Exponential Stochastic Cellular Automata for Massively Parallel Inference

Manzil Zaheer Carnegie Mellon
Michael Wick Oracle Labs
Jean-Baptiste Tristan Oracle Labs
Alex Smola Carnegie Mellon
Guy L Steele Jr Oracle Labs

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 or 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) [28] and map-reduce [8] have enabled machine learning algorithms to scale to larger and large datasets. However, these frameworks are not always applicable to Bayesian latent variable models with rich statistical dependencies and intractable gradients. Variational methods [15] and Markov chain Monte-Carlo (MCMC) [10] 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. However, for latent variable models such as latent Dirichlet allocation (LDA), calculating gradients involves expensive expectations over rich sets of variables [27].

MCMC is an appealing alternative, but traditional 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 [11] is infeasible for LDA, due to its dependence structure. We propose an alternate approach based on stochastic cellular automata (SCA). The automaton is massively parallel like conventional cellular automata, but employs stochastic updates.

Our proposed algorithm, exponential SCA (ESCA), is a specific way of mapping inference in latent variable models with complete data likelihood in the exponential family into an instance of SCA. ESCA 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) [14] require storing the variational parameters, while MCMC-based methods, such as YahooLDA [30] require storing the latent variables. Thus, ESCA substantially reduces memory costs, enabling larger datasets to fit in memory, and significantly reducing communication costs in distributed environments.

Furthermore, the sufficient statistics dramatically improves efficiency. Typically, updating a cell requires first assembling the values of all the neighboring cells before aggregating them into a local stochastic update. In ESCA, the sufficient statistics adequately summarize the states of the neighbors; the computational load is small and perfectly balanced across the cells.

We demonstrate how ESCA is a flexible framework for exponential latent variable models such as LDA. In our experiments, we process over 3 billion documents at a rate of 570 million tokens per second on a small cluster of 4 commodity servers. That said, ESCA is much more general. Table 1 explicitly lists some of the more common modeling choices for which ESCA can be easily employed. Our algorithm implicitly simulates stochastic expectation maximization (SEM), and is thus provably correct in the sense that it converges in distribution to a stationary point of the *posterior*.

Preliminary work. Under review by AISTATS 2016. Do not distribute.

Table 1: Examples of some popular models to which ESCA is applicable.

mix. component/emitter	Bernoulli	Multinomial	Gaussian	Poisson
Categorical	Latent Class Model [9]	Unigram Document Clustering	Mixture of Gaussians [22]	
Dirichlet Mixture	Grade of Membership Model [36]	Latent Dirichlet Allocation [2]	Gaussian-LDA [6]	GaP Model [3]

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 [19] is an excellent introduction to the subject, and a dissertation [18] contains a comprehensive and precise presentation of SCA.

The automaton, as a (stochastic) discrete-time, discrete-space dynamical system, 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

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

Our goal is to obtain a MAP estimate for the parameters η that explain the data $\mathbf{x}$ through the latent variables $\mathbf{z}$. However, in general all latent variables depend on each other via the global parameters η, and thus the local evolution function ϕ would have to examine the values of every cell in the automaton.

Fortunately, if we further suppose that the complete data likelihood is in the exponential family, i.e.,

$$p(z_i, x_i | \eta) = \exp\left(\langle T\left(z_i, x_i\right), \eta\rangle - g(\eta)\right) \tag{2}$$

then the complete and sufficient statistics are given by

$$T(\mathbf{z}, \mathbf{x}) = \sum_i T(z_i, x_i) \tag{3}$$

and we can thus express any estimator of interest as a function of just $T(\mathbf{z}, \mathbf{x})$. 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 must first describe stochastic EM (SEM). Suppose we want the MAP estimate for η and employ a traditional expectation maximization (EM) approach:

$$\max_\eta p(\mathbf{x}, \eta) = \max_\eta \int p(\mathbf{z}, \mathbf{x}, \eta)\mu(d\mathbf{z})$$

EM finds a mode of $p(\mathbf{x}, \eta)$ by iterating two steps:

E-step Compute in parallel $p(z_i | x_i, \eta^{(t)})$.

M-step Find $\eta^{(t+1)}$ that maximizes the expected value of the log-likelihood with respect to the conditional probability, i.e.

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

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) [4]. SEM substitutes the E-step for an S-step that replaces the full distribution with a single sample:

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.

The S-step has other important consequences. Notice that the M-step is now a simple function of current sufficient statistics. This implies that the conditional distribution for the next S-step is expressible in terms of the complete sufficient statistics

$$p(z_i|x_i;\eta^{(t)}) = f(z_i, T(\mathbf{x}, \mathbf{z}^{(t)})).$$

Thus each S-step depends only upon the sufficient statistics generated by the previous step. Therefore, we can operate directly on sufficient statistics without the need to assign or store latent variables/states. Moreover it opens up avenues for distributed and parallel implementations that execute on an SCA.

2.3 ESCA for Latent Variable Models

SEM produces an alternating sequence of S and M steps in which the M step produces the parameters necessary for the next S step. Since we can compute these parameters on the fly there is no need for an explicit M step. Instead, ESCA produces a sequence consisting only of S steps. We require the exponential family to ensure that these steps are both efficient and compact. We now present ESCA more formally.

Define an SCA over the state space $\mathcal{S}$ of the form:

$$\mathcal{S} = \mathcal{Z} \longrightarrow \mathcal{K} \times \mathcal{X} \tag{4}$$

where $\mathcal{Z}$ is the set of cell identifiers (e.g., one per token in a text corpus), $\mathcal{K}$ 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.

$$s_0 = z \mapsto (k_z, x) \tag{5}$$

We now need to describe the evolution function Φ. First, assuming that we have a state s and a cell z, we define the following distribution:

$$p_z(k|s) = f(z, T(s)) \tag{6}$$

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:

$$\begin{gathered} \phi(s, z) = (k', x) \\ \text{where} \quad s(z) = (k, x) \quad \text{and} \quad k' \sim p_z(\,\cdot\,|s) \end{gathered} \tag{7}$$

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.

$$\Phi(s) = z \mapsto \phi(s, z) \tag{8}$$

Finally, the SCA algorithm simulates the evolution function Φ starting with s_0.

As explained earlier, due to our assumption of complete data likelihood belonging to the exponential family, we never have to represent the states explicitly, and instead employ the sufficient statistics.

