Asynchronous Gibbs sampling

Alexander Terenin
Imperial College London

Joint work with Dan Simpson and David Draper

AISTATS '20

June 17th, 2020

https://avt.im/ · @avt_im

Imperial College London

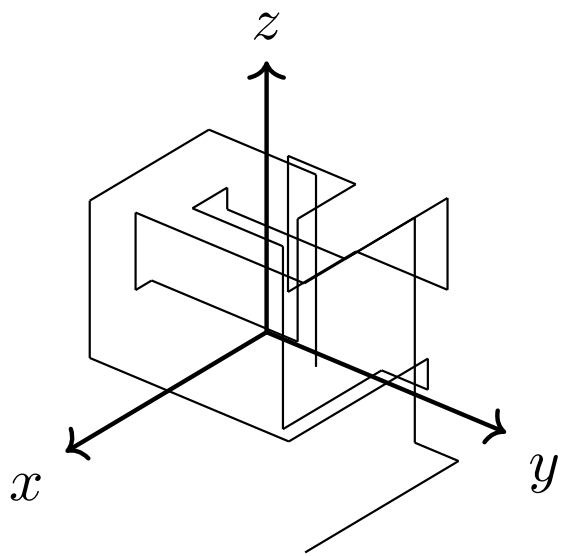
University of Toronto

Markov Chain Monte Carlo and Gibbs sampling

Sample from target π with density $f(x, y, z)$ using full conditionals

$$f(x \mid y, z) \qquad f(y \mid x, z) \qquad f(z \mid x, y)$$

one-by-one, just like optimization by cyclic coordinate descent.



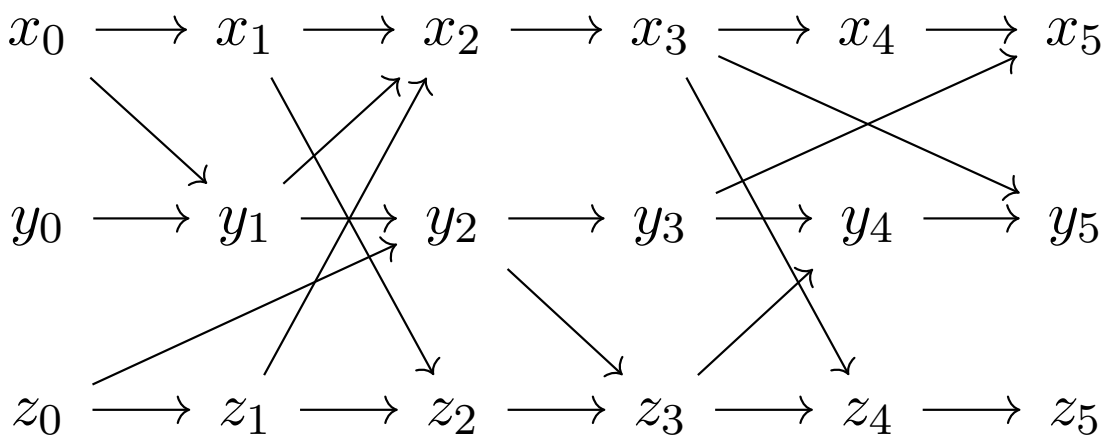
Under some regularity conditions, converges to π.

Parallelizing Gibbs sampling

To parallelize Gibbs, consider executing the full conditionals

$$f(x \mid y, z) \qquad f(y \mid x, z) \qquad f(z \mid x, y)$$

asynchronously, without waiting for previous update to finish.



Reduces synchronization costs for distributed systems ✓

Markov property is lost and convergence theory is unclear ☒

Does it work in practice?

Sometimes. Widely-used in topic modeling community.

