Asynchronous Gibbs Sampling

Alexander Terenin (eBay, Inc. and UC Santa Cruz)
Dan Simpson (University of Bath, UK) · David Draper (UC Santa Cruz)

Abstract

Bayesian statistical models are attractive in the Big Data setting, but Markov Chain Monte Carlo methods – the cornerstone of modern Bayesian computation – do not extend easily to the cluster setting, where the data is too big to fit on one machine. We present a novel scheme to parallelize MCMC with no synchronization or locking, avoiding typical performance bottlenecks, particularly in settings where the number of model parameters grows with data size.

Big Data: the problem

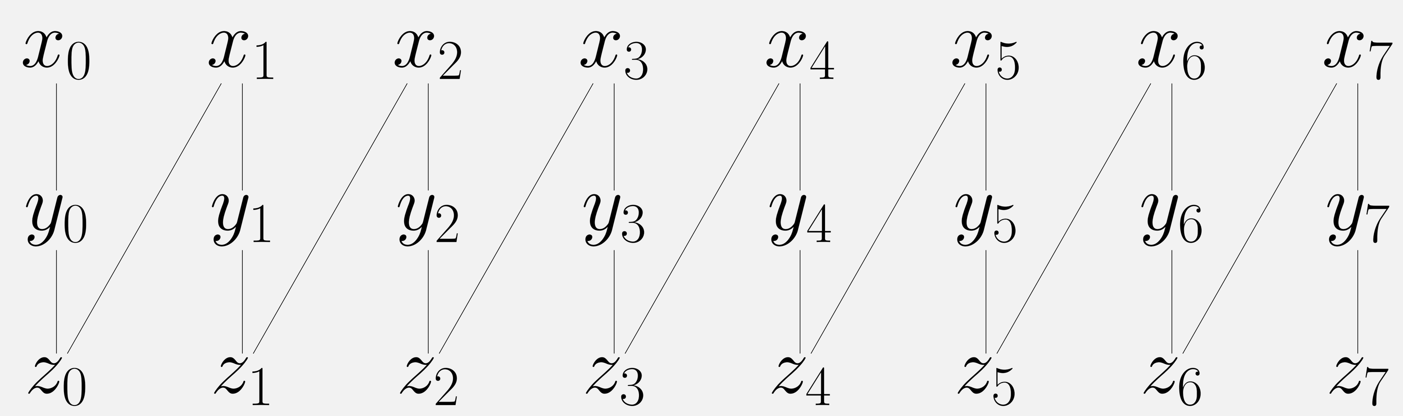
Data: *too big to fit on one machine*

- Lives on a Hadoop cluster: fault-tolerant distributed storage
  - Big Data $\Longleftrightarrow$ Big Hardware
- Computational cost of *1 data point* ≈ cost of *all N* points
  - Data *must* be operated on *in parallel*
  - Need parallel algorithms that scale *superlinearly*
- Model dimensionality often grows with N

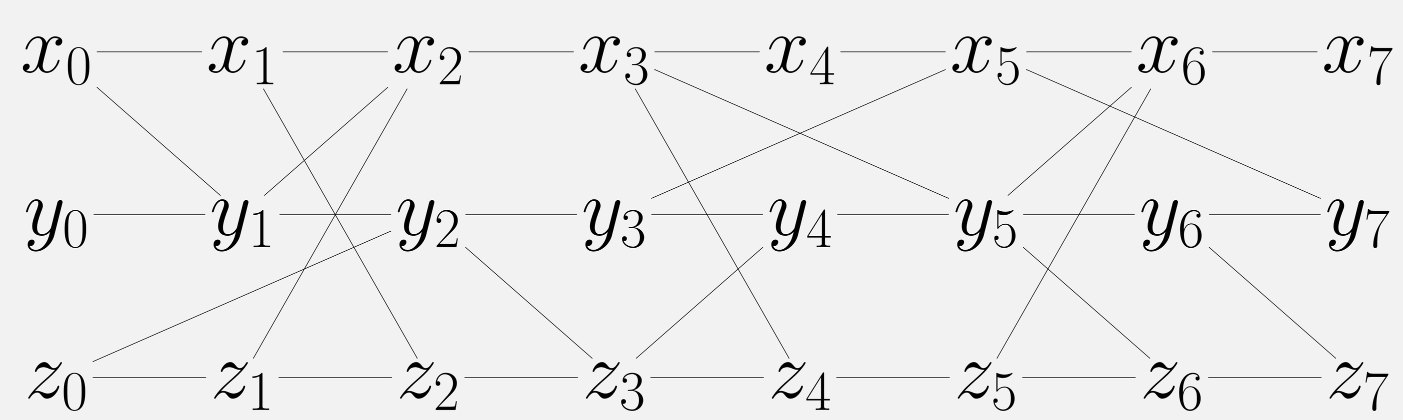
Previous Work on Scalable Bayesian Inference

