left[\|\hat{g}\|^2\right] \leq (1+\delta)^2 \|g\|^2 + \sigma^2 \ . \quad \text{(B.11)}$$

*Proof of Lemma 9.*

$$\|\hat{g}\|^2 = \|g\|^2 + 2\langle g, \hat{g} - g \rangle + \|\hat{g} - g\|^2 \quad \text{(B.12)}$$

Take the expectation, we obtain the result

$$\mathbb{E}\,\|\hat{g}\|^2 = \|g\|^2 + 2\mathbb{E}\,\langle g, \hat{g} - g \rangle + \mathbb{E}\,\|\hat{g} - g\|^2 \ , \quad \text{(B.13)}$$

where expand the mean-squared error into the bias squared plus variance

$$\mathbb{E}\,\|\hat{g} - g\|^2 = \|\mathbb{E}\,\hat{g} - g\|^2 + \mathbb{E}\,\|\hat{g} - \mathbb{E}\,\hat{g}\|^2 \leq \delta^2 \|g\|^2 = \sigma^2 \ . \quad \text{(B.14)}$$

Therefore

$$\mathbb{E}\,\|\hat{g}\|^2 \leq \|g\|^2 + 2\delta\|g\|^2 + (\delta^2\|g\|^2 + \sigma^2) = (1+\delta)^2\|g\|^2 + \sigma^2 \quad \text{(B.15)}$$

□

We now begin the proof of Theorem 8 which builds off the proof in (Ghadimi and Lan, 2013).

*Proof of Theorem 8.* From the $L$-smoothness of $\mathcal{L}$, assumption (A-3), we have

$$\mathcal{L}(\theta) - \mathcal{L}(\theta') - |\langle g(\theta), \theta - \theta' \rangle| \leq \frac{L}{2}\|\theta' - \theta\|^2, \quad \forall \theta, \theta' \ . \quad \text{(B.16)}$$

Substituting $\theta = \theta_{n+1}$ and $\theta' = \theta_n$, where $\theta_{n+1}$ and $\theta_n$ are connected through SGD Eq. (4.7), we obtain

$$\mathcal{L}(\theta_{n+1}) \leq \mathcal{L}(\theta_n) + \langle g(\theta_n), \theta_{n+1} - \theta_n \rangle + \frac{L}{2}\|\theta_{n+1} - \theta_n\|^2 \tag{B.17}$$

$$= \mathcal{L}(\theta_n) - \gamma_n \langle g(\theta_n), \hat{g}(\theta_n) \rangle + \frac{L}{2}\gamma_n^2 \|\hat{g}(\theta_n)\|^2 \ . \tag{B.18}$$

Taking the expectation with respect to $\hat{g}(\theta_n)$ on both sides and using Lemmas 8 and 9 gives us

$$\mathbb{E}\,\mathcal{L}(\theta_{n+1}) = \mathcal{L}(\theta_n) - \gamma_n \mathbb{E}\,\langle g(\theta_n), \hat{g}(\theta_n) \rangle + \frac{L}{2}\gamma_n^2 \mathbb{E}\,\|\hat{g}(\theta_n)\|^2 \tag{B.19}$$

$$\leq \mathcal{L}(\theta_n) - \gamma_n(1-\delta)\|g(\theta_n)\|^2 + \frac{L}{2}\gamma_n^2((1+\delta)^2\|g(\theta_n)\|^2 + \sigma^2) \ . \tag{B.20}$$

Rearranging terms with $\gamma_n$ gives

$$\frac{\gamma_n(1-\delta)}{2}\left(2 - \gamma_n \frac{L(1+\delta)^2}{(1-\delta)}\right) \cdot \|g(\theta_n)\|^2 \leq \mathcal{L}(\theta_n) - \mathbb{E}\,\mathcal{L}(\theta_{n+1}) + \gamma_n^2 \frac{L\sigma^2}{2} \ . \tag{B.21}$$

As we assume the stepsizes are $\gamma_n < \frac{1-\delta}{L(1+\delta)^2}$, therefore $(2 + \gamma_n \frac{L(1+\delta)^2}{(1-\delta)}) < 1$ and we can drop these terms. Taking the summation over $n$ and taking the expectation with respect to $\hat{g}(\theta_n)$ for $n = 1, \ldots, N$ we obtain

$$\sum_{n=1}^{N} \gamma_n \frac{(1-\delta)}{2} \cdot \min_{n \in [1,N+1]} \|g(\theta_n)\|^2 \leq \mathcal{L}(\theta_1) - \mathbb{E}\,\mathcal{L}(\theta_{N+1}) + \sum_{n=1}^{N} \gamma_n^2 \frac{L\sigma^2}{2} \ . \tag{B.22}$$

Finally, we divide both sides by $\sum_n \gamma_n \frac{1-\delta}{2}$ and apply $\mathbb{E}\,\mathcal{L}(\theta_{N+1}) \geq \min_{\theta^*} \mathcal{L}(\theta^*)$ to obtain the result

$$\min_{n \in [1,N+1]} \|g(\theta_n)\|^2 \leq \frac{2D_{\mathcal{L}} + L\sigma^2 \sum_{n=1}^{N} \gamma_n^2}{(1-\delta)\sum_{n=1}^{N} \gamma_n} \ , \tag{B.23}$$

where $D_{\mathcal{L}} = \mathcal{L}(\theta_1) - \min_{\theta^*} \mathcal{L}(\theta^*)$.

If we use a constant stepsize $\gamma_n = \gamma$ for all $n \in [1, N]$, then the optimal stepsize for $N$ steps of SGD is

$$\gamma = \sqrt{\frac{2D_{\mathcal{L}}}{NL\sigma^2}} \quad \text{which achieves} \quad \min_{n \in [1,N+1]} \|g(\theta_n)\|^2 \leq \frac{1}{1-\delta} \cdot \sqrt{\frac{8D_{\mathcal{L}}L\sigma^2}{N}} \ . \tag{B.24}$$

If instead a decaying $\mathcal{O}(n^{-1/2})$ stepsize is used, then the numerator of Eq. (B.23) grows as a harmonic series $\mathcal{O}(\sum_n n^{-1}) = \mathcal{O}(\log n)$, while the denominator grows $\mathcal{O}(\sum_n n^{-1/2}) = \mathcal{O}(n^{1/2})$. Therefore the overall rate is $\mathcal{O}(n^{-1/2} \log n)$. □

### *B.1.3 Comparison of Bounds to (Chen and Luss, 2018)*

In Chapter 4.2.3 for Theorem 8, we assume the *relative bias* is bounded, that is $\|\mathbb{E}\left[\hat{g}(\theta)\right] - g(\theta)\| \leq \delta\|g(\theta)\|$ for all $\theta$ (Eq. (4.8)). Chen and Luss (2018) prove similar results to Theorem 8, where they assume the relative error of each gradient is bounded in high probability, that is there exists $\delta, \epsilon > 0$ such that

$$\Pr(\|\hat{g}(\theta) - g(\theta)\| \leq \delta\|g(\theta)\|) > 1 - \epsilon \ , \text{ for all } \theta \ . \tag{B.25}$$

Although Markov's inequality implies that if the relative bias is bounded by $\delta \cdot \epsilon$, when Eq. (B.25) holds for $\delta, \epsilon$, their non-convex optimization results only hold in high probability rather than uniformly. A key drawback of their results, is that the relative error must be bounded in high probability for all steps of SGD ($\hat{g}_{1:N}$); therefore the required $\epsilon$ for each step depends on the total number of SGD steps during training (see Chen and Luss, 2018, Eq.(7) and Theorem 5). Specifically, Chen and Luss (2018) observe that the probability the relative error is controlled for all $N$ steps is bounded by $1 - \epsilon_{\text{total}} \leq (1 - \epsilon)^N$ under the additional assumption that the noise in $\hat{g}(\theta)$ is independent. For their results to hold with probability $1 - \epsilon_{\text{total}}$ after $N$ steps, each gradient must have a relative error bound with $\epsilon \leq 1 - (1 - \epsilon_{\text{total}})^{1/N}$. Chen and Luss (2018) achieve this by restricting $\epsilon \leq \epsilon_{\text{total}}/N$. Our result assumes the relative error is bounded in expectation, which sides steps this issue. However our results are not as robust in the sense that they do not hold if the noise in $\hat{g}(\theta)$ does not have an expected value (e.g. if $\hat{g}(\theta) - g(\theta)$ is Cauchy).