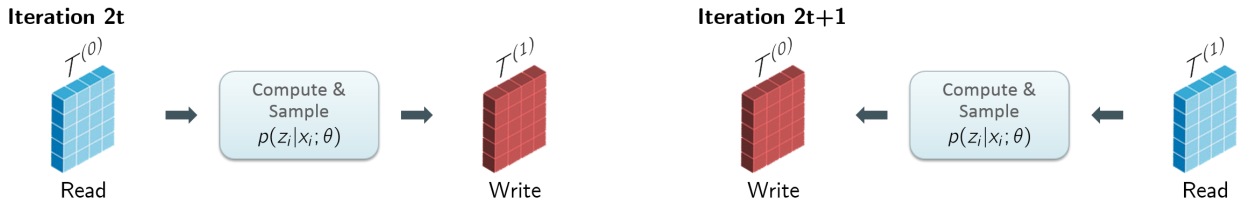
An implementation can, for example, have two copies of the data structure containing sufficient statistics $T^{(0)}$ and $T^{(1)}$. We do not compute the values $T(\mathbf{z}, \mathbf{x})$ but keep track of the sum as we impute values to the cells/latent variables. During iteration $2t$ of the evolution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

Algorithm 1 ESCA

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

Furthermore, in the case of discrete latent variable, updating sufficient statics only requires increments to the data structure $T^{(r)}$ allowing the use of approximate counters [21, 5]. Approximate counters greatly reduce memory costs for the counts: e.g., only 4 or 8 bits per counter. Recent empirical evidence demonstrates that



(a) Phase 1

(b) Phase 2

Figure 1: Efficient (re)use of buffers

approximate counters preserve statistical performance without compromising runtime performance [32]. In fact, speed often increases because not every increment to the counter results in a write to memory. Note, due to the compression, maintaining two buffers requires less memory than one. Finally, if the latent variables are discrete valued then we can leverage the fast Vose' alias method [34] to sample. The $O(|\mathcal{K}|)$ construction cost for the alias method can be easily amortized because the rule is homogeneous and thus alias tables can be shared. Details about alias sampling method is provided in Appendix E.

2.4 Wide applicability of ESCA

As stated previously, ESCA is technically applicable to any model in which the complete data likelihood is in the exponential family. Designing an ESCA algorithm for a model of interest requires simply deriving the S-step for the local update function in the automaton. The S-step is the full conditional (Equation 6) which is easy to derive for many models; for example, mixture models in which (1) the data and parameter components are conjugate and (2) the latent variables and priors are conjugate. We list a few examples of such models in Table 1 and provide additional details in Appendix F. Of course, the extent to which the models enable exploitation of sparsity varies.

ESCA is also applicable to models such as restricted Boltzmann machines (RBMs). For example, if the data were a collection of images, each cell could independently compute the S-step for its respective image. For RBMs the cell would flip a biased coin for each latent variable, and for deep Boltzmann machines, the cells could perform Gibbs sampling. We save a precise derivation and empirical evaluation for future work.

2.5 Understanding the limitations of ESCA

While ESCA has tremendous potential as a computational model for machine learning, in some cases, using it to obtain MAP estimates is not clear.

Consider an Ising model on an d-dimensional torus $\mathbb{H}$

$$p(x) \propto \prod_{\langle i,j\rangle\in\mathbb{H}} \exp(w_{ij}x_ix_j) \tag{9}$$

in which x_i takes on values in $\{-1,1\}$. The equilibrium distribution of SCA with a Gibbs update is then [24]

$$q(x) \propto \prod_{\langle i,j\rangle\in\mathbb{H}} \cosh(w_{ij}x_ix_j). \tag{10}$$

Note that the hyperbolic cosine function (cosh) is symmetric in the sense that $\cosh(r) = \cosh(-r)$. For values $r \geq 0$ cosh is a good approximation and has a maximum that corresponds to the exponential function; however, for values $r < 0$, the cosh is a poor approximation for the exponential function.

Let x_1, x_2 be two random variables taking on values in $\{-1,1\}$. We define a simple two-variable Ising model on a trivial one-dimensional torus:

$$p(x_1,x_2) \propto \exp(x_1x_2) \tag{11}$$

We can enumerate and quantify the state space under both SCA $q(x_1,x_2)$ and the true distribution $p(x_1,x_2)$:

state	x_1	x_2	$x_1 * x_2$	$q(x_1,x_2) \propto$	$p(x_1,x_2) \propto$
0	-1	-1	1	cosh(1)	exp(1)
1	-1	1	-1	cosh(-1)	exp(-1)
2	1	-1	-1	cosh(-1)	exp(-1)
3	1	1	1	cosh(1)	exp(1)

Since cosh is symmetric, all states are equally probable for SCA and states 1 and 2 are MAP states. Yet, under the true distribution, they are not. Consequently, SCA with a Gibbs rule for the local evolution function can yield incorrect MAP estimates.

Fortunately, in most cases we are interested in a model over a dataset in which the data is *i.i.d.* That is, we can fix our example as follows. Rather than parallelizing a single Ising model at the granularity of pixels (over a single torus or grid), we instead parallelize the Ising model at the granularity of the data (over multiple tori, one for each image). Then, we could employ Gibbs sampling on each image for the S-step.

2.6 Convergence

We now address the critical question of how the invariant measure of ESCA for the model presented in

Section 2.1 is related to the true MAP estimates. First, note that SCA is ergodic [18], a result that immediately applies if we ignore the deterministic components of our automata (corresponding to the observations). Now that we have established ergodicity, we next study the properties of the stationary distribution and find that the modes correspond to MAP estimates.

We make a few mild assumptions about the model:

- The observed data Fisher information is non-singular, i.e. $I(\eta) \succ 0$.
- For the Fisher information for $\mathbf{z}|\mathbf{x}$, we need it to be non-singular and central limit theorem, law of large number to hold, i.e. $\mathbb{E}_{\eta_0}[I_Z(\eta_0)] \succ 0$ and
$$\sup_{\eta} \left| \frac{1}{n} \sum_{i=1}^{n} I_{z_i}(\eta) - \mathbb{E}_{\eta_0}[I_X(\eta)] \right| \to 0 \text{ as } n \to \infty$$
- We assume that $\frac{1}{n}\sum_{i=1}^{n} \nabla_\eta \log p(x_i; \eta) = 0$ has at least one solution, let $\hat{\eta}$ be a solution.