Journal of Machine Learning Research 10 (2009) 1801-1828 Submitted 6/08; Revised 2/09; Published 8/09

Distributed Algorithms for Topic Models

David Newman NEWMAN@UCI.EDU
Arthur Asuncion ASUNCION@ICS.UCI.EDU
Padhraic Smyth SMYTH@ICS.UCI.EDU
Max Welling WELLING@ICS.UCI.EDU
Department of Computer Science
University of California, Irvine
Irvine, CA 92697, USA

Editor: Andrew McCallum

Abstract

We describe distributed algorithms for two widely-used topic models, namely the Latent Dirichlet Allocation (LDA) model, and the Hierarchical Dirichet Process (HDP) model. In our distributed algorithms the data is partitioned across separate processors and inference is done in a parallel, distributed fashion. We propose two distributed algorithms for LDA. The first algorithm is a straightforward mapping of LDA to a distributed processor setting. In this algorithm processors concurrently perform Gibbs sampling over local data followed by a global update of topic counts. The algorithm is simple to implement and can be viewed as an approximation to Gibbs-sampled LDA. The second version is a model that uses a hierarchical Bayesian extension of LDA to directly account for distributed data. This model has a theoretical guarantee of convergence but is more complex to implement than the first algorithm. Our distributed algorithm for HDP takes the straightforward mapping approach, and merges newly-created topics either by matching or by topic-id. Using five real-world text corpora we show that distributed learning works well in practice. For both LDA and HDP, we show that the converged test-data log probability for distributed learning is indistinguishable from that obtained with single-processor learning. Our extensive experimental results include learning topic models for two multi-million document collections using a 1024-processor parallel computer.

Keywords: topic models, latent Dirichlet allocation, hierarchical Dirichlet processes, distributed parallel computation

1. Introduction

Very large data sets, such as collections of images or text documents, are becoming increasingly common, with examples ranging from collections of online books at Google and Amazon, to the large collection of images at Flickr. These data sets present major opportunities for machine learning, such as the ability to explore richer and more expressive models than previously possible, and provide new and interesting domains for the application of learning algorithms.

However, the scale of these data sets also brings significant challenges for machine learning, particularly in terms of computation time and memory requirements. For example, a text corpus with one million documents, each containing one thousand words, will require approximately eight GBytes of memory to store the billion words. Adding the memory required for parameters, which usually exceeds memory for the data, creates a total memory requirement that exceeds that available

Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling

Christopher De Sa CDESA@STANFORD.EDU
Department of Electrical Engineering, Stanford University, Stanford, CA 94309
Kunle Olukotun KUNLE@STANFORD.EDU
Department of Electrical Engineering, Stanford University, Stanford, CA 94309
Christopher Ré CHRISMRE@STANFORD.EDU
Department of Computer Science, Stanford University, Stanford, CA 94309

Abstract

Gibbs sampling is a Markov chain Monte Carlo technique commonly used for estimating marginal distributions. To speed up Gibbs sampling, there has recently been interest in parallelizing it by executing asynchronously. While empirical results suggest that many models can be efficiently sampled asynchronously, traditional Markov chain analysis does not apply to the asynchronous case, and thus asynchronous Gibbs sampling is poorly understood. In this paper, we derive a better understanding of the two main challenges of asynchronous Gibbs: bias and mixing time. We show experimentally that our theoretical results match practical outcomes.

1. Introduction

Gibbs sampling is one of the most common Markov chain Monte Carlo methods used with graphical models (Koller & Friedman, 2009). In this setting, Gibbs sampling (Algorithm 1) operates iteratively by choosing at random a variable from the model at each timestep, and updating it by sampling from its conditional distribution given the other variables in the model. Often, it is applied to inference problems, in which we are trying to estimate the marginal probabilities of some query events in a given distribution.

For sparse graphical models, to which Gibbs sampling is often applied, each of these updates needs to read the values of only a small subset of the variables; therefore each update can be computed very quickly on modern hardware. Because of this and other useful properties of Gibbs sampling, many systems use Gibbs sampling to perform inference on big data (Lunn et al., 2009; McCallum et al., 2009; Newman et al., 2007; Smola & Narayanamurthy, 2010; Theis et al., 2012; Zhang & Ré, 2014).

