Exponential Stochastic Cellular Automata for Massively Parallel Inference

Manzil Zaheer
Carnegie Mellon University,
manzil@cmu.edu

Michael Wick
Oracle Labs,
michael.wick@oracle.com

Jean-Baptiste Tristan
Oracle Labs,
jean.baptiste.tristan@oracle.com

Alex Smola
Carnegie Mellon University,
alex@smola.org

Guy L. Steele Jr.
Oracle Labs,
guy.steele@oracle.com

Abstract

We propose an embarrassingly parallel, memory efficient inference algorithm for latent variable models in which the complete data likelihood is in the exponential family. The algorithm is a stochastic cellular automaton and converges to a valid *maximum a posteriori* fixed point. Applied to latent Dirichlet allocation we find that our algorithm is over an order of magnitude faster than the fastest current approaches. A simple C++/MPI implementation on a 4-node cluster samples 570 million tokens per second. We process 3 billion documents and achieve predictive power competitive with collapsed Gibbs sampling and variational inference.

1 Introduction

In the past decade, frameworks such as stochastic gradient descent (SGD) [18] and map-reduce [5] have enabled machine learning algorithms to scale to larger and larger datasets. However, these frameworks are not always applicable to Bayesian latent variable models with rich statistical dependencies and intractable gradients. Variational methods [12] and Markov chain Monte-Carlo (MCMC) [7] have thus become the *sine qua non* for inferring the posterior in these models.

Sometimes—due to the concentration of measure phenomenon associated with large sample sizes—computing the full posterior is unnecessary and *maximum a posteriori* (MAP) estimates suffice. It is hence tempting to employ gradient descent, but for latent variable models such as latent Dirichlet allocation (LDA), calculating gradients involves expensive expectations over rich sets of variables [17]. MCMC is an appealing alternative, but algorithms such as the Gibbs sampler are inherently sequential and the extent to which they can be parallelized depends heavily upon how the structure of the statistical model interacts with the data. For instance, chromatic sampling [8] is infeasible for LDA, due to its dependence structure. Instead, we employ stochastic cellular automata (SCA), which like conventional cellular automata are massively parallel, but with stochastic updates.

We propose exponential SCA (ESCA) for inference in latent variable models with complete data likelihood in the exponential family. ESCA is embarassingly parallel because it is an SCA, and has a minimal memory footprint because it stores only the data and the sufficient statistics (by the very definition of sufficient statistics, the footprint cannot be further reduced). In contrast, variational approaches such as stochastic variational inference (SVI) [11] require storing the variational parameters, while MCMC-based methods, such as YahooLDA [20] require storing the latent variable assignments. Furthermore, our algorithm employs double-buffering for lock-free parameter updates (assuming atomic increments) while enabling the use of approximate counters. Thus, we substantially reduce memory costs and communication requirements in distributed enviroments.

2 Exponential SCA

Stochastic cellular automata (SCA), also known as probabilistic cellular automata, or locally-interacting Markov chains, are a stochastic version of a discrete-time, discrete-space dynamical system in which a noisy local update rule is homogeneously and synchronously applied to every site of a discrete space. They have been studied in statistical physics, mathematics, and computer science, and some progress has been made toward understanding their ergodicity and equilibrium properties. A recent survey [14] is an excellent introduction to the subject, and a dissertation [13] contains a comprehensive and precise presentation of SCA. Formally, the automaton, is given by an evolution function $\Phi : \mathcal{S} \longrightarrow \mathcal{S}$ over the state space $\mathcal{S} = \mathcal{Z} \longrightarrow C$ which is a mapping from the space of cell identifiers Z to cell values C. The global evolution function applies a local function $\phi_{\mathfrak{z}}(c_1, c_2, \cdots, c_r) \mapsto c$ s.t. $c_i = s(\mathfrak{z}_i)$ to every cell $\mathfrak{z} \in \mathcal{Z}$. That is, ϕ examines the values of each of the neighbors of cell $\mathfrak{z}$ and then stochastically computes a new value c. The dynamics begin with a state $s_0 \in S$ that can be configured using the data or a heuristic. Exponential SCA (ESCA) is based on SCA but achieves better computational efficiency by exploiting the structure of the sufficient statistics for latent variable models in which the complete data likelihood is in the exponential family. Most importantly, the local update function ϕ for each cell depends only upon the sufficient statistics and thus does *not* scale linearly with the number of neighbors.