These assumptions are reasonable. For example in case of mixture models (or topic models), it just means all component must be exhibited at least once and all components are unique. The details of this case are worked out in Appendix D. Also when the number of parameters grow with the data, e.g., for topic models, the second assumption still holds. In this case, we resort to corresponding result from high dimensional statistics by replacing the law of large numbers with Donsker's theorem and everything else falls into place.

Consequently, we show 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 by borrowing the results known for SEM [26]. In particular, we have:

Theorem 1 *Let the assumptions stated above hold and $\tilde{\eta}$, is the estimate from ESCA. Then as the number of* i.i.d. *data point goes to infinity, i.e. $n \to \infty$, we have*
$$\sqrt{n}(\tilde{\eta} - \hat{\eta}) \overset{\mathfrak{D}}{\to} \mathcal{N}\left(0, I(\eta_0)^{-1}[I - F(\eta_0)^{-1}\right) \qquad (12)$$
where $F(\eta_0) = I + \mathbb{E}_{\eta_0}[I_X(\eta_0)](I(\eta_0) + \mathbb{E}_{\eta_0}[I_X(\eta_0)])$.

This result implies that SEM flocks around a stationary point under very reasonable assumptions and tremendous computational benefits. Also, for such complicated models, reaching a stationary point is the best that most methods achieve anyway. Now we switch gears to adopt ESCA for LDA and perform some simple experimental evaluations.

3 ESCA for LDA

Topic modeling, and latent Dirichlet allocation (LDA) [2] in particular, have become a must-have of analytics platforms and consequently needs to scale to larger and larger datasets. In LDA, we model each document m of a corpus of M documents as a distribution θ_m that represents a mixture of topics. There are K such topics, and we model each topic k as a distribution ϕ_k over the vocabulary of words that appear in our corpus. Each document m contains N_m words w_{mn} from a vocabulary of size V, and we associate a latent variable z_{mn} to each of the words. The latent variables can take one of K values that indicate which topic the word belongs to. Both distributions θ_m and ϕ_k have a Dirichlet prior, parameterized respectively with a constant α and β. See Appendix B for more details.

3.1 Existing systems

Many of the scalable systems for topic modeling are based on one of two core inference methods: the collapsed Gibbs sampler (CGS) [12], and variational inference (VI) [2] and approximations thereof [1]. To scale LDA to large datasets, or for efficiency reasons, we may need to distribute and parallelize them. Both algorithms can be further approximated to meet such implementation requirements.

Collapsed Gibbs Sampling In collapsed Gibbs sampling the full conditional distribution of the latent topic indicators given all the others is
$$p(z_{mn} = k | \boldsymbol{z}^{\neg mn}, \boldsymbol{w}) \propto \boxed{(D_{mk} + \alpha)\frac{W_{kw_{mn}} + \beta}{T_k + \beta V}} \qquad (13)$$
where D_{mk} is the number of latent variables in document m that equal k, W_{kv} is the number of latent variables equal to k and whose corresponding word equals v, and T_k is the number of latent variables that equal k, all excluding current z_{mn}.

CGS is a sequential algorithm in which we draw latent variables in turn, and repeat the process for several iterations. The algorithm performs well statistically, and has further benefited from breakthroughs that lead to a reduction of the sampling complexity [38, 17]. This algorithm can be approximated to enable distribution and parallelism, primarily in two ways. One is to partition the data, perform one sampling pass and then assimilate the sampler states, thus yielding an approximate distributed version of CGS (AD-LDA) [25]. Another way is to partition the data and allow each sampler to communicate with a distributed central storage continuously. Here, each sampler sends the differential to the global state-keeper and receives from it the latest global value. A very scalable system built on this principle and leveraging inherent sparsity of LDA is YahooLDA [30]. Further improvement and sampling using alias table was incorporated in lightLDA [40]. Contemporaneously, a nomadic distribution scheme and sampling using Fenwick tree was proposed in F+LDA [39].

Variational Inference In variational inference (VI), we seek to optimize the parameters of an approximate distribution that assumes independence of the latent variables to find a member of the family that is close to the true posterior. Typically, for LDA, document-topic proportions and topic indicators are latent variables and topics are parameter. Then, coordinate ascent alternates between them.

One way to scale VI is stochastic variational inference (SVI) which employs SGD by repeatedly updating the topics via randomly chosen document subsets [14]. Adding a Gibbs step to SVI introduces sparsity for additional efficiency [20]. In some ways this is analogous to our S-step, but in the context of variational inference, the conditional is much more expensive to compute, requiring several rounds of sampling.

Another approach, CVB0, achieves scalability by approximating the collapsed posterior [31]. Here, they minimize the free energy of the approximate distribution for a given parameter γ_{mnk} and then use the zero-order Taylor expansion [1].

$$\gamma_{mnk} \propto \boxed{(D_{mk} + \alpha) \times \frac{W_{kw_{mn}} + \beta}{T_k + \beta\, V}} \tag{14}$$

where D_{mk} is the fractional contribution of latent variables in document m for topic k, W_{kv} is the contribution of latent variables for topic k and whose corresponding word equals v, and T_k is the the contribution of latent variables for topic k. Inference updates the variational parameters until convergence. It is possible to distribute and parallelize CVB0 over tokens [1]. VI and CVB0 are the core algorithms behind several scalable topic modeling systems including Mr.LDA [41] and the Apache Spark machine-learning suite.

Remark It is worth noticing that Gibbs sampling and variational inference, despite being justified very differently, have at their core the very same formulas (shown in a box in formula (13) and (14)). Each of which are literally deciding how important is some topic k to the word v appearing in document m by asking the questions: “How many times does topic k occur in document m?”, “How many times is word v associated with topic k?”, and “How prominent is topic k overall?”. It is reassuring that behind all the beautiful mathematics, something simple and intuitive is happening. As we see next, ESCA addresses the same questions via analogous formulas.

3.2 An ESCA Algorithm for LDA

To re-iterate, the point of using such a method for LDA is that the parallel update dynamics of the ESCA gives us an algorithm that is simple to parallelize, distribute and scale. In the next section, we will evaluate how it works in practice. For now, let us explain how we design our SCA to analyze data.