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

Algorithm 1 Gibbs sampling

Require: Variables x_i for $1 \leq i \leq n$, and distribution π.
for $t = 1$ **to** T **do**
  Sample s uniformly from $\{1, \ldots, n\}$.
  Re-sample x_s uniformly from $\mathbf{P}_\pi(X_s | X_{\{1,\ldots,n\}\setminus\{s\}})$.
end for

Since Gibbs sampling is such a ubiquitous algorithm, it is important to try to optimize its execution speed on modern hardware. Unfortunately, while modern computer hardware has been trending towards more parallel architectures (Sutter, 2005), traditional Gibbs sampling is an inherently sequential algorithm; that is, the loop in Algorithm 1 is not directly parallelizable. Furthermore, for sparse models, very little work happens within each iteration, meaning it is difficult to extract much parallelism from the body of this loop. Since traditional Gibbs sampling parallelizes so poorly, it is interesting to study variants of Gibbs sampling that can be parallelized. Several such variants have been proposed, including applications to latent Dirichlet allocation (Newman et al., 2007; Smola & Narayanamurthy, 2010) and distributed constraint optimization problems (Nguyen et al., 2013).

In one popular variant, multiple threads run the Gibbs sampling update rule in parallel without locks, a strategy called *asynchronous* or HOGWILD! execution—in this paper, we use these two terms interchangeably. This idea was proposed, but not analyzed theoretically, in Smola & Narayanamurthy (2010), and has been shown to give empirically better results on many models (Zhang & Ré, 2014). But when can we be sure that HOGWILD! Gibbs sampling will produce accurate results? Except for the case of Gaussian random variables (Johnson et al., 2013), there

Does it work in practice?

Sometimes not. Shown to diverge on certain Gaussian targets.

Analyzing Hogwild Parallel Gaussian Gibbs Sampling

Matthew J. Johnson
EECS, MIT
mattjj@mit.edu

James Saunderson
EECS, MIT
jamess@mit.edu

Alan S. Willsky
EECS, MIT
willsky@mit.edu

Abstract

Sampling inference methods are computationally difficult to scale for many models in part because global dependencies can reduce opportunities for parallel computation. Without strict conditional independence structure among variables, standard Gibbs sampling theory requires sample updates to be performed sequentially, even if dependence between most variables is not strong. Empirical work has shown that some models can be sampled effectively by going "Hogwild" and simply running Gibbs updates in parallel with only periodic global communication, but the successes and limitations of such a strategy are not well understood.

As a step towards such an understanding, we study the Hogwild Gibbs sampling strategy in the context of Gaussian distributions. We develop a framework which provides convergence conditions and error bounds along with simple proofs and connections to methods in numerical linear algebra. In particular, we show that if the Gaussian precision matrix is generalized diagonally dominant, then any Hogwild Gibbs sampler, with any update schedule or allocation of variables to processors, yields a stable sampling process with the correct sample mean.

1 Introduction

Scaling probabilistic inference algorithms to large datasets and parallel computing architectures is a challenge of great importance and considerable current research interest, and great strides have been made in designing parallelizeable algorithms. Along with the powerful and sometimes complex new algorithms, a very simple strategy has proven to be surprisingly successful in some situations: running Gibbs sampling updates, derived only for the sequential setting, in parallel without globally synchronizing the sampler state after each update. Concretely, the strategy is to apply an algorithm like Algorithm 1. We refer to this strategy as "Hogwild Gibbs sampling" in reference to recent work [1] in which sequential computations for computing gradient steps were applied in parallel (without global coordination) to great beneficial effect.