2.1 Latent Variable Exponential Family

Latent variable models are useful when reasoning about partially observed data such as collections of text or images in which each *i.i.d.* data point is a document or image. Since the same local model is applied to each data point, they have the following form

$$p(\mathbf{z}, \mathbf{x}, \eta) = p(\eta) \prod_i p\left(z_i, x_i | \eta\right). \qquad (1)$$

Our goal is to obtain a MAP estimate for the parameters η that explain the data $\mathbf{x}$ through the latent variables $\mathbf{z}$. To expose maximum parallelism, we want each cell in the automaton to correspond to a data point and its latent variable. However, this is problematic because in general all latent variables depend on each other via the global parameters η and a naive approach to updating a single cell would then require examining every other cell in the automaton.

Fortunately, if we further suppose that the complete data likelihood is in the exponential family, i.e., $p(z_i, x_i|\eta) = \exp\left(\langle T\left(z_i, x_i\right), \eta\rangle - g(\eta)\right)$ then the sufficient statistics are given by $T(\mathbf{z}, \mathbf{x}) = \sum_i T(z_i, x_i)$ and we can thus express any estimator of interest as a function of just $T(\mathbf{z}, \mathbf{x})$ which factorizes over the data. Further, when employing expectation maximization (EM), the M-step is possible in closed form for many members of the exponential family. This allows us to reformulate the cell level updates to depend only upon the sufficient statistics instead of the neighboring cells. The idea is that, unlike SCA (or MCMC in general) which produces a sequence of states that correspond to complete variable assignments $s^0, s^1, \ldots$ via a transition kernel $q(s^{t+1}|s^t)$, ESCA produces a sequence of sufficient statistics $T^0, T^1, \ldots$ directly via an evolution function $\Phi(T^t) \mapsto T^{t+1}$.

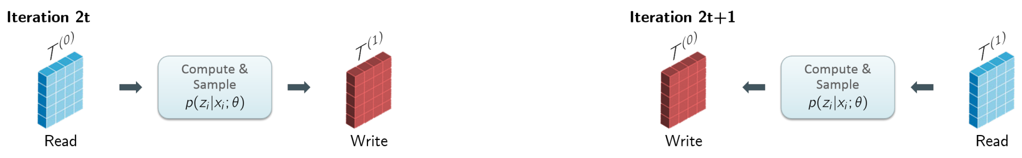
2.2 Stochastic EM

Before we present ESCA, we first describe stochastic EM (SEM). Suppose we want the MAP estimate for η, $\max_\eta p(\mathbf{x}, \eta) = \max_\eta \int p(\mathbf{z}, \mathbf{x}, \eta)\mu(d\mathbf{z})$ and employ expectation maximization (EM):

E-step Compute in parallel $p(z_i|x_i, \eta^{(t)})$.
M-step Find $\eta^{(t+1)}$ that maximizes the expected log-likelihood with respect to the conditional

$$\eta^{(t+1)} = \arg\max_\eta \mathbb{E}_{\mathbf{z}|\mathbf{x},\eta^{(t)}}\left[\log p(\mathbf{z}, \mathbf{x}, \eta)\right] = \xi^{-1}\left(\frac{1}{n+n_0}\sum_i \mathbb{E}_{\mathbf{z}|\mathbf{x},\eta^{(t)}}\left[T(z_i, x_i)\right] + T_0\right)$$

where $\xi(\eta) = \nabla g(\eta)$ is invertible as $\nabla^2 g(\theta) \succ 0$ and n_0, T_0 parametrize the conjugate prior. Although EM exposes substantial parallelism, it is difficult to scale, since the dense structure $p(z_i|x_i, \eta^{(t)})$ defines values for all possible outcomes for z and thus puts tremendous pressure on memory bandwidth. To overcome this we introduce sparsity by employing stochastic EM (SEM) [3]. SEM introduces an S-step after the E-step that replaces the full distribution with a single sample:



(a) Phase 1 (b) Phase 2

Figure 1: Efficient (re)use of buffers

S-step Sample $z_i^{(t)} \sim p(z_i|x_i;\eta^{(t)})$ in parallel.