We begin by writing the stochastic EM steps for LDA (derivation is in Appendix B):

E-step: independently in parallel compute the conditional distribution locally:

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} \tag{15}$$

S-step: independently in parallel draw z_{ij} from the categorical distribution:

$$z_{mn} \sim \mathsf{Categorical}(q_{mn1}, ..., q_{mnK}) \tag{16}$$

M-step: independently in parallel compute the new parameter estimates:

$$\begin{aligned} \theta_{mk} &= \frac{D_{mk} + \alpha - 1}{N_m + K\alpha - K} \\ \phi_{kv} &= \frac{W_{kv} + \beta - 1}{T_k + V\beta - V} \end{aligned} \tag{17}$$

We simulate these inference steps in ESCA, which is a dynamical system with evolution function $\Phi : \mathcal{S} \longrightarrow \mathcal{S}$ over the state space $\mathcal{S}$. For LDA, the state space $\mathcal{S}$ is

$$\mathcal{S} = \mathcal{Z} \longrightarrow \mathcal{K} \times \mathcal{M} \times \mathcal{V} \tag{18}$$

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.

The initial state s_0 is the map defined as follows: for every occurrence of the word v in document m, we associate a cell z to the triple (k_z, m, v) where k_z is chosen uniformly at random from $\mathcal{K}$ and independently from $k_{z'}$ for all $z' \neq z$. This gives us

$$s_0 = z \mapsto (k_z, m, v) \tag{19}$$

We now need to describe the evolution function Φ. First, assuming that we have a state s and a cell z, we define the following distribution:

$$p_z(k|s) \propto \boxed{(D_{mk} + \alpha) \times \frac{W_{kv} + \beta}{T_k + \beta\, V}} \tag{20}$$

where $D_{mk} = \Big|\{\, z \mid \exists v.\ s(z) = (k, m, v)\,\}\Big|$, $W_{kv} = \Big|\{\, z \mid \exists m.\ s(z) = (k, m, v)\,\}\Big|$, and $T_k = \Big|\{\, z \mid \exists m.\ \exists v.\ s(z) = (k, m, v)\,\}\Big|$. Note that we have chosen our local update rule slightly different without an offset of -1 for the counts corresponding to the mode of the Dirichlet distributions and requiring $\alpha, \beta > 1$. Instead, our local update rule allows us to have the relaxed requirement $\alpha, \beta > 0$ which is more common for LDA inference algorithms.

Assuming that $s(z) = (k, m, v)$ and that k' is a sample from p_z (hence the name "stochastic" cellular automaton) we define the local update function as:

$$\begin{aligned} \phi(s, z) &= (k', m, v) \\ \text{where} \quad s(z) = (k, m, v) \quad &\text{and} \quad k' \sim p_z(\cdot\,|s) \end{aligned} \tag{21}$$

That is, the document and word of the cell remain unchanged, but we choose a new topic 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.

$$\Phi(s) = z \mapsto \phi(s, z) \tag{22}$$

Finally, the SCA algorithm simulates the evolution function Φ starting with s_0. Of course, since LDA's complete data likelihood is in the exponential family, we never have to represent the states explicitly, and instead employ the sufficient statistics.

Our implementation has two copies of the count matrices D^i, W^i, and T^i for $i = 0$ or 1 (as in CGS or CVB0, we do not compute the values D_{ik}, W_{kv}, and T_k but keep track of the counts as we assign topics to the cells/latent variables). During iteration i of the evolution function, we apply Φ by reading $D^{i \bmod 2}$, $W^{i \bmod 2}$, and $T^{i \bmod 2}$ and incrementing $D^{i+1 \bmod 2}$, $W^{i+1 \bmod 2}$, and $T^{i+1 \bmod 2)}$ as we assign topics.

3.3 Advantages of ESCA for LDA

The positive consequences of ESCA as a choice for inference on LDA are many:

- Our memory footprint is minimal since we only store the data and sufficient statistics. In contrast to MCMC methods, we do not store the assignments to latent variables $\mathbf{z}$. In contrast to variational methods, we do not store the variational parameters γ. Further, variational methods require K memory accesses (one for each topic) per word. In contrast, the S-step ensures we only have a single access (for the sampled topic) per word. Such reduced pressure on the memory bandwidth can improve performance significantly for highly parallel applications.
- We can further reduce the memory footprint by compressing the sufficient statistics with approximate counters [21, 5]. This is possible because updating the sufficient statistics only requires increments as in Mean-for-Mode [32]. In contrast, CGS decrements counts, preventing the use of approximate counters.
- Our implementation is lock-free (in that it does not use locks, but assumes atomic increments) because the double buffering ensures we never read or write to the same data structures. There is less synchronization, which at scale is significant.
- Finally, our algorithm is able to fully benefit from Vose's alias method [35] because homogeneous update rule for SCA ensures that the cost for constructing the alias table is amortized across the cells. To elaborate, the SCA update Equation (20) decomposes as

$$p_z(k|s) \propto \left[D_{mk} \frac{W_{kv} + \beta}{T_k + \beta V}\right] + \left[\alpha \frac{W_{kv} + \beta}{T_k + \beta V}\right] \tag{23}$$

allowing us to treat it as a discrete mixture and divide the sampling procedure into a two steps. First, we toss a biased coin to decide which term of the equation to sample, and second, we employ a specialized sampler depending on the chosen term. The first term is extremely sparse (documents comprise only a small handful of topics) and a basic sampling procedure suffices. The second term is not sparse, but is independent of the current document m and depends only on the W and T matrices. Moreover, as mentioned earlier, during iteration i, we will be only reading values from non-changing $W^{i \bmod 2}$, and $T^{i \bmod 2}$ matrices. As a result, at the start of each iteration we can precompute, from the W and T matrices, tables for use with Vose's alias method, which enables sampling from the second term in a mere 3 CPU operations. Thus, the evolution for ESCA is extremely efficient.