## *B.2 Additional Experiments*

This section provides additional tables and figures for the experiments in Chapter 4.4.

In our experiments, we selected the stepsize $\gamma$ for SGD by performing a grid search over powers of 10 and selected the largest stepsize that did not diverge for fixed TBPTT (with $K = 15$ for the synthetic tasks, $K = 100$ for the language modeling tasks, and $K = 6$ for the temporal point process tasks). We also consider adaptive and decaying stepsizes (specifically

ADADELTA, SGD with Momentum, and epoch-wise stepsize decay); however, we did not see a significant difference in results.

### *B.2.1 Synthetic 'Copy' Experiment*

Figure B.1 shows the validation PPL for the two 'Copy' experiments. The left pair of figures show the validation PPL while the right pair shows the cumulative minimum (i.e. the 'best') validation PPL. The test PPL plots in Figure 4.1 are piecewise constant evaluated using these 'best' validation PPL parameters. The top row corresponds to the fixed-memory $m = 10$ copy experiment, and we see the loss decays relatively smoothly. The bottom row corresponds to the variable-memory $m \in [5, 10]$ copy experiment, and we see heavy oscillation in the validation error as it decays.

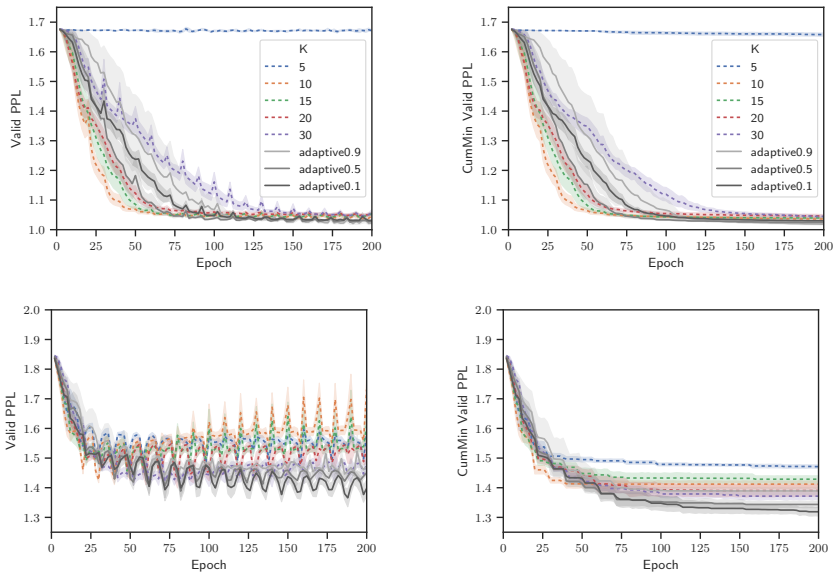


Figure B.1: Synthetic Copy Supplement: (left) Valid PPL vs epoch, (right) 'Best' Valid PPL vs epoch (Top) fixed $m = 10$, (bottom) variable $m \in [5, 10]$. Solid dark lines are our adaptive TBPTT methods, dashed colored lines are fixed TBPTT baselines.

### *B.2.2 Language Modeling Experiment*

Figure B.2 shows the validation PPL for the two language modeling experiments. The left pair of figures show the validation PPL while the right pair shows the cumulative minimum (i.e. the 'best') validation PPL. The top row corresponds to the PTB experiment. We see that fixed TBPTT with small $K$ quickly begins to over-fit (as the validation PPL increases). With larger $K$, fixed TBPTT achieves lower validation (and test) PPL, but requires more epochs. We see that the adaptive TBPTT with $\delta = 0.1$, achieves a better PPL much more rapidly. The bottom row corresponds to Wiki2 experiment, where we see that the adaptive TBPTT and best fixed TBPTT method perform similarly.

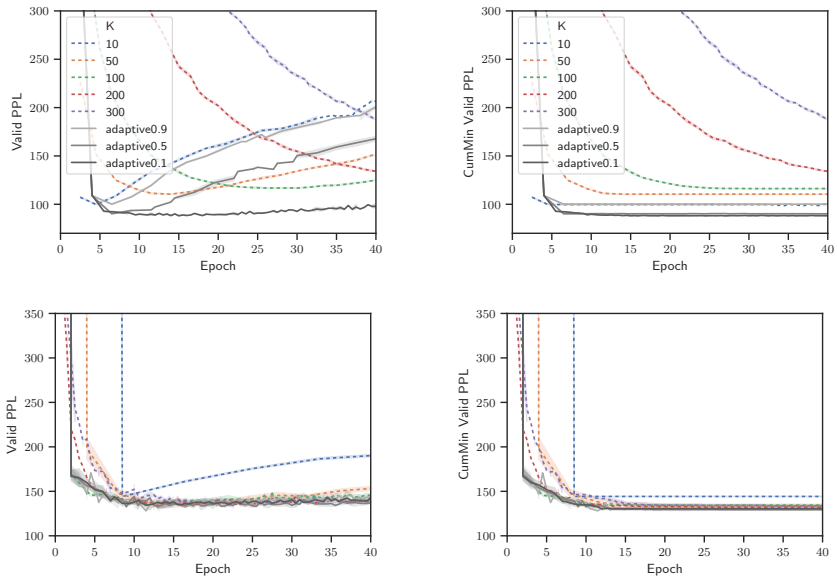


Figure B.2: Language Modeling Supplement: (left) Valid PPL vs epoch, (right) 'Best' Valid PPL vs epoch. (Top) PTB, (bottom) Wiki2. Solid dark lines are our adaptive TBPTT methods, dashed colored lines are fixed TBPTT baselines.

# Appendix C

# APPROXIMATE COLLAPSED GIBBS CLUSTERING

This chapter of the Appendix provides supplementary details for Chapter 5. We first present additional details for the mixture of Student-$t$ and time series clustering models. We then outline the error analysis for applying EP to collapsed Gibbs. We finally present additional figures and tables for the experiments.

## *C.1 Mixture of Multivariate Student-$t$*

This section provides additional details for Sec. 5.3.1 on multivariate Student-$t$ distributions (MVT). We first provide the details for the naive and blocked (partially collapsed) Gibbs sampler based on data augmentation. We then provide the details on how to approximate the collapsed log-likelihood and moments required for our EP approximation.

### *C.1.1 Naive Sampler Steps*

For notation, we will let $2\alpha$ be the degrees of freedom of the MVT distribution and reserve $\nu$ for the degrees of freedom in the inverse-Wishart distribution.

*Sampling $z$*

$$p(z_i \,|\, z_{-i}, \mu, \Sigma, u, y) \propto \mathcal{N}(y_i \,|\, \mu_{z_i}, \Sigma_{z_i}/u_{i,z_i}) p(z_i \,|\, z_{-i}) \tag{C.1}$$

which can be evaluated for each $z_i = k$ and then normalized.