Subsequently, we perform the M-step using the imputed data instead of the expectation. This simple modification overcomes the computational drawbacks of EM for cases in which sampling from $p(z_i|x_i;\eta^{(t)})$ is feasible. We can now employ fast samplers, such as the alias method, exploit sparsity, reduce CPU-RAM bandwidth while still maintaining massive parallelism. More importantly, the S-step also enables all three steps to now be expressed in terms of the current sufficient statistics. This enables distributed and parallel implementations that efficiently execute on an SCA.

2.3 ESCA for Latent Variable Models

We now present ESCA as SEM on an SCA in which each cell corresponds to a data point with its associated latent variables. Define an SCA over the state space $\mathcal{S}$ of the form $\mathcal{S} = \mathcal{Z} \longrightarrow \mathcal{K} \times \mathcal{X}$, where $\mathcal{Z}$ is the set of cell identifiers (e.g., one per data point), $\mathcal{K}$ is the domain of latent variables, and $\mathcal{X}$ is the domain of the observed data. The initial state s_0 is the map defined as follows: for every data point, we associate a cell z to the pair (k_z, x) where k_z is chosen at random from $\mathcal{K}$ and independently from $k_{z'}$ for all $z' \neq z$. This gives us the initial state $s_0 = z \mapsto (k_z, x)$.

We now need to describe the evolution function Φ. For state s and cell z define the distribution:

$$p_z(k|s) = f(z, T(s)) \tag{2}$$

Assuming that $s(z) = (k, x)$ and that k' is a sample from p_z (hence the name "stochastic" cellular automaton) we define the local update function as $\phi(s, z) = (k', x)$ where $s(z) = (k, x)$ and $k' \sim p_z(\,\cdot\,|s)$. That is, the observed data remain unchanged, but we choose a new latent variable according to the distribution p_z induced by the state. We obtain the evolution function of the stochastic cellular automaton by applying the function ϕ uniformly on every cell $\Phi(s) = z \mapsto \phi(s, z)$. Finally, the SCA algorithm simulates the evolution function Φ starting with s_0. We remark that ESCA converges weakly to a distribution with mean equal to some root of the score function ($\nabla_\eta \log p(x_i;\eta)$) and thus a MAP fixed point. See Appendix D for details.

Our implementation has two copies of the data structure containing sufficient statistics $T^{(0)}$ and $T^{(1)}$. We do not compute the values $T(\mathbf{z}, \mathbf{x})$ but maintain their sum as we impute values of the cells/latent variables. During iteration $2t$ of the evolution function, we apply Φ by reading from $T^{(0)}$ and incrementing $T^{(1)}$ as we sample the latent variables (Figure 1). Then in the next iteration $2t+1$ we reverse the roles of data structure, i.e. read from $T^{(1)}$ and increment $T^{(0)}$. See Algorithm 1.

Algorithm 1 ESCA

```
1: Randomly initialize each cell
2: for $t = 0 \to$ num iterations do
3:     for all cell $z$ independently in parallel do
4:         Read sufficient statistics from $T^{(t \bmod 2)}$
5:         Compute stochastic updates using $p_z(k|s)$
6:         Write sufficient statistics to $T^{(t+1 \bmod 2)}$
7:     end for
8: end for
```

Use of such read/write buffers offer a virtually lock-free (assuming atomic increments) implementation scheme for ESCA and is analogous to double-buffering in computer graphics. Although there is a synchronization barrier after each round, its effect is mitigated because each cell's work depends only upon the sufficient statistics and thus does the same amount of work. Therefore, evenly balancing the work load across computation nodes is trivial, even for a heterogeneous cluster.

3 ESCA for LDA

Latent Dirichlet allocation (LDA) [1] is a must-have for analytic platforms and consequently needs to scale. LDA models each document m of M documents as a distribution θ_m over K topics. A topic k is a distribution ϕ_k over V vocabulary words. A document m comprises N_m words w_{mn} each with a latent variable z_{mn} indicating a topic assignment. Both distributions θ_m and ϕ_k have a Dirichlet prior, parameterized respectively with a constant α and β. See Appendix B for details.