3.3.1 Connection to SGD

We can view ESCA as implicit SGD on MAP for LDA. This connection alludes to the convergence rate of ESCA. To illustrate, we consider θ only. As pointed out in [37, 29], one EM step is:

$$\theta_m^+ = \theta_m + M \frac{\partial \log p}{\partial \theta_{mk}}$$

which is gradient descent with Frank-Wolfe type update and line search. Similarly, for ESCA using stochastic EM, one step is

$$\theta_{mk}^+ = \frac{Dn_{mk}}{N_m} = \frac{1}{N_m} \sum_{n=1}^{N_m} \delta(z_{mn} = k)$$

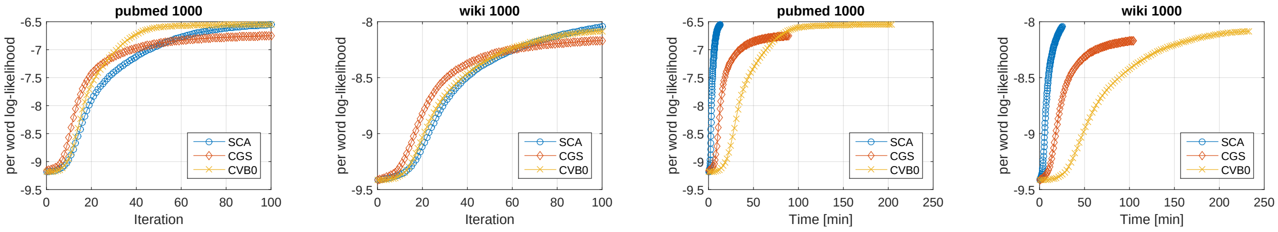
Again vectorizing and re-writing as earlier:

$$\theta_m^+ = \theta_m + Mg$$

where $M = \frac{1}{N_m}\left[\operatorname{diag}(\theta_m) - \theta_m \theta_m^T\right]$ and $g = \frac{1}{\theta_{mk}} \sum_{n=1}^{N_m} \delta(z_{mn} = k)$. The vector g can be shown to be an unbiased noisy estimate of the gradient, i.e.

$$\mathbb{E}[g] = \frac{1}{\theta_{mk}} \sum_{n=1}^{N_i} \mathbb{E}[\delta(z_{ij} = k)] = \frac{\partial \log p}{\partial \theta_{mk}}$$

Thus, a single step of SEM on our SCA is equivalent to a single step of SGD. Consequently, we could further embrace the connection to SGD and use a subset of the



(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.

data for the S and M steps, similar to incremental EM [23]. Note that in the limit in which batches comprise just a single token, the algorithm emulates a collapsed Gibbs sampler. This interpretation strengthens the theoretical justification for many existing approximate Gibbs sampling approaches.

4 Experiments

To evaluate the strength and weaknesses of our algorithm, we compare against parallel and distributed implementations of CGS and CVB0. We also compare our results to performance numbers reported in the literature including those of F+LDA and lightLDA.

Software & hardware All three algorithms are implemented in simple C++11. We implement multi-threaded 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 also implemented a version of ESCA with a sparse representation for the array D of counts of 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. For random number generation we employ Intel©Digital Random Number Generators through instruction RDRAND, which uses thermal noise within the silicon to output a random stream of bits at 3 Gbit/s, producing true random numbers.

Datasets We experiment on two public datasets, both of which are cleaned by removing stop words and rare words: PubMed abstracts and English Wikipedia. We also run on a third proprietary dataset.

Dataset	V	M	Tokens
PubMed	141,043	8,200,000	737,869,085
Wikipedia	210,233	6,631,176	1,133,050,514
Large	∼140,000	∼3 billion	∼171 billion

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 [13]; other systems such as YahooLDA and Mallet also use this as the default parameter setting. The results are presented in Figure 2. and some more experiments in Appendix G.

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) [40].

5 Discussion

We have described a novel inference method for latent variable models that simulates a stochastic cellular automaton. The equilibrium of the dynamics are MAP fixed points and the algorithm has many desirable computational properties: it is embarrassingly parallel, memory efficient, and like HOGWILD!, is virtually lock-free. Further, for many models, it enables the use of approximate counters and the alias method. Thus, we were able to achieve an order of magnitude speed-up over the current state-of-the-art inference algorithms for LDA with accuracy comparable to collapsed Gibbs sampling.

In general, we cannot always guarantee the correct invariant measure [7], and found that parallelizing improperly causes convergence to incorrect MAP fixed points. Even so, SCA is used for simulating Ising models in statistical physics [33]. Interestingly, in previous work [16], it has been shown that stochastic cellular automata are closely related to equilibrium statistical models and the stationary distribution is known for a large class of finite stochastic cellular automata.

References

[1] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proc. Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 27–34, Arlington, Virginia, USA, 2009. AUAI Press.

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

[3] 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.

[4] 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.

[5] Miklós Csűrös. Approximate counting with a floating-point counter. In M. T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics (COCOON 2010)*, number 6196 in Lecture Notes in Computer Science, pages 358–367. Springer Berlin Heidelberg, 2010. See also http://arxiv.org/pdf/0904.3062.pdf.

[6] 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.

[7] Donald A. Dawson. Synchronous and asynchronous reversible Markov systems. *Canadian mathematical bulletin*, 17:633–649, 1974.

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

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

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

[11] 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.

[12] 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.

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

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

[15] 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.

[16] Joel L. Lebowitz, Christian Maes, and Eugene R. Speer. Statistical mechanics of probabilistic cellular automata. *Journal of statistical physics*, 59:117–170, April 1990.

[17] Aaron Q. Li, Amr Ahmed, Sujith Ravi, and Alexander J. Smola. Reducing the sampling complexity of topic models. In *20th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, 2014.

[18] 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.

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

[20] David Mimno, Matt Hoffman, and David Blei. Sparse stochastic inference for latent dirichlet allocation. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1599–1606, New York, NY, USA, July 2012. Omnipress.

[21] Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, October 1978.

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

[23] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[24] A. U. Neumann and B. Derrida. Finite size scaling study of dynamical phase transitions in two dimensional models: Ferromagnet, symmetric and non symmetric spin glasses. *J. Phys. France*, 49:1647–1656, 08 1988.

[25] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *J. Machine Learning Research*, 10:1801–1828, December 2009. http://dl.acm.org/citation.cfm?id=1577069.1755845.

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

[27] 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.

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

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

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

[31] 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.

[32] Jean-Baptiste Tristan, Joseph Tassarotti, and Guy L. Steele Jr. Efficient training of LDA on a GPU by Mean-For-Mode Gibbs sampling. In *32nd International Conference on Machine Learning*, volume 37 of *ICML 2015*, 2015. Volume 37 of the Journal in Machine Learning Research: Workshop and Conference Proceedings.

[33] Gérard Y. Vichniac. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):96–116, January 1984.

[34] Michael D. Vose. A linear algorithm for generating random numbers with a given distribution. *Software Engineering, IEEE Transactions on*, 1991.

[35] Michael D Vose. A linear algorithm for generating random numbers with a given distribution. *Software Engineering, IEEE Transactions on*, 17(9):972–975, 1991.