*Sampling $\mu, \Sigma$*

$$p(\mu_k, \Sigma_k \,|\, z, u, y) = \mathcal{NIW}(\mu_k, \Sigma_k \,|\, \mu_p, \kappa_p, \nu_p, \Psi_p) = \mathcal{N}(\mu_k \,|\, \mu_p, \Sigma_k/\kappa_p) \cdot \mathcal{IW}(\Sigma_k \,|\, \nu_p, \Psi_p) \tag{C.2}$$

where $\mathcal{IW}$ is the inverse-Wishart distribution

$$\mathcal{IW}(\Sigma \mid \nu, \Psi) = \frac{|\Psi|^{\nu/2}}{2^{\nu d/2}\Gamma_d(\nu/2)}|\Sigma|^{-\frac{\nu+d+1}{2}}e^{-\frac{1}{2}\operatorname{tr}(\Psi X^{-1})}$$

and

$$\begin{aligned}
\mu_p &= (\kappa_0\mu_0 + \sum_{z_i=k} u_i y_i)/\kappa_p \\
\kappa_p &= \kappa_0 + \sum_{z_i=k} u_i \\
\nu_p &= \nu_0 + \sum_{z_i=k} 1 \\
\Psi_p &= \Psi_0 + \kappa_0\mu_0\mu_0^T + \sum_{z_i=k} u_i y_i y_i^T - \kappa_p\mu_p\mu_p^T
\end{aligned}$$

when $(\mu_0, \kappa_0, \nu_0, \Psi_0)$ are the parameters of the prior.

*Sampling u*

$$p(u_{i,k} \mid z, \mu, \Sigma, y) = \begin{cases} \Gamma(u_{i,k} \mid \alpha, \alpha) & \text{if } z_i \neq k \\ \Gamma(u_{i,k} \mid \alpha_1^*, \alpha_2^*) & \text{if } z_i = k \end{cases} \tag{C.3}$$

where $\alpha_1^* = \alpha + d/2$ and $\alpha_2^* = \alpha + (y_i - \mu_k)^T\Sigma_k^{-1}(y_i - \mu_k)$.

For the correctness of the sampler, we must to sample a separate $u_{i,k}$ for each observation-cluster pair $(i, k)$.

### *C.1.2 Blocked Sampler Steps*

Given a conjugate prior for $\mu, \Sigma$ (normal inverse-Wishart), the posterior over $\mu, \Sigma$ for fixed $z$ and $u$ is normal inverse-Wishart (see Eq. (C.2)).

Therefore we can integrate out $\mu$ and $\Sigma$ in the likelihood of Eq. (C.1) to obtain

$$p(z_i \mid z_{-i}, u, y) \propto \int \mathcal{N}\left(y_i \mid \mu_{z_i}, \frac{\Sigma_{z_i}}{u_{i,z_i}}\right) q(\mu_{z_i}, \Sigma_{z_i} \mid y_{-i}) \times p(z_i \mid z_{-i})\, d\mu_{z_i} d\Sigma_{z_i}$$

where $q(\mu_{z_i}, \Sigma_{z_i} \mid y_{-i}) = \mathcal{NIW}(\mu_{z_i}, \Sigma_{z_i} \mid \mu_p, \kappa_p, \nu_p, \Psi_p)$ is the NIW posterior calculated without observation $i$.

Taking the integral, we obtain

$$p(z_i|z_{-i}, u, y) = t_{2\alpha_p}\left(y_i \,|\, \mu_p, \Sigma_p\right) \tag{C.4}$$

where $t$ is a MVT distribution with mean $\mu_p$, covariance matrix $\Sigma_p = \frac{(\kappa_p + u_{i,z_i})\Psi_p}{(\kappa_p \cdot u_{i,z_i})(\nu_p - d + 1)}$ and degrees of freedom $2\alpha_p = \nu_p - d + 1$.

### *C.1.3 EP Approximate Log-likelihood*

We now present how to approximate the collapsed likelihood, approximating $p(\mu, \Sigma|y, z, u)$ with a normal inverse-Wishart $q(\mu, \Sigma)$.

The normalizing constant (a.k.a. the likelihood approximation) for fixed $u$ is given by the block sampler where our prior is our cavity distribution $q(\mu_{z_i}, \Sigma_{z_i} \,|\, y_{-i})$.

Therefore we can (tractably) estimate the normalizing constant by numerically integrating out (the univariate) $u_{i,k}$: the integrand is a MVT evaluated at $y_i$ with changing variance $\Sigma_p(u_{i,k})$ (see Eq. (C.4)).

### *C.1.4 EP Moment Update*

To update our EP approximation $q(\mu, \Sigma)$ we must calculate the moments of the sufficient statistics of $\mu, \Sigma$. For a normal inverse-Wishart the sufficient statistics and their moments are

$$\begin{aligned}
T_1 &= \Sigma^{-1}\mu & \mathbb{E}\left[T_1\right] &= \nu\Psi^{-1}\mu \\
T_2 &= \mu^T\Sigma^{-1}\mu & \mathbb{E}\left[T_2\right] &= \nu\mu^T\Psi^{-1}\mu + d/\kappa \\
T_3 &= \nu\Sigma^{-1} & \mathbb{E}\left[T_3\right] &= \nu\Psi^{-1} \\
T_4 &= -\log|\Sigma| & \mathbb{E}\left[T_4\right] &= \psi_d(\nu/2) + d\log 2 - \log|\Psi|
\end{aligned} \quad \Rightarrow$$

where $\psi_d$ is the multivariate digamma function.

If $u$ was a point mass, then the titled moments would be straightforward to calculate; just plug in the appropriate $\mu, \kappa, \nu, \Psi$ as function of $u$. Because we must integrate with respect to $\Gamma(u \,|\, \alpha, \alpha)$, we can approximate the integral with a Riemann sum. The moments can be calculated efficiently for a vector of $u$ by recognizing they all differ by at most a

rank-one update to the parameters $\mu, \kappa, \nu, \Psi$ and using the Woodbury matrix identity and determinant matrix lemma.

All that remains is to solve for the new posterior parameters by matching moments. This can be done by solving a system of equations. Note that for $\nu$, we must solve a 1-dimensional root finding problem to handle the digamma function $\psi_d$, which can be done quickly.

## *C.2 Time Series Clustering*

This section provides additional details for Sec. 5.3.2 on time series clustering. We describe how to calculate log-likelihoods using the Kalman smoother and how to calculate the posterior moments of $\eta$ for our EP approximation.

We consider time series clustering model is given by Eq. (5.9). For the rest of this section, we assume conditioning on all parameters except $x$, $z$, and $\eta$ (i.e. $a$, $\lambda$, $\sigma_x^2$, $\sigma_y^2$), unless otherwise noted. The Gibbs sampling distribution for these other likelihood parameters can be found in the appendix of Ren et al. (Ren et al., 2017).

### *C.2.1 Naive Log-likelihood*

Collapsing only $x$, the naive Gibbs sampler likelihood for $z_i$ is given by Eq. (5.10), which is

$$p(y_i|z_i, \eta) = \int \prod_{t=1}^{T} p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t})\, dx_t \tag{C.5}$$

By assumption, both the conditional distribution of $y_{i,t}$ given $x_{i,t}$ and the conditional distribution of $x_{i,t}$ given $x_{i,t-1}$ are Gaussian

$$p(y_{i,t} \,|\, x_{i,t}) = \mathcal{N}(y_{i,t} \,|\, x_{i,t}, \sigma_y^2) \tag{C.6}$$

$$p(x_{i,t} \,|\, x_{i,t-1}) = \mathcal{N}(x_{i,t} \,|\, a_i x_{i,t} + \lambda_i \eta_{z_i,t}, \sigma_x^2) \ . \tag{C.7}$$

The likelihood Eq. C.5 is then calculated using the Kalman filter (Bishop, 2006), which consists of iteratively applying 'predict' and 'update' steps. Due to the perturbations $\lambda_i \eta_{z_i,t}$ there is a slight adjustment in the predict step (Ren et al., 2017).

Let $\mu_{t|t-1}, \sigma^2_{t|t-1}$ denote the predictive mean and variance of $x_{i,t}$ given $y_{i,1:t-1}, \eta_{z_i}$ and let $\mu_{t|t}, \sigma^2_{t|t}$ denote the filtered mean and variance of $x_{i,t}$ given $y_{i,1:t}, \eta_{z_i}$. We can iteratively calculate the predictive and filtered parameters by applying 'predict' and 'update' steps.