This Hogwild Gibbs sampling strategy has long been considered a useful hack, perhaps for preparing decent initial states for a proper serial Gibbs sampler, but extensive empirical work on Approximate Distributed Latent Dirichlet Allocation (AD-LDA) [2, 3, 4, 5, 6], which applies the strategy to generate samples from a collapsed LDA model, has demonstrated its effectiveness in sampling LDA models with the same predictive performance as those generated by standard serial Gibbs [2, Figure 3]. However, the results are largely empirical and so it is difficult to understand how model properties and algorithm parameters might affect performance, or whether similar success can be expected for any other models. There have been recent advances in understanding some of the particular structure of AD-LDA [6], but a thorough theoretical explanation for the effectiveness and limitations of Hogwild Gibbs sampling is far from complete.

Sampling inference algorithms for complex Bayesian models have notoriously resisted theoretical analysis, so to begin an analysis of Hogwild Gibbs sampling we consider a restricted class of models that is especially tractable for analysis: Gaussians. Gaussian distributions and algorithms are tractable because of their deep connection with linear algebra. Further, Gaussian sampling is of

Not much general convergence theory

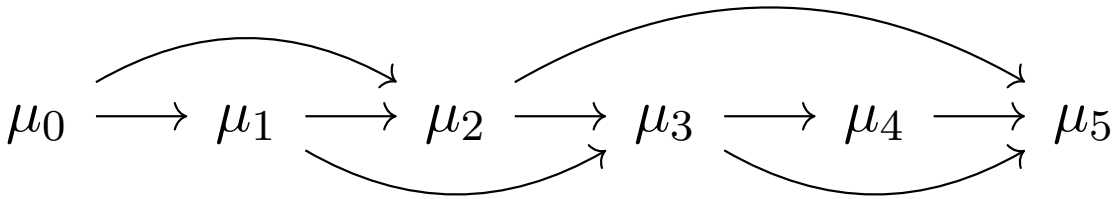
Asynchronous convergence theory

This work: study conditions under which asynchronous convergence reduces to the standard sequential case.

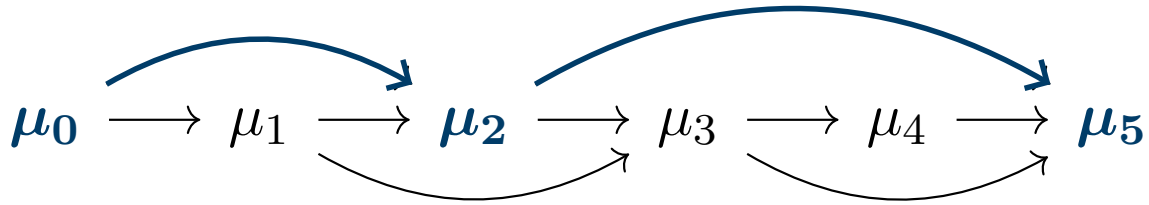
How? A Markov chain x_i can be seen as a random algorithm over X *or* as a deterministic algorithm over the measure space $\mathcal{M}_1(X)$.

$$x_0 \longrightarrow x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow x_4 \longrightarrow x_5$$

$$\mu_0 \longrightarrow \mu_1 \longrightarrow \mu_2 \longrightarrow \mu_3 \longrightarrow \mu_4 \longrightarrow \mu_5$$



Asynchronous convergence theory



THEOREM.