[36] 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.

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

[38] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proc. 15th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, KDD '09, pages 937–946, New York, 2009. ACM.

[39] 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.

[40] 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.

[41] Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad L Alkhouja. Mr. lda: A flexible large scale topic modeling package using variational inference in mapreduce. In *Proceedings of the 21st international conference on World Wide Web*, pages 879–888. ACM, 2012.

A (Stochastic) EM in General

Expectation-Maximization (EM) is an iterative method for finding the maximum likelihood or *maximum a posteriori* (MAP) estimates of the parameters in statistical models when data is only partially, or when model depends on unobserved latent variables. This section is inspired from `http://www.ece.iastate.edu/~namrata/EE527_Spring08/emlecture.pdf`

We derive EM algorithm for a very general class of model. Let us define all the quantities of interest.

Table 2: Notation

Symbol	Meaning
$\mathbf{x}$	Observed data
$\mathbf{z}$	Unobserved data
$(\mathbf{x}, \mathbf{z})$	Complete data
$f_{\mathbf{X};\eta}(\mathbf{x};\eta)$	marginal observed data density
$f_{\mathbf{Z};\eta}(\mathbf{z};\eta)$	marginal unobserved data density
$f_{\mathbf{X},\mathbf{Z};\eta}(\mathbf{x},\mathbf{z};\eta)$	complete data density/likelihood
$f_{\mathbf{Z}\mid\mathbf{X};\eta}(\mathbf{z}\mid\mathbf{x};\eta)$	conditional unobserved-data (missing-data) density.

Objective: To maximize the marginal log-likelihood or posterior, i.e.

$$L(\eta) = \log f_{\mathbf{X};\eta}(\mathbf{x};\eta). \tag{24}$$

Assumptions:

1. z_i are independent given η. So

$$f_{\mathbf{Z};\eta}(\mathbf{z};\eta) = \prod_{i=1}^{N} f_{Z_i;\eta}(z_i;\eta), \tag{25}$$

2. x_i are independent given missing data z_i and η. So

$$f_{\mathbf{X},\mathbf{Z};\eta}(\mathbf{x},\mathbf{z};\eta) = \prod_{i=1}^{N} f_{X_i,Z_i;\eta}(x_i,z_i;\eta). \tag{26}$$

As a consequence we obtain:

$$f_{\mathbf{Z}|\mathbf{X};\eta}(\mathbf{z}|\mathbf{x};\eta) = \prod_{i=1}^{N} f_{Z_i|X_i;\eta}(z_i|x_i;\eta), \tag{27}$$

Now,

$$L(\eta) = \log f_{\mathbf{X};\eta}(\mathbf{x};\eta) = \log f_{\mathbf{X},\mathbf{Z};\eta}(\mathbf{x},\mathbf{z};\eta) - \log f_{\mathbf{Z}|\mathbf{X};\eta}(\mathbf{z}|\mathbf{x};\eta) \tag{28}$$

or, summing across observations,

$$L(\eta) = \sum_{i=1}^{N} \log f_{X_i;\eta}(x_i;\eta) = \sum_{i=1}^{N} \log f_{X_i,Z_i;\eta}(x_i,z_i;\eta) - \sum_{i=1}^{N} \log f_{Z_i|X_i;\eta}(z_i|x_i;\eta). \tag{29}$$

Let us take the expectation of the above expression with respect to $f_{Z_i|X_i;\eta}(z_i|x_i;\eta_p)$, where we choose $\eta = \eta_p$:

$$\begin{aligned} &\sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta}\left[\log f_{X_i;\eta}(x_i;\eta)|x_i;\eta_p\right] \\ &\quad = \sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta}\left[\log f_{X_i,Z_i;\eta}(x_i,z_i;\eta)|x_i;\eta_p\right] - \sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta}\left[\log f_{Z_i|X_i;\eta}(z_i|x_i;\eta)|x_i;\eta_p\right] \end{aligned} \tag{30}$$

Since $L(\eta) = \log f_{\mathbf{X};\eta}(\mathbf{x};\eta)$ does not depend on $\mathbf{z}$, it is invariant for this expectation. So we recover:

$$\begin{aligned} L(\eta) &= \sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta} \left[\log f_{X_i,Z_i;\eta}(x_i, z_i; \eta)|x_i; \eta_p\right] - \sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta} \left[\log f_{Z_i|X_i;\eta}(z_i|x_i; \eta)|x_i; \eta_p\right] \\ &= Q(\eta|\eta_p) - H(\eta|\eta_p). \end{aligned} \tag{31}$$

Now, (31) may be written as

$$Q(\eta|\eta_p) = L(\eta) + \underbrace{H(\eta|\eta_p)}_{\leq H(\eta_p|\eta_p)} \tag{32}$$

Here, observe that $H(\eta|\eta_p)$ is maximized (with respect to η) by $\eta = \eta_p$, i.e.

$$H(\eta|\eta_p) \leq H(\eta_p|\eta_p) \tag{33}$$

Simple proof using Jensen's inequality.

As our objective is to maximize $L(\eta)$ with respect to η, if we maximize $Q(\eta|\eta_p)$ with respect to η, it will force $L(\eta)$ to increase. This is what is done repetitively in EM. To summarize, we have:

E-step : Compute $f_{Z_i|X_i;\eta}(z_i|x_i; \eta_p)$ using current estimate of $\eta = \eta_p$.

M-step : Maximize $Q(\eta|\eta_p)$ to obtain next estimate η_{p+1}.