The predict step is

$$\begin{aligned} \mu_{t|t-1} &= a_i \mu_{t-1|t-1} + \lambda_i \eta_{z_i,t} \\ \sigma^2_{t|t-1} &= a_i^2 \sigma^2_{t-1|t-1} + \sigma_x^2 \ . \end{aligned} \tag{C.8}$$

The update step is

$$\begin{aligned} \mu_{t|t} &= \mu_{t|t-1} + K_t \cdot (y_{i,t} - \mu_{t|t-1}) \\ \sigma^2_{t|t-1} &= (1 - K_t)\sigma^2_{t|t-1} \ . \end{aligned} \tag{C.9}$$

where $K_t$ is *Kalman* gain

$$K = \sigma^2_{t|t-1} / (\sigma^2_{t|t-1} + \sigma^2_{y_i}) \ .$$

We calculate the log-likelihood of $y_i$ by factorizing over time

$$\log p(y_i \mid z_i, \eta) = \sum_{t=1}^{T} \log p(y_{i,t} \mid y_{i,<t}, z_i, \eta) \ , \tag{C.10}$$

where $p(y_{i,t}|y_{i,<t}, z_i, \eta)$ is the Gaussian

$$p(y_{i,t}|y_{i,<t}, z_i, \eta) = \int p(y_{i,t}|x_{i,t}) p(x_{i,t}|y_{i,<t}, z_i, \eta)\, dx_{i,t} = \mathcal{N}(y_{i,t}|\mu_{t|t-1}, \sigma^2_{t|t-1} + \sigma^2_{y_i}) \ .$$

### *C.2.2 Collapsed Log-likelihood*

Collapsing both $x$ and $\eta$, the collapsed Gibbs sampler likelihood for $z_i$ is

$$p(y_i|z, y_{-i}) = \int\int \prod_{t=1}^{T} p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) \times p(\eta_{z_i} \mid y_{-i}, z_{-i})\ d\eta\ dx_{t:T} \ . \tag{C.11}$$

Although the distribution $p(\eta_{z_i}|y_{-i}, z_{-i})$ is known to be a $T$-dimensional multivariate Gaussian, computing its parameters and directly evaluating this integral, Eq. (C.11), is computationally prohibitive even for moderate sizes of $T$: inverting the covariance matrix requires $\mathcal{O}(T^3)$ computation.

Ren et al. (Ren et al., 2017) exploited the time-series structure of Fig. 5.1 (bottom-right) to calculate the collapsed likelihood by factorizing over time

$$p(y_i \mid z, y_{-i}) = \prod_{t=1}^{T} p(y_{i,t} \mid y_{-i,t}, y_{1:t-1}, z) \ , \tag{C.12}$$

where each conditional distribution in the product $p(y_{i,t}|y_{-i,t}, y_{1:t-1})$ can be calculated from the joint distribution

$$p(y_t \mid y_{1:t-1}) = \int p(y_t \mid x_t) p(x_t \mid y_{1:t-1}) \, dx_t \ .$$

Here, we let $y_t$ and $x_t$ denote the vector of values at time $t$ for series in cluster $z_i$. Recall that the values of other series $z_j \neq z_i$ are conditionally independent. The predictive distribution $p(x_t \mid y_{1:t-1})$ is calculated by the predict step of the multivariate generalization of the Kalman filter (Bishop, 2006; Ren et al., 2017).

Let $\mu_{t|t-1}, \Sigma_{t|t-1}$ denote the predictive mean and variance of $x_t$ given $y_{1:t-1}$ and let $\mu_{t|t}, \Sigma_{t|t}$ denote the filtered mean and variance of $x_t$ given $y_{1:t}$. We can iteratively calculate the predictive and filtered parameters by applying 'predict' and 'update' steps.

The predict step is

$$\begin{aligned} \mu_{t|t-1} &= A\mu_{t-1|t-1} \\ \Sigma_{t|t-1} &= A\Sigma_{t-1|t-1}A^T + \mathbb{I}\sigma_x^2 + \lambda\lambda^T \ , \end{aligned} \tag{C.13}$$

where $A = \mathrm{diag}(a)$. Note that the additional covariance term $\lambda\lambda^T$ couples the series together and is due to collapsing out $\eta_t$.

The update step is

$$\begin{aligned} \mu_{t|t} &= \mu_{t|t-1} + K_t \cdot (y_{i,t} - \mu_{t|t-1}) \\ \Sigma_{t|t-1} &= (\mathbb{I} - K_t)\Sigma_{t|t-1} \ . \end{aligned} \tag{C.14}$$

where $K_t$ is *Kalman* gain

$$K = \Sigma_{t|t-1} \cdot (\Sigma_{t|t-1} + \mathrm{diag}(\sigma_y^2))^{-1} \ .$$

Note that we must solve linear systems in the update step and in calculating the conditional likelihood. As these linear systems are of dimension $\mathcal{O}(N_k)$, practical numerical solvers have a runtime complexity $\mathcal{O}(N_k^3)$. As a result the full runtime complexity of evaluating Eq. (C.12) is $\mathcal{O}(TN_k^3)$ for each cluster assignment $z_i = k$.

### *C.2.3 EP Approximate Log-likelihood*

To approximate the collapsed likelihood, we use EP to keep track of a diagonal Gaussian approximations $q(\eta_k|z)$ for $\Pr(\eta_k|y,z)$. Because $q$ is diagonal, it factorizes over time

$$q(\eta_k|z) = \prod_{t=1}^{T} \mathcal{N}(\eta_{k,t} \,|\, \mu_{k,t}, \sigma^2_{k,t}) \ . \tag{C.15}$$

To calculate the cavity distribution $q(\eta_k|z_{-i})$, we remove the site approximation $\tilde{f}_i(\eta_k)$ from $q(\eta_k|z)$. This can be done by subtracting the natural parameters (mean-precision and precision).

If $\tilde{f}_i(\eta_k) = C_i \mathcal{N}(\eta_k|\mu_i, \sigma_i^2)$, then the mean and diagonal variance of the cavity distribution is

$$\mu_k^{(-i)} = {\sigma^2}_k^{(-i)} \cdot \left(\mu_k/{\sigma^2}_k - \mu_i/{\sigma^2}_i\right) \tag{C.16}$$

$${\sigma^2}_k^{(-i)} = \left(1/{\sigma^2}_k - 1/{\sigma^2}_i\right)^{-1} \ . \tag{C.17}$$

Our approximation for the collapsed likelihood is

$$\begin{aligned} p(y_i|z, y_{-i}) &\approx \int \prod_{t=1}^{T} \Big[ \int p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) \, dx_{i,t} \Big] \times q(\eta_{z_i}|z_{-i}) \, d\eta \\ &= \int \prod_{t=1}^{T} \Big[ \int p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) q(\eta_{z_i,t}|z_{-i}) \, d\eta_{z_i,t} \Big] \times p(y_{i,t} \,|\, x_{i,t}) \, dx_{i,t} \\ &= \int \prod_{t=1}^{T} \mathcal{N}\left(x_{i,t} \,|\, a_i x_{i,t} + \lambda_i \mu_{z_i,t}^{(-i)} \,,\, \sigma_x^2 + \lambda_i^2 {\sigma^2}_{z_i,t}^{(-i)}\right) \times \mathcal{N}(y_{i,t} \,|\, x_{i,t}, \sigma_y^2) \, dx_{i,t} \ . \end{aligned} \tag{C.18}$$

Note that the integral product of Eq. (C.18) (second line) is similar in form to the naive likelihood (Eq. (C.5)); both take the form

$$\int \prod_{t=1}^{T} p(y_{i,t}|x_{i,t})p(x_{i,t}|x_{i,t-1})\, dx_{i,t} \ .$$

The only difference (between Eq. (C.18) (third line) and Eq. (C.7)) is that latent process $x_i$ is 'smoothed' by marginalizing over the cavity distribution of $\eta_{z_i}$, the variance is a bit larger ($\lambda_i^2 \sigma^{2(-i)}_{z_i,t}$) and the mean shift uses $\mu^{(-i)}_{z_i,t}$, instead of using the point estimate $\eta_{z_i}$ from the previous iteration. Therefore, we can calculate our approximation with the univariate Kalman filter (Eqs.(C.8) and (C.9)) in $\mathcal{O}(T)$ time.