atomic state reads/writes
+ bounded asynchronicity

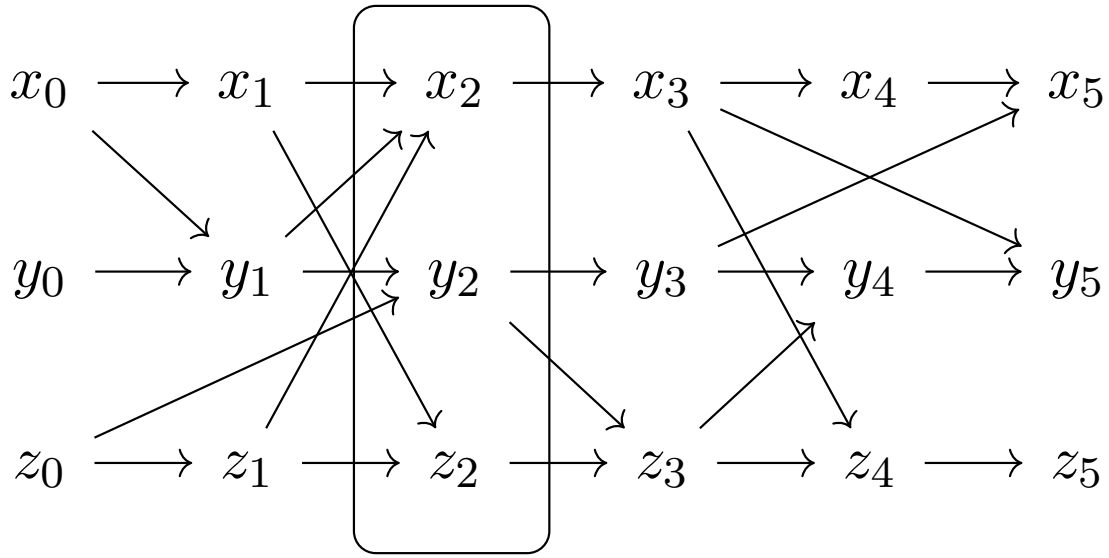
$\Longrightarrow$

sequential convergence *lifts*
to asynchronous convergence

Why? Roughly: the asynchronous chain *contains* a sequential chain.

Asynchronous Gibbs sampling

Gibbs sampling: reads/writes are only atomic on *parts* of the state.

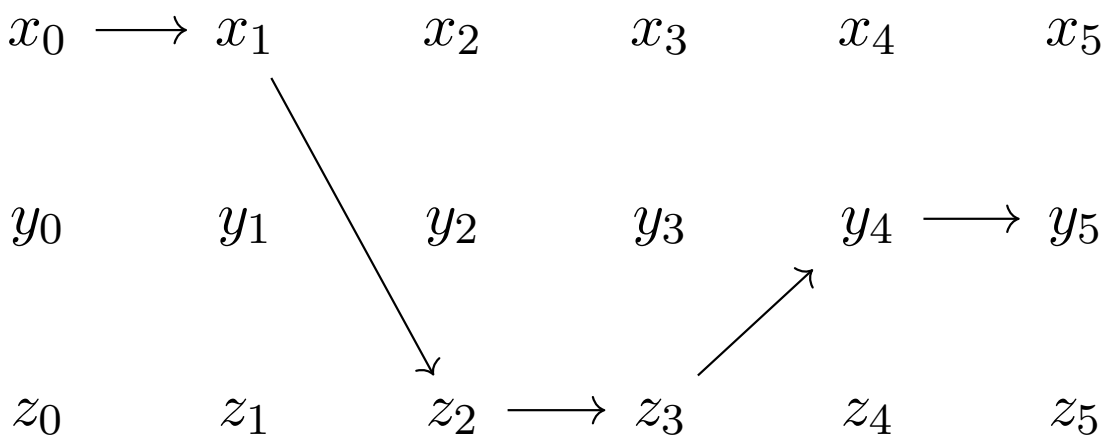


Key idea: place asynchronous Gibbs within the general asynchronous compute theory of Baudet and Bertsekas.

What is the underlying sequential chain? Does it converge?

Asynchronous Gibbs sampling

Idea: construct an instantaneous parallel Gibbs sampler, representing asynchronous algorithm under instantaneous communication.



Introduce a Metropolis–Hastings step, with acceptance probability 1 in sequential case, but *not necessarily 1* in asynchronous case.

Call this new chain the *exact asynchronous Gibbs sampler*.