Now assume that the complete data likelihood belongs to the exponential family, i.e.

$$f_{X_i,Z_i;\eta}(x_i, z_i; \eta) = \exp\left(\langle T\left(z_i, x_i\right), \eta\rangle - g(\eta)\right) \tag{34}$$

then

$$\begin{aligned} Q(\eta|\eta_p) &= \sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta} \left[\log f_{X_i,Z_i;\eta}(x_i, z_i; \eta)|x_i; \eta_p\right] \\ &= \sum_{i=1}^{N} \mathbb{E}_{Z_i|X_i;\eta} \left[\langle T\left(z_i, x_i\right), \eta\rangle - g(\eta)|x_i; \eta_p\right] \end{aligned} \tag{35}$$

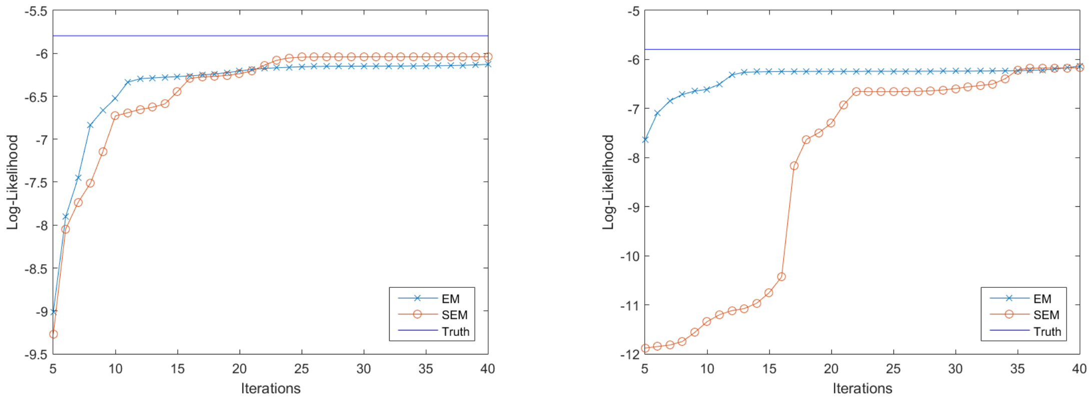
To find the maximizer, differentiate and set it to zero:

$$\frac{1}{N} \sum_{i} \mathbb{E}_{Z_i|X_i;\eta} \left[\langle T\left(z_i, x_i\right), \eta\rangle |x_i; \eta_p\right] = \frac{dg(\eta)}{d\eta} \tag{36}$$

and one can obtain the maximizer by solving this equation.

Stochastic EM (SEM) introduces an additional simulation after the E-step that replaces the full distribution with a single sample:

S-step Sample $z_i \sim f_{Z_i|X_i;\eta}(z_i|x_i; \eta_p)$



(a) Same initialization

(b) Bad initialization for SEM

Figure 3: Performance of SEM

This essentially means we replace $\mathbb{E}[\cdot]$ with an empirical estimate. Thus, instead of solving (36), we simply have:

$$\frac{1}{N}\sum_i T(z_i, x_i) = \frac{dg(\eta)}{d\eta}. \tag{37}$$

Computing and solving this system of equations is considerably easier than (36).

Now to demonstrate that SEM is well behaved and works in practice, we run a small experiment. Consider the problem of estimating the parameters of a Gaussian mixture. We choose a 2-dimensional Gaussian with $K = 30$ clusters and 100,000 training points and 1,000 test points. We run EM and SEM with the following initialization:

- Both SEM and EM are provided the same initialization.
- SEM is deliberately provided a bad initialization, while EM is not.

The log-likelihood on the heldout test set is shown in Figure 3.

B (S)EM Derivation for LDA

We derive an EM procedure for LDA.

B.1 LDA Model

In LDA, we model each document m of a corpus of M documents as a distribution θ_m that represents a mixture of topics. There are K such topics, and we model each topic k as a distribution ϕ_k over the vocabulary of words that appear in our corpus. Each document m contains N_m words w_{mn} from a vocabulary of size V, and we associate a latent variable z_{mn} to each of the words. The latent variables can take one of K values that indicate which topic the word belongs to. We give each of the distributions θ_m and ϕ_k a Dirichlet prior, parameterized respectively with a constant α and β. More concisely, LDA has the following mixed density.

$$p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \left[\prod_{m=1}^{M}\prod_{n=1}^{N_m} \mathrm{Cat}(w_{mn} \mid \phi_{z_{mn}})\ \mathrm{Cat}(z_{mn} \mid \theta_m)\right]\left[\prod_{m=1}^{M} \mathrm{Dir}(\theta_m \mid \alpha)\right]\left[\prod_{k=1}^{K} \mathrm{Dir}(\phi_k \mid \beta)\right] \tag{38}$$

The choice of a Dirichlet prior is not a coincidence: we can integrate all of the variables θ_m and ϕ_k and obtain the following closed form solution.

$$p(\boldsymbol{w}, \boldsymbol{z}) = \left[\prod_{m=1}^{M} \mathrm{Pol}\big(\{z_{m'n} \mid m' = m\}, K, \alpha\big)\right]\left[\prod_{k=1}^{K} \mathrm{Pol}\big(\{w_{mn} \mid z_{mn} = k\}, V, \beta\big)\right] \tag{39}$$

where Pol is the Polya distribution

$$\mathrm{Pol}(S, X, \eta) = \frac{\Gamma(\eta K)}{\Gamma(|S| + \eta X)} \prod_{x=1}^{X} \frac{\Gamma\big(\big|\{z \mid z \in S, z = x\}\big| + \eta\big)}{\Gamma(\eta)} \tag{40}$$

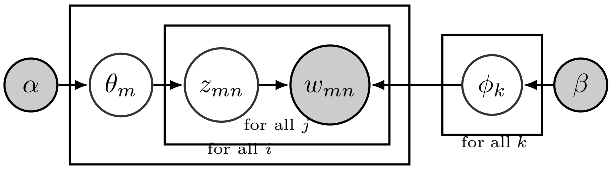


Figure 4: LDA Graphical Model

Algorithm 2 LDA Generative Model

input: $\boldsymbol{\alpha}, \boldsymbol{\beta}$

```
for $k = 1 \to K$ do
    Choose topic $\boldsymbol{\phi}_k \sim \mathsf{Dir}(\boldsymbol{\beta})$
end for
for all document $m$ in corpus $D$ do
    Choose a topic distribution $\boldsymbol{\theta}_m \sim \mathsf{Dir}(\boldsymbol{\alpha})$
    for all word index $n$ from 1 to $N_m$ do
        Choose a topic $z_{mn} \sim \mathsf{Categorical}(\boldsymbol{\theta}_m)$
        Choose word $w_{mn} \sim \mathsf{Categorical}(\boldsymbol{\phi}_{z_{mn}})$
    end for
end for
```

The joint probability density can be expressed as:

$$\begin{aligned} p(W, Z, \theta, \phi|\alpha, \beta) &= \left[\prod_{k=1}^{K} p(\phi_k|\beta)\right]\left[\prod_{m=1}^{M} p(\theta_m|\alpha) \prod_{n=1}^{N_m} p(z_{mn}|\theta_m) p(w_{mn}|\phi_{z_{mn}})\right] \\ &\propto \left[\prod_{k=1}^{K}\prod_{v=1}^{V} \phi_{kv}^{\beta-1}\right]\left[\prod_{m=1}^{M}\left(\prod_{k=1}^{K} \theta_{mk}^{\alpha-1}\right)\prod_{n=1}^{N_m} \theta_{m z_{mn}} \phi_{z_{mn} w_{mn}}\right] \end{aligned} \tag{41}$$