Our modified predict step (replacing Eq. (C.8)) is

$$\begin{aligned} \mu_{t|t-1} &= a_i \mu_{t-1|t-1} + \lambda_i \mu^{(-i)}_{z_i,t} \\ \sigma^2_{t|t-1} &= a_i^2 \sigma^2_{t-1|t-1} + \sigma_x^2 + \lambda_i^2 \sigma^{2(-i)}_{z_i,t} \ . \end{aligned} \tag{C.19}$$

### *C.2.4 EP Moment Update*

After selecting a new cluster assignment $z_i$, we update our likelihood approximation $\tilde{f}_i(\eta)$. We do this by selecting the parameters of $\tilde{f}_i(\eta)$ to minimize the local KL divergence between the *tilted* distribution $\tilde{p}(\eta|z)$

$$\tilde{p}(\eta|z) \propto f_i(z_i, \eta) q(\eta|z_{-i}) = p(y_i \mid z_i, \eta) q(\eta|z_{-i})$$

and the approximate distribution $q(\eta|z) \propto \tilde{f}_i(\eta) q(\eta|z_{-i})$. For Gaussian approximations (and more generally exponential families), minimizing the KL divergence is equivalent to matching the expected values of $q(\eta|z)$'s sufficient statistics. Because our approximation is a diagonal Gaussian, its sufficient statistics are the marginal means and variances at each time point $t$.

Therefore, we learn parameters of $q(\eta|z)$ to match the marginal means and variances of $\tilde{p}(\eta_t|z)$ and then solve for $\tilde{f}_i$ by removing the cavity distribution $q(\eta|z_{-i})$ from $q(\eta|z)$ in a similar manner to Eqs. (C.16) and (C.17).

Finally, the marginal mean and variance of the tilted distribution $\tilde{p}(\eta|z)$ can be efficiently calculated using the forward and backward messages passed in the Kalman smoother (Bishop, 2006).

The forward message is the filtered distribution of $x_t$ from the Kalman filter

$$\alpha(x_t) = p(x_t \,|\, y_{1:t}) \propto p(y_{1:t}, x_t) = \int p(y_t \,|\, x_t) p(x_t \,|\, x_{t-1}) \alpha(x_{t-1}) \; dx_{t-1}$$

where

$$p(x_t \,|\, x_{t-1}) = \int p(x_t \,|\, x_{t-1}, \eta_t) q(\eta_t) \; d\eta_t \;\; .$$

The backward message is the likelihood of future observations

$$\beta(x_t) = p(y_{t+1:T} \,|\, x_t) = \int \beta(x_{t+1}) p(y_{t+1} \,|\, x_{t+1}) p(x_{t+1} \,|\, x_t) \; dx_{t+1} \;\; .$$

Then, the marginal distribution at time $\tau$ of $\eta_\tau$ is

$$\begin{aligned}
\tilde{p}(\eta_\tau \,|\, z) &\propto \int p(y_i \,|\, z_i, \eta) q(\eta|z_{-i}) \; d\eta_{-\tau} \\
&= \int \left[ \int \prod_t p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) \, dx_i \right] q(\eta|z_{-i}) \, d\eta_{-\tau} \\
&= \int \Big[ \prod_{t<\tau} p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) q(\eta_{z_i,t}|z_{-i}) \\
&\qquad \times \; p(y_{i,\tau}|x_{i,\tau}) p(x_{i,\tau}|x_{i,\tau-1}, \eta_{z_i,\tau}) q(\eta_{z_i,\tau}|z_{-i}) \\
&\qquad \times \prod_{t>\tau} p(y_{i,t}|x_{i,t}) p(x_{i,t}|x_{i,t-1}, \eta_{z_i,t}) q(\eta_{z_i,t}|z_{-i}) \Big] \; dx_i d\eta_{-\tau} \\
&= \int \alpha(x_{i,\tau-1}) \times p(y_{i,\tau}|x_{i,\tau}) p(x_{i,\tau}|x_{i,\tau-1}, \eta_{z_i,\tau}) q(\eta_{z_i,\tau}|z_{-i}) \times \beta(x_{i,\tau}) \; dx_{i,\tau-1} dx_{i,\tau} \;\; .
\end{aligned} \tag{C.20}$$

All terms within the integral on the final line of Eq. (C.20) are Gaussian. Integrating out $x_{i,\tau}$ and $x_{i,\tau-1}$, gives us the tilted marginal distribution for $\eta_{z_i,\tau}$.

Thus we can calculate the mariginal means and variances by passing the same messages as the univariate Kalman smoother in $\mathcal{O}(T)$ time.

### *C.3 EP Convergence*

This section outlines convergence analysis of our approximate collapsed EP sampler. Our task is to analyze the approximation accuracy of our EP approximation $q^{(t)} = \prod_{k=1}^{K} q_k^{(t)}(\theta_k)$ when our target distribution, the posterior, is changing $p^{(t)} = p(\theta \mid y, z^{(t)})$. as it depends on the sampled assignment $z^{(t)}$ which varies with each iteration.

We first review what happens at each iteration and then discuss error bounds.

Suppose $q^{(t)}$ is our approximation for $p(\theta \mid y, z^{(t)})$. Our sampling algorithm proceeds as follows, for each iteration:

1. Select a latent assignment $z_i^{(t)}$ to reassign

2. Calculate the cavity distribution $q^{(t)}(\theta \mid z_{-i}^{(t)})$,

$$q_k^{(t)}(\theta \mid z_{-i}^{(t)}) = \begin{cases} q_k^{(t)}(\theta_k)/\tilde{f}_i^{(t)}(\theta_k) & \text{if } z_i^{(t)} = k \\ q_k^{(t)}(\theta_k) & \text{otherwise} \end{cases} , \tag{C.21}$$

   where $\tilde{f}_i^{(t)}$ is the site approximation $i$ at time $t$

3. Approximate the collapsed likelihood for each $z_i$ assignment

$$\tilde{p}(y_i \mid z_i, z_{-i}^{(t)}, y_{-i}) = \int p(y_i \mid z_i, \theta) \cdot q^{(t)}(\theta \mid z_{-i}^{(t)}) \tag{C.22}$$

4. Sample a new $z_i^{(t+1)} = k$ proportional to the prior and collapsed likelihood

5. Calculate the new site approximation $\tilde{f}_i^{(t+1)}$

$$\tilde{f}_i^{(t+1)} = \frac{\operatorname{argmin}_{q \in \mathcal{Q}} \mathrm{KL}\left(\tilde{p}_i(\theta, z_i^{(t+1)}) \,||\, q(\theta)\right)}{q_{z_i^{(t+1)}}^{(t)}(\theta \mid z_{-i}^{(t)})}$$

   where $\tilde{p}_i$ is the *tilted distribution*

$$\tilde{p}_i(\theta, z_i^{(t+1)}) = q_{z_i^{(t+1)}}^{(t)}(\theta \mid z_{-i}^{(t)}) \cdot f_i(\theta) \ . \tag{C.23}$$

6. Update the global approximations

$$q_k^{(t+1)} = \begin{cases} q_k^{(t)} / \tilde{f}_i^{(t)} & \text{if } z_i^{(t)} = k \text{ and } z_i^{(t+1)} \neq k \\ q_k^{(t)} \cdot \tilde{f}_i^{(t+1)} & \text{if } z_i^{(t)} \neq k \text{ and } z_i^{(t+1)} = k \\ q_k^{(t)} \cdot \tilde{f}_i^{(t+1)} / \tilde{f}_i^{(t)} = q_k^{(t)} & \text{if } z_i^{(t)} = z_i^{(t+1)} = k \\ q_k^{(t)} & \text{otherwise} \end{cases} .$$

Note that one step of our sampler only changes $q_{z_i^{(t)}}$, $q_{z_i^{(t+1)}}$ and $\tilde{f}_i$.