Asynchronous Gibbs sampling

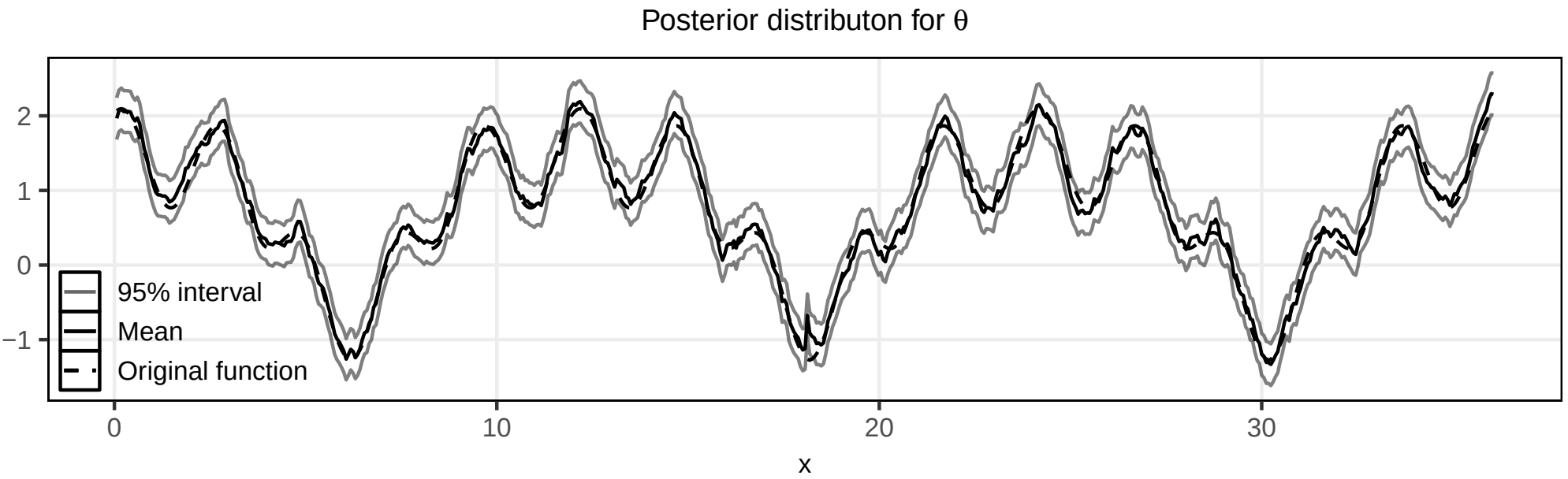
THEOREM. (INFORMAL) Let $\mathbf{X}^k$ be an exact asynchronous Gibbs sampler. Suppose under instantaneous communication that $\mathbf{X}^k$ converges sufficiently quickly as $k \to \infty$. Then $\mathbf{X}^k$ also converges asynchronously in the sense of Baudet and Bertsekas.

No explicit conditions on target distribution ✓
Conditions on chain and cluster are restrictive ☒

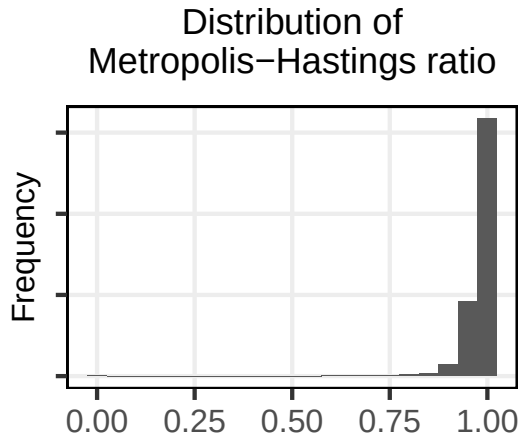
Analysis reduces to sequential case ✓
Metropolis–Hastings ratio: new convergence diagostic ✓