ESCA simulates the inference steps of SEM which we derive for LDA in Appendix B. For LDA, the state space is $\mathcal{S} = \mathcal{Z} \longrightarrow \mathcal{K} \times \mathcal{M} \times \mathcal{V}$ where $\mathcal{Z}$ is the set of cell identifiers (one per token in our corpus), $\mathcal{K}$ is a set of K topics, $\mathcal{M}$ is a set of M document identifiers, and $\mathcal{V}$ is a set of V identifiers for the vocabulary words. From this we obtain the full conditional of LDA for line 5 of Algorithm 1

$$p_z(k|s) \propto \boxed{(D_{mk} + \alpha) \times \frac{W_{kv} + \beta}{T_k + \beta V}} \tag{3}$$

where $D_{mk} = \left|\left\{ z_{mn} \mid z_{mn} = k \right\}\right|$, $W_{kv} = \left|\left\{ z_{mn} \mid w_{mn} = v,\, z_{mn} = k \right\}\right|$, and $T_k = \sum_{v=1}^{V} W_{kv}$. It reassuring to see that the boxed region of Equation 3 is similar to respective formulas in collapsed Gibbs sampling (CGS) [9] and collapsed variational Bayes (CVB0) [21]. For LDA, ESCA implicitly performs SGD with Frank-Wolfe updates, alluding to a convergence rate (Appendix C).

4 Experiments

Software & hardware All algorithms are implemented in C++11. We implement multithreaded parallelization within a node using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We implement ESCA with a sparse arrays D counting topics per documents and Vose's alias method to draw from discrete distributions. We run our experiments on a small cluster of 4 nodes connected through 10Gb/s Ethernet. Each node has two 9-core Intel Xeon E5 processors for a total of 36 hardware threads per node.

Datasets We employ three datasets: PubMed abstracts (141,043 vocabulary words, 8.2 million documents, 737 million tokens), Wikipedia (210,223 words, 6.6 million documents, 1.1 billion tokens) and a large proprietary dataset (140,000 words, 3 billion documents, and 171 billion tokens).

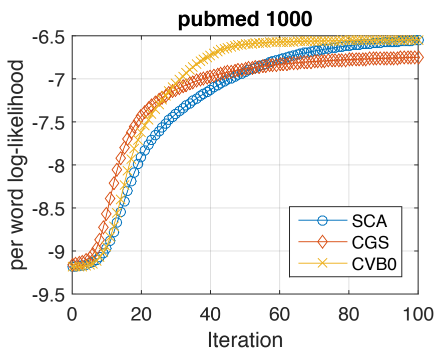
Evaluation To evaluate the proposed method we use predicting power as a metric by calculating the per-word log-likelihood (equivalent to negative log of perplexity) on 10,000 held-out documents conditioned on the trained model. We set $K = 1000$ to demonstrate performance for a large number of topics. The hyper parameters are set as $\alpha = 50/K$ and $\beta = 0.1$ as suggested in [10]; other systems such as YahooLDA and Mallet also use this as the default parameter setting. The results are in Figure 2 and additional experiments are in Appendix H. Finally, for the large dataset, our implementation of ESCA (only 300 lines of C++) processes 570 million tokens per second (tps) on our modest 4-node cluster. In comparison, some of the best existing systems achieve 112 million tps (F+LDA, personal communication) and 60 million tps (lightLDA) [25]. See Table 1 for details.

Table 1: Comparison with existing scalable LDA frameworks.

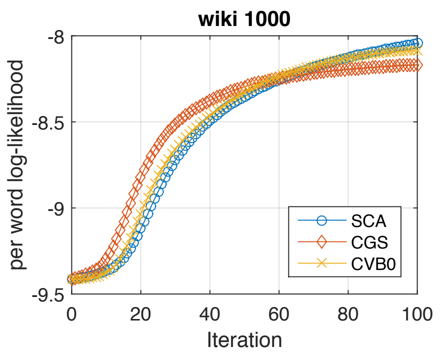
Method	Dataset	Infrastructure	Processing speed
YahooLDA [20]	140K vocab, 8.2M docs, 797M tokens	10 machines 2010	12.87M tokens/s
lightLDA [25]	50K vocab, 1.2B docs, 200B tokens	24 machines 2014	60M tokens/s
F+LDA [24]	1M vocab, 29M docs, 1.5B tokens	32 machines 2014	110M tokens/s
ESCA	**210K vocab, 6B docs, 128B tokens**	**8 Amazon c4.8x large**	**503M tokens/s**