Outside of these iterations (steps 1-6), we periodically (e.g. after one scan through the data) run a full EP update without resampling a $z_i$.

*Error Bounds*

There are two types of error bounds that we could consider:

1. **Global** bounds on the approximation error $q_k^{(t)}(\theta_k)$ to the exact posterior $p(\theta_k | z^{(t)})$ for each cluster $k = 1, \ldots, K$

2. **Local** bounds on the site approximation $\tilde{f}_i^{(t)}$ to the optimal site approximations $\tilde{f}_i^*$ given $z^{(t)}$

Note that the local bounds are stronger than the global bounds, as the global approximation $q_k^{(t)}$ is the sum of the local approximations

$$\underbrace{q_k^{(t)}}_{\text{global approximation}} = \underbrace{p_0}_{\text{prior term}} + \underbrace{\sum_{i:z_i=k} \tilde{f}_i^{(t)}}_{\text{local approximation}} .$$

Suppose $z_i$ was selected to be resampled at step $(t)$. If $z_i$ does not change then, we have the standard EP update and its convergence guarantees (or lack thereof).

If $z_i$ changes between time $(t)$ and $(t+1)$, then our EP sampler only changes three terms: $q_{z_i^{(t)}}$, $q_{z_i^{(t+1)}}$ and $\tilde{f}_i$. All other approximations (i.e., $q_k$ for all $k \neq z_i^{(t)}, z_i^{(t+1)}$ and $\tilde{f}_j$ for $j \neq i$)

remain the same. The global approximation error only grows in the two cluster changed and the increase in error depends only on how well removing $\tilde{f}^{(t)}$ approximates the loss of $p(y_i|z_i, \phi)$ in cluster $z_i^{(t)}$ and how well adding $\tilde{f}_i^{(t+1)}$ approximates the addition of $p(y_i|z_i, \phi)$ in cluster $z_i^{(t+1)}$.

### *C.3.1 Empirical Experiments*

The section describes a series of experiments to quantify the error induced by only updating local sites compared against running full EP at each iteration. For this experiment, we consider components that are GSM.

$$\begin{aligned} p(y \mid \phi_k) &= (1-r) \cdot \mathcal{N}(y \mid \phi_k, \sigma^2) + r \cdot \mathcal{N}(y \mid \phi_k, C\sigma^2) \\ &= f(y \mid \phi_k, r, C, \sigma^2) \ . \end{aligned} \tag{C.24}$$

We measure the distance between our approximation $q^{(t)}$ and $q^*$ at time $(t)$ (by running EP to convergence when $z^{(t)}$ is fixed) using KL divergence and the percent error of recovering the posterior means and variances for $\phi$.

In our experiments, we vary the *proportion* probability $r$ from $[0, 0.5]$, the *mean difference* $\Delta = (\phi_1 - \phi_2)/\operatorname{Var}(y)$, and *scale ratio* $C$. Varying $r$ and $C$ determines how difficult the likelihood is to approximate with a site approximation, while varying $\Delta$ determines how rapidly $z$ changes. When $r = 0$, the problem is conjugate, so the error is zero and when $\Delta$ is large, $z$ rarely changes.

In all cases we find the error incurred by only using local updates does indeed level off (e.g. does not grow unbounded) as there number of iterations increase. Furthermore this size of this error depends on the setting $r, \Delta, C$.

Fig. C.1 presents KL results for $n = 100$ when starting the site approximations from the prior (i.e. flat). Note that after one pass, the approximation roughly level off (this includes the setting of $\Delta = 0$, where $z$ is rapidly changing).

Fig. C.2 presents KL results for $n = 100$ when starting the site approximations from full EP. In this case, the error grows until it levels off at the same constant KL as starting from

Table C.1: Median Percent Error and Median Absolute Percent Error (1 = 1%) for $q^{(t)}$'s mean and variance after a full pass through the data starting with flat site approximations. Standard deviation estimates are presented in parenthesis.

| | $C$ | $r$ | MeanPE | MeanAPE | VarPE | VarAPE |
|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| 2 | 2 | 0.2 | 0.05 (0.12) | 0.02 (0.1) | 0.03 (0.42) | 0.14 (0.3) |
| 3 | 2 | 0.5 | 0.02 (0.16) | 0.04 (1.43) | 0.05 (0.3) | 0.15 (0.2) |
| 4 | 5 | 0 | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| 5 | 5 | 0.2 | -0.01 (1.81) | 0.3 (1.47) | 0.31 (0.87) | 0.5 (0.55) |
| 6 | 5 | 0.5 | 3.6 (70.48) | 0.21 (62.94) | -0.31 (0.85) | 0.57 (0.56) |
| 7 | 10 | 0 | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| 8 | 10 | 0.2 | 1.56 (14.55) | 0.35 (13.08) | 0.72 (2.8) | 1.04 (2.06) |
| 9 | 10 | 0.5 | -0.2 (3.74) | 0.6 (3.11) | 1.27 (3.95) | 1.85 (2.82) |

flat approximations.

Finally Tables. C.1 and C.2, show the percent error and absolute percent error of the mean and variance for $\phi$ when $\Delta = 0.5$.

### *C.4 Synthetic Time Series Trace Plots*

In this section we provide additional plots showing the trace plots of the model parameters $A, \lambda, \sigma_x^2, \sigma_y^2, x$ for the synthetic data experiments in Sec. 5.3.2.

In Figs. C.7(a-d), we plot the mean squared error (MSE) between the sampled parameter $\hat{\theta}$ and the true parameter $\theta^*$ of the synthetic data. We can see that the collapsed sampler and our approximately collapsed EP sampler have similar performance. In Fig. C.8, we plot box-plots comparing 'collapsed' and 'EP', showing it accurately estimates both the mean and variance.

Table C.2: Median Percent Error and Median Absolute Percent Error ($1 = 1\%$) for $q^{(t)}$'s mean and variance after a full pass through the data. starting with full EP site approximations. Standard deviation estimates are presented in parenthesis.

| | $C$ | $r$ | MeanPE | MeanAPE | VarPE | VarAPE |
|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| 2 | 2 | 0.2 | 0.02 (1.32) | 0.02 (0.1) | 0 (0.24) | 0.14 (0.3) |
| 3 | 2 | 0.5 | 0.01 (0.05) | 0.04 (1.43) | 0.01 (0.23) | 0.15 (0.2) |
| 4 | 5 | 0 | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| 5 | 5 | 0.2 | -0.15 (0.53) | 0.3 (1.47) | 0.09 (0.69) | 0.5 (0.55) |
| 6 | 5 | 0.5 | 0.02 (1.84) | 0.21 (62.94) | -0.11 (2.11) | 0.57 (0.56) |
| 7 | 10 | 0 | 0 (0) | 0 (0) | 0 (0) | 0 (0) |
| 8 | 10 | 0.2 | 0.08 (1.46) | 0.35 (13.08) | -0.21 (1.69) | 1.04 (2.06) |
| 9 | 10 | 0.5 | -0.2 (8.4) | 0.6 (3.11) | 0.28 (3.52) | 1.85 (2.82) |

## *C.5 Seattle Housing Data*

This section provides additional details for the Seattle housing data example Sec. 5.4.2.