Experiments: settings where asynchronous Gibbs works fine

Gaussian process regression



A hierarchical model for mixed-effects regression

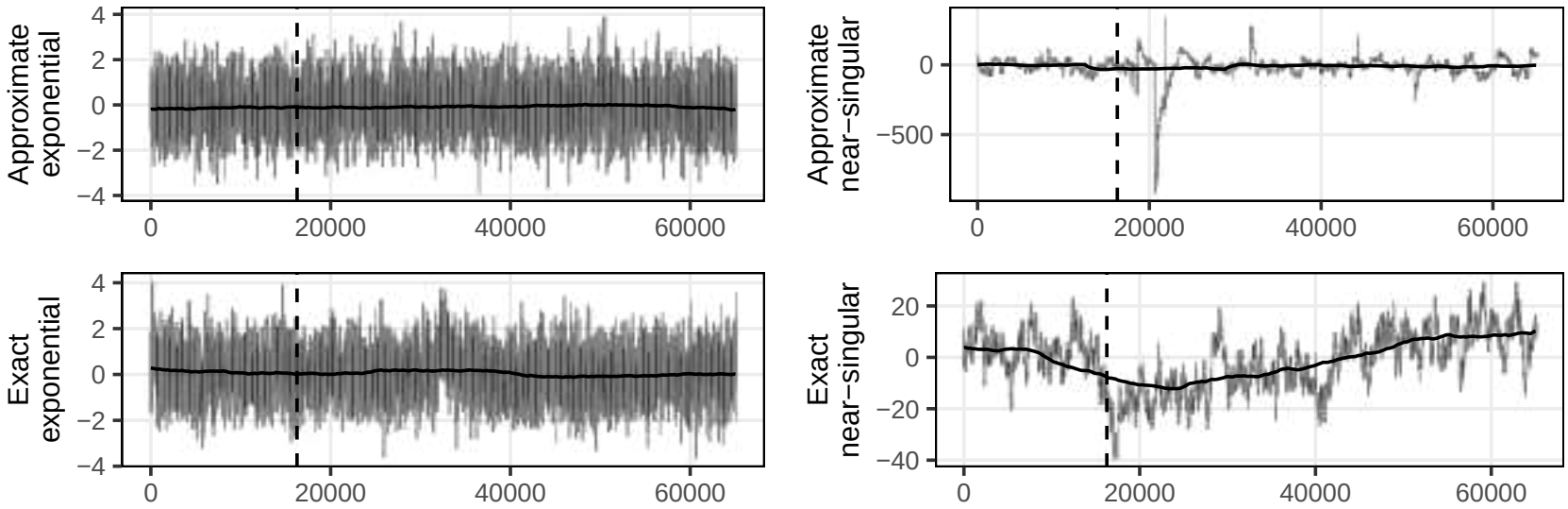


Experiments: settings where asynchronous Gibbs fails

Targets: two different 8-dimensional Gaussians with covariances

$$\Sigma^{(e)}_{ij} = \exp\left(-\frac{|i-j|}{2}\right) \qquad \Sigma^{(s)}_{ij} = \begin{cases} 87.5 & i = j, \\ -12.5 & i \neq j. \end{cases}$$

Works fine for $\boldsymbol{\Sigma}^{(e)}$, and *very badly* for $\boldsymbol{\Sigma}^{(s)}$.
Metropolis–Hastings step helps significantly.