- Data that fits on *one machine*
  - Approximate Methods: Variational Bayes [2], ABC [6]
  - Exact Methods: Langevin Diffusion based methods [4]
- Data *too big for one machine*
  - *Consensus Monte Carlo* [5]: for models where dimensionality doesn't grow with N (exact for some models)
  - *Hogwild*: [3] asynchronous stochastic gradient descent, point estimation only, widely used in machine learning

Standard Gibbs Sampling



Asynchronous Gibbs Sampling



The Algorithm

1. Start with a *latent variable* model, partition data among workers
2. Run Gibbs steps on each worker
   - Condition on *most recent local variables*
   - Condition on *most recent known non-local* variables, which *may be out of date* in the cluster as a whole
   - *Draw* samples, *transmit* to other workers
   - Never stop, never synchronize, never wait

Implementation

- Highly non-trivial: not expressible in the MapReduce paradigm (typical high-level parallel computation framework used in Hadoop)
- Expressible in shared memory and actor models of parallelism
- Written in Scala, a modern language well-suited to parallel use cases
- Code available via eBay Software Foundation on GitHub

Convergence

Two kinds of algorithms: different communication between workers

- *Exact Asynchronous Gibbs*: apply Metropolis-Hastings correction
- *Approximate Asynchronous Gibbs*: accept all updates

Theorem: *Exact Asynchronous Gibbs converges*

- Intuitive sketch of proof
  1. *Instant communication $\Longrightarrow$ exact algorithm converges*
     - Define a Markov chain that (1) selects a worker, (2) selects a full conditional, (3) proposes a new state from that full conditional at every worker, (4) performs a Metropolis step on each worker
     - Proof via detailed balance, workers accept/reject independently
  2. *Exact algorithm converges $\Longrightarrow$ asynchronous convergence*
     - Proof via result on convergence of asynchronous algorithms

Approximate Asynchronous Gibbs: Noisy Monte Carlo [1]

- Replace Metropolis-Hastings ratio with *biased estimator – 1*
- Perform *diagnostic check* at runtime to ensure *bias is small*

Future Work

- Better understanding of Noisy Monte Carlo [1] approximations
- Implement algorithm in *MPI* for use on traditional supercomputers

Results

Gaussian Process Regression: *71,500 latent variables* (toy problem)

- *Correct answer* in *20 minutes* on 143-core cluster

Large hierarchical model: *1,000,000 latent variables* (real-world)

- *Same answer* as sequential-scan Gibbs (up to Monte Carlo error)

Standard Gibbs Sampler 8-core machine, high memory, problem-specific parallelization	*12 hrs*
Asynchronous Gibbs 160-core cluster	*1 hr*

References

[1] P. Alquier, N. Friel, R. Everitt, and A. Boland. Noisy Monte Carlo: Convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47, 2016.
[2] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.
[3] F. Niu, B. Recht, C. Re, and S. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.
[4] G. O. Roberts and O. Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.
[5] S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch. Bayes and Big Data: The Consensus Monte Carlo algorithm. In *Bayes 250*. Vol. 16, 2013.
[6] S. Tavaré, D. J. Balding, R. C. Griffiths, and P. Donnelly. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518, 1997.

Acknowledgments

We are grateful to UC Santa Cruz and the Baskin School of Engineering for funding this research, and to eBay, Inc. for funding and providing computational resources.