Discussion

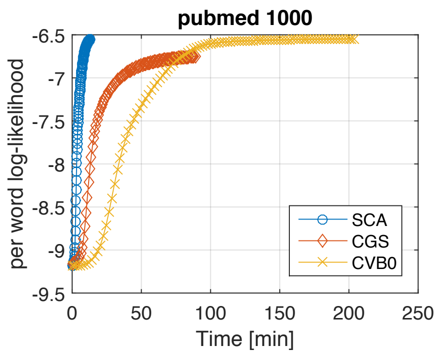
We proposed an embarassingly parallel, memory efficient MAP inference algorithm that executes on an SCA and applies to a large class of latent variable models. Our algorithm exposes many system level optimizations such as approximate counters, and outperforms current best approaches.



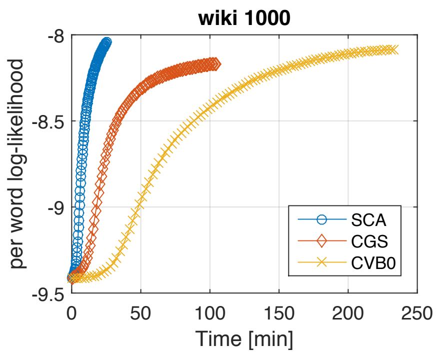
(a) PubMed, $K=1000$, $\alpha=0.05$, $\beta=0.1$



(b) Wikipedia, $K=1000$, $\alpha=0.05$, $\beta=0.1$



(c) Pubmed, $K=1000$, $\alpha=0.05$, $\beta=0.1$



(d) Wikipedia, $K=1000$, $\alpha=0.05$, $\beta=0.1$

Figure 2: Evolution of log likelihood on Wikipedia and Pubmed over number of iterations and time.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[2] J. Canny. Gap: a factor model for discrete data. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 122–129. ACM, 2004.

[3] Gilles Celeux and Jean Diebolt. The sem algorithm: a probabilistic teacher algorithm derived from the em algorithm for the mixture problem. *Computational statistics quarterly*, 2(1):73–82, 1985.

[4] Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804, Beijing, China, July 2015. Association for Computational Linguistics.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[6] Anton K Formann and Thomas Kohlmann. Latent class analysis in medical research. *Statistical methods in medical research*, 5(2):179–211, 1996.

[7] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1995.

[8] Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: from colored fields to thin junction trees. In *International Conference on Artificial Intelligence and Statistics*, pages 324–332, 2011.

[9] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proc. National Academy of Sciences of the United States of America*, 101(suppl 1):5228–5235, 2004.

[10] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[11] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, May 2013.

[12] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, November 1999.

[13] Pierre-Yves Louis. *Automates Cellulaires Probabilistes : mesures stationnaires, mesures de Gibbs associées et ergodicité*. PhD thesis, Université des Sciences et Technologies de Lille and il Politecnico di Milano, September 2002.

[14] Jean Mairesse and Irène Marcovici. Around probabilistic cellular automata. *Theoretical Computer Science*, 559:42–72, November 2014.

[15] R. Neal. Markov chain sampling methods for dirichlet process mixture models. Technical Report 9815, University of Toronto, 1998.

[16] Søren Feodor Nielsen. The stochastic em algorithm: estimation and asymptotic results. *Bernoulli*, pages 457–489, 2000.

[17] 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.

[18] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[19] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. Relationship between gradient and em steps in latent variable models.

[20] Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Proc. VLDB Endowment*, 3(1-2):703–710, September 2010.

[21] Whye Yee Teh, David Newman, and Max Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 19*, NIPS 2006, pages 1353–1360. MIT Press, 2007.

[22] Max A Woodbury, Jonathan Clive, and Arthur Garson. Mathematical typology: a grade of membership technique for obtaining disease definition. *Computers and biomedical research*, 11(3):277–298, 1978.

[23] Lei Xu and Michael I Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural computation*, 8(1):129–151, 1996.

[24] Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, SVN Vishwanathan, and Inderjit S Dhillon. A scalable asynchronous distributed algorithm for topic modeling. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1340–1350. International World Wide Web Conferences Steering Committee, 2015.

[25] Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1351–1361. International World Wide Web Conferences Steering Committee, 2015.