### *C.5.1 Data Details*

We use the same dataset as Ren et al. (Ren et al., 2017). This consists of 124,480 transactions in 140 US census-tracts of the city of Seattle from July 1997 to September 2013. The time index for each transaction is at the monthly level, therefore $T = 194$, with multiple observations for in certain series-month pairs, an no observations for other series-month pairs.

Each housing transaction contains the following house-specific covariates: (i) number of bathrooms, (ii) finished square-feet, and (iii) lot-size square-feet. We convert the house-specific covariates into feature variables by taking their log-values and applying $B$-splines with knots at their quartiles. Let $u_\ell$ denote the collection of features for house $\ell$.

### *C.5.2 Housing Price Model*

To predict housing prices, we copy the model used by Ren et al. (Ren et al., 2017)

$$x_{i,t} = a_i x_{i,t} + \lambda_i \eta_{z_i,t} + \epsilon_{i,t} \tag{C.25}$$

$$y_{i,t,\ell} = g_t + \beta_i u_\ell + x_{i,t} + \nu_{i,t,\ell} \ , \tag{C.26}$$

where $y_{i,t,\ell}$ denotes the log-price of house $\ell$ in region $i$ at time $t$.

The model for $y_{i,t,\ell}$, (Eq. (C.26)), consists of four parts: (i) a global housing price trend $g_t$ based on monthly seasonality, (ii) a series-specific regression $\beta_i u_\ell$, (iii) the latent residual process $x_{i,t}$, (iv) white noise $\nu_{i,t,\ell}$.

The global trend $g_t$ is removed in a preprocessing step by the following regression for parameters $\alpha_g$ and $\beta_g$

$$y_{i,t,\ell} \approx g_t = \alpha_g S(t) + \beta_g u_\ell \ , \tag{C.27}$$

where $S(t)$ is a smooth spline basis over time $t$. After learning $\alpha_g$ and $\beta_g$ in the preprocessing step, the global trend $g_t$ is fixed.

After removing the global trend, the residual process is modeled as the combination of region-specific regression and a latent AR(1) process. Inference over $\beta_i$ and $x_{i,t}$ as well as all other model parameters is achieved by Gibbs sampling. Ren et al. provide the complete Gibbs sampling formulas (Ren et al., 2017).

The difference between our two methods, collapsed and EP, is in how we sample the series cluster assignments $z_i$. For collapsed Gibbs, we run the expensive Kalman filter over individual clusters, while for EP Gibbs, we use the approximate likelihood described in Sec. 5.3.2 and Sec. C.2.

### *C.5.3 Additional Results*

We now present some diagnostics on the training data and the metrics of baseline models on the test data.

The other baseline models are:

- 'global', the global trend $g_t$ from Eq. (C.27).
- 'global+reg', the global trend $g_t$ plus individual series-specific regression $\beta_i u_\ell$.

The metrics on the training data are presented in Table C.3. The metrics on the test data are presented in Table C.4.

In both cases, the algorithms using the time series clustering model (collapsed and EP) vastly outperform the spline regression based models ('global' and 'global+regression').

Table C.3: Training metrics on Seattle housing data for different algorithms averaged over 10 initializations. Parenthetical values are one standard deviation.

| metric | collapsed | EP | global | global+reg |
|---|---|---|---|---|
| RMSE | 119230 (150) | 119270 (220) | 205380 | 202050 |
| Mean APE | 12.68 (0.01) | 12.69 (0.01) | 24.20 | 23.69 |
| Median APE | 9.50 (0.01) | 9.49 (0.01) | 18.60 | 18.00 |
| 90th APE | 27.07 (0.01) | 27.1 (0.02) | 50.35 | 49.04 |

Table C.4: Test metrics on Seattle housing data for different algorithms averaged over 10 initializations. Parenthetical values are one standard deviation.

| metric | collapsed | EP | global | global+reg |
|---|---|---|---|---|
| RMSE | 125280 (50) | 125280 (80) | 182150 | 180285 |
| Mean APE | 16.20 (0.01) | 16.20 (0.01) | 24.20 | 23.55 |
| Median APE | 12.07 (0.01) | 12.07 (0.01) | 18.59 | 18.17 |
| 90th APE | 34.17 (0.07) | 34.22 (0.05) | 50.48 | 49.31 |
| Runtime | 121.6 (8.1) | 62.8 (3.7) | - | - |

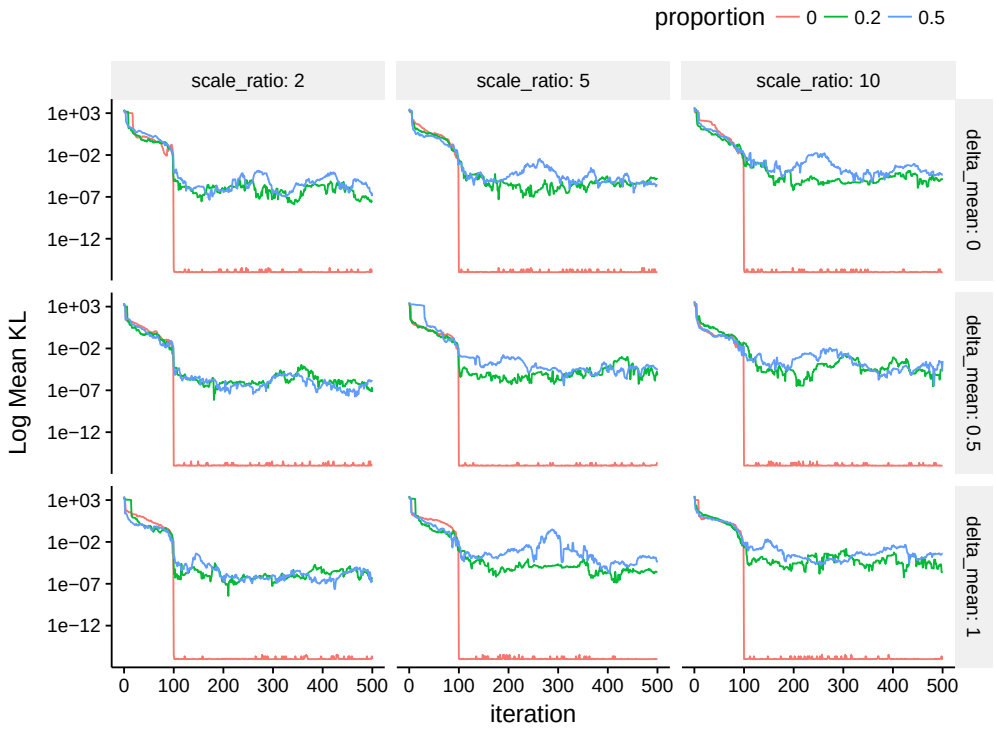


Figure C.1: KL error starting with flat site approximations.