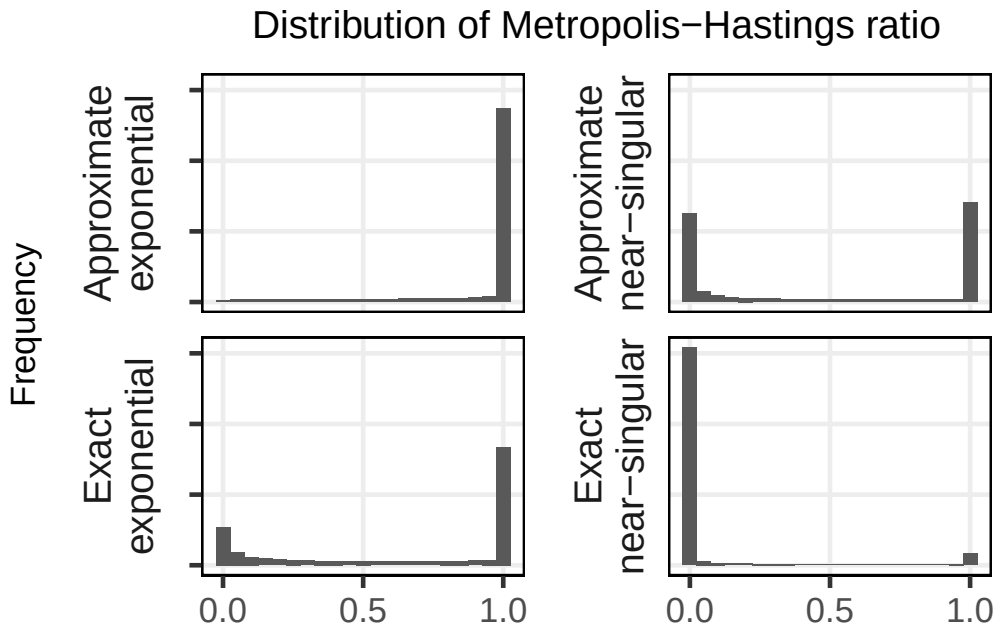


Experiments: settings where asynchronous Gibbs fails

Targets: two different 8-dimensional Gaussians with covariances

$$\Sigma^{(e)}_{ij} = \exp\left(-\frac{|i-j|}{2}\right) \qquad \Sigma^{(s)}_{ij} = \begin{cases} 87.5 & i = j, \\ -12.5 & i \neq j. \end{cases}$$

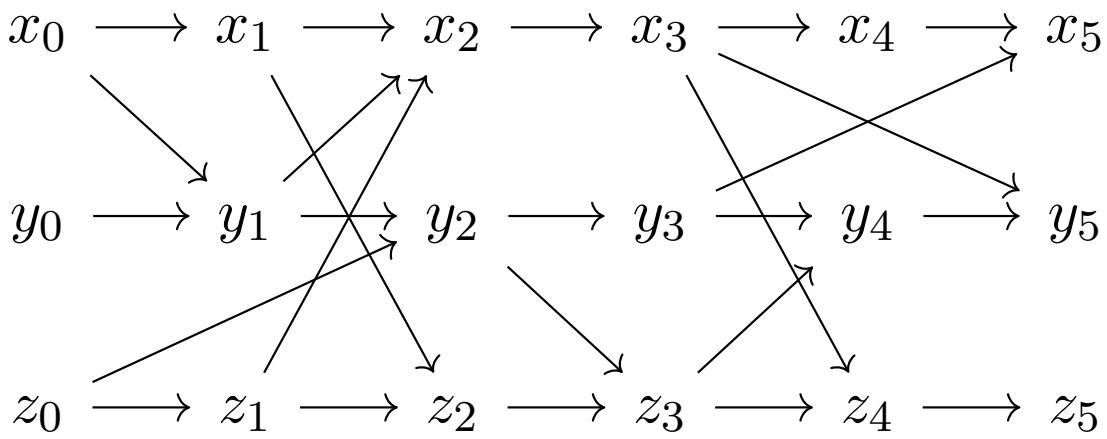
Works fine for $\boldsymbol{\Sigma}^{(e)}$, and *very badly* for $\boldsymbol{\Sigma}^{(s)}$.
Metropolis–Hastings step helps significantly.



Takeaways

Asynchronous MCMC behavior: complex and not well-understood.

- Our work: analysis using ordinary MCMC theory is *possible*.
- The theory is target-generic, at a cost of strong regularity requirements on algorithm and distributed system.



Practitioners: I recommend considering asynchronous Gibbs as a method-of-last-resort only when there are no other options.

Concluding remarks

Thank you for your attention!

https://avt.im/ · @avt_im

Imperial College London

University of Toronto

A. Terenin, D. Simpson, D. Draper. Asynchronous Gibbs sampling. AISTATS, 2020. Available at: https://arxiv.org/abs/1509.08999