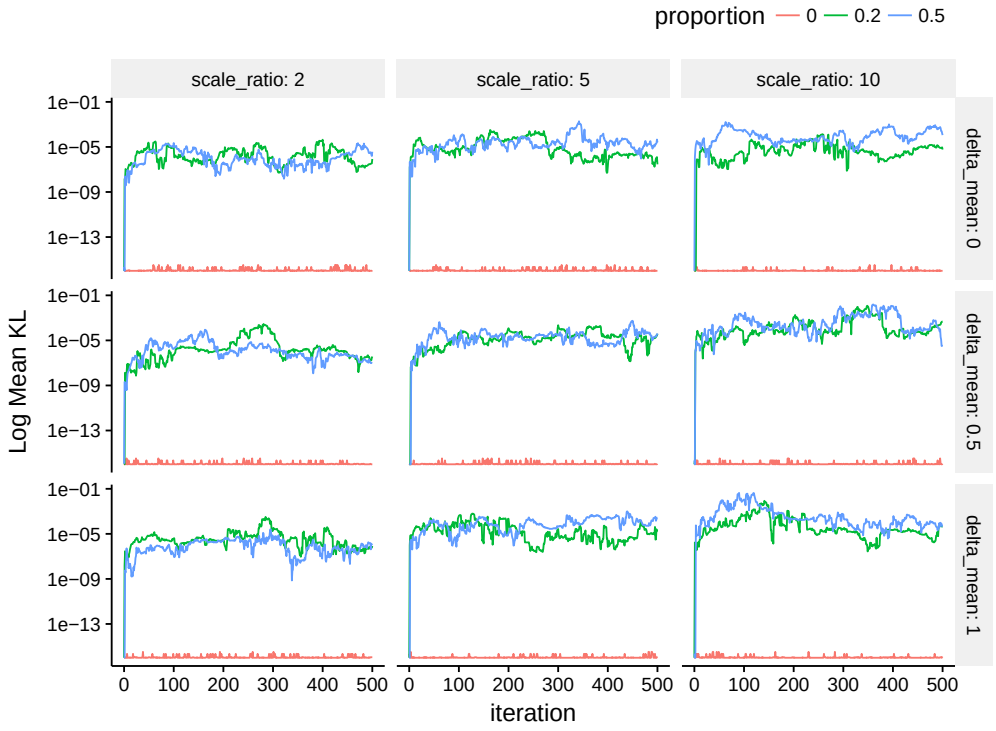


Figure C.2: KL error starting with site approximations from full EP.

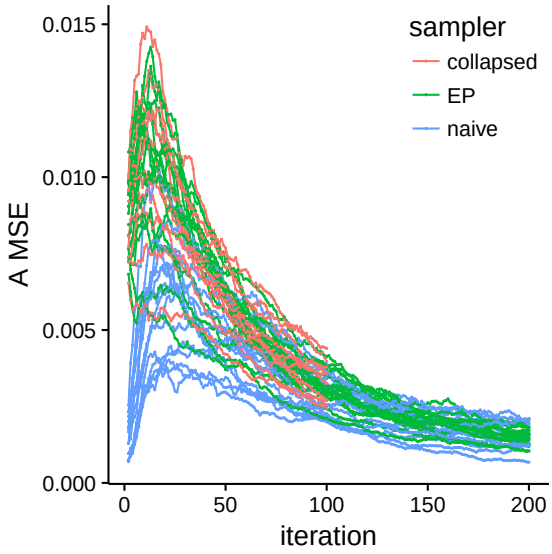


Figure C.3: *

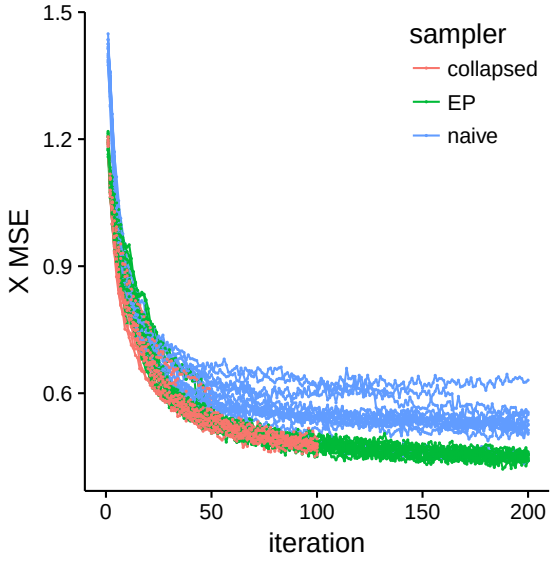


Figure C.4: *

(a) Trace of MSE $A$

(b) Trace of MSE $x$

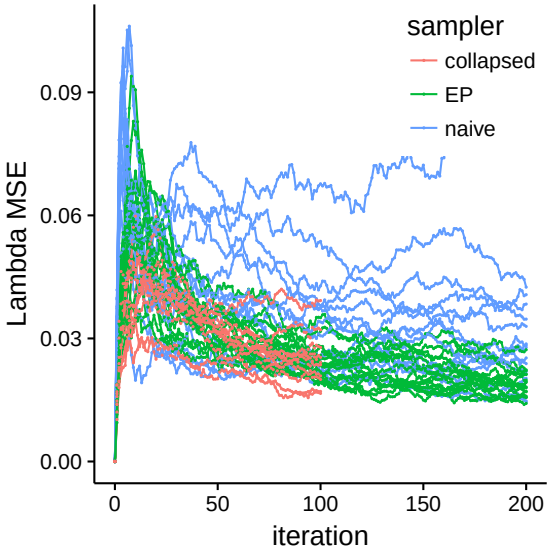


Figure C.5: *

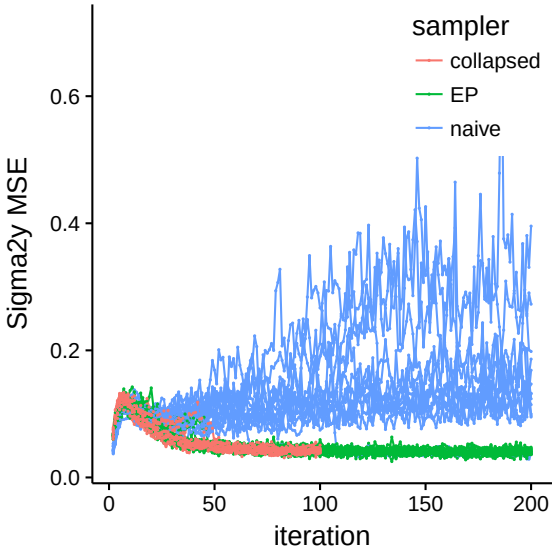


Figure C.6: *

(c) Trace of MSE $\lambda$

(d) Trace of MSE $\sigma_y^2$

Figure C.7: MSE Traceplots of the synthetic timeseries samplers

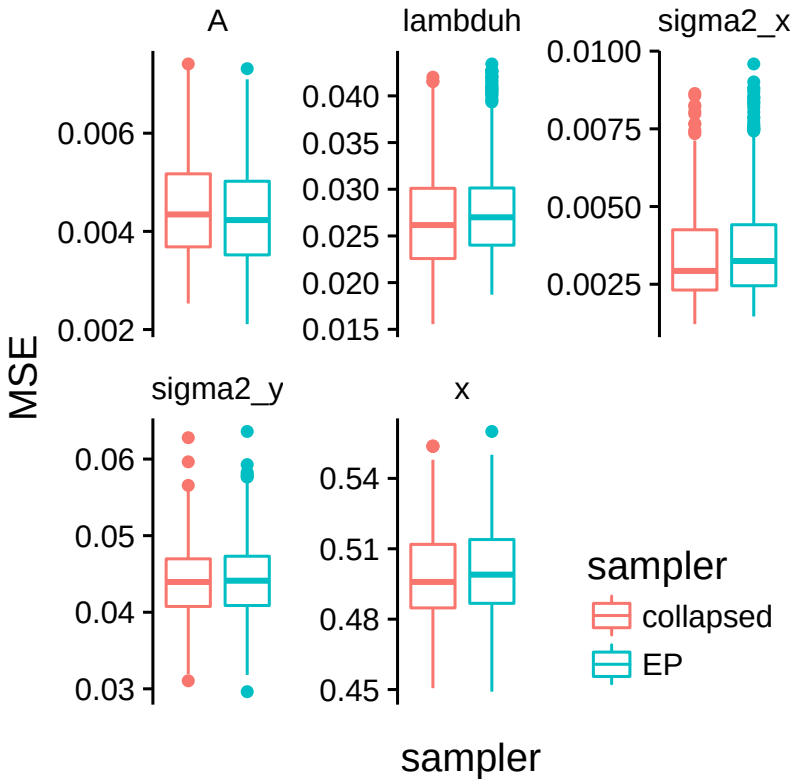


Figure C.8: Box-plot of posterior samples after burn-in