B.2 Expectation Maximization

We begin by marginalizing the latent variable Z and finding the lower bound for the likelihood/posterior:

$$\begin{aligned}
\log p(W,\theta,\phi|\alpha,\beta) &= \log \sum_Z p(W,Z,\theta,\phi|\alpha,\beta) \\
&= \sum_{m=1}^{M}\sum_{n=1}^{N_m} \log \sum_{k=1}^{K} p(z_{mn}=k|\theta_m)p(w_{mn}|\phi_k) \\
&\qquad + \sum_{k=1}^{K}\log p(\phi_k|\beta) + \sum_{m=1}^{M}\log p(\theta_m|\alpha) \\
&= \sum_{m=1}^{M}\sum_{n=1}^{N_m} \log \sum_{k=1}^{K} q(z_{mn}=k|w_{mn}) \frac{p(z_{mn}=k|\theta_m)p(w_{mn}|\phi_k)}{q(z_{mn}=k|w_{mn})} \\
&\qquad + \sum_{k=1}^{K}\log p(\phi_k|\beta) + \sum_{m=1}^{M}\log p(\theta_m|\alpha) \\
\text{(Jensen Inequality)} \quad &\geq \sum_{m=1}^{M}\sum_{n=1}^{N_m}\sum_{k=1}^{K} q(z_{mn}=k|w_{mn}) \log \frac{p(z_{mn}=k|\theta_m)p(w_{mn}|\phi_k)}{q(z_{mn}=k|w_{mn})} \\
&\qquad + \sum_{k=1}^{K}\log p(\phi_k|\beta) + \sum_{m=1}^{M}\log p(\theta_m|\alpha)
\end{aligned} \tag{42}$$

Let us define the following functional:

$$\begin{aligned}
F(q,\theta,\phi) := &- \sum_{m=1}^{M}\sum_{n=1}^{N_m} D_{KL}(q(z_{mn}|w_{mn})||p(z_{mn}|w_{mn},\theta_m,\phi)) \\
&+ \sum_{m=1}^{M}\sum_{n=1}^{N_m} p(w_{mn}|\theta_m,\phi) + \sum_{k=1}^{K}\log p(\phi_k|\beta) + \sum_{m=1}^{M}\log p(\theta_m|\alpha)
\end{aligned} \tag{43}$$

B.2.1 E-Step

In the E-step, we fix θ, ϕ and maximize F for q. As q appears only in the KL-divergence term, it is equivalent to minimizing the KL-divergence between $q(z_{mn}|w_{mn})$ and $p(z_{mn}|w_{mn},\theta_m,\phi)$. We know that for any distributions f and g the KL-divergence is minimized when $f = g$ and is equal to 0. Thus, we have

$$\begin{aligned}
q(z_{mn}=k|w_{mn}) &= p(z_{mn}=k|w_{mn},\theta_m,\phi) \\
&= \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}}
\end{aligned} \tag{44}$$

For simplicity of notation, let us define

$$\boxed{q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}}} \tag{45}$$

B.2.2 M-Step

In the E-step, we fix q and maximize F for θ, ϕ. As this will be a constrained optimization (θ and ϕ must lie on simplex), we use standard constrained optimization procedure of Lagrange multipliers. The Lagrangian can be expressed as:

$$
\begin{aligned}
\mathcal{L}(\theta, \phi, \lambda, \mu) &= \sum_{m=1}^{M} \sum_{m=1}^{N_m} \sum_{k=1}^{K} q(z_{mn} = k | w_{mn}) \log \frac{p(z_{mn} = k | \theta_m) p(w_{mn} | \phi_k)}{q(z_{mn} = k | w_{mn})} + \sum_{k=1}^{K} \log p(\phi_k | \beta) \\
&\quad + \sum_{m=1}^{M} \log p(\theta_m | \alpha) + \sum_{k=1}^{K} \lambda_k \left(1 - \sum_{v=1}^{V} \phi_{kv}\right) + \sum_{m=1}^{M} \mu_i \left(1 - \sum_{k=1}^{K} \theta_{mk}\right) \\
&= \sum_{m=1}^{M} \sum_{n=1}^{N_m} \sum_{k=1}^{K} q_{mnk} \log \theta_{mk} \phi_{k w_{mn}} + \sum_{k=1}^{K} \sum_{v=1}^{V} (\beta_v - 1) \log \phi_{kv} + \sum_{m=1}^{M} \sum_{k=1}^{K} (\alpha_k - 1) \log \theta_{mk} \\
&\quad + \sum_{k=1}^{K} \lambda_k \left(1 - \sum_{v=1}^{V} \phi_{kv}\right) + \sum_{m=1}^{M} \mu_m \left(1 - \sum_{k=1}^{K} \theta_{mk}\right) + \text{const.}
\end{aligned}
\tag{46}
$$

Maximising θ Taking derivative with respect to θ_{mk} and setting it to 0, we obtain

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta_{mk}} &= 0 = \sum_{j=1}^{N_m} \frac{q_{mnk} + \alpha_k - 1}{\theta_{mk}} - \mu_m \\
\mu_m \theta_{mk} &= \sum_{j=1}^{N_i} q_{mnk} + \alpha_k - 1
\end{aligned}
\tag{47}
$$

After solving for μ_m, we finally obtain

$$
\theta_{mk} = \frac{\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1}{\sum_{k'=1}^{K} \sum_{j=1}^{N_m} q_{mnk'} + \alpha_{k'} - 1}
\tag{48}
$$

Note that $\sum_{k'=1}^{K} q_{mnk'} = 1$, we reach at the optimizer:

$$
\boxed{\theta_{mk} = \frac{1}{N_m + \sum(\alpha_{k'} - 1)} \left(\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1\right)}
\tag{49}
$$

Maximising ϕ Taking derivative with respect to ϕ_{kv} and setting it to 0, we obtain

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \phi_{kv}} &= 0 = \sum_{m=1}^{M} \sum_{n=1}^{N_m} \frac{q_{mnk} \delta(v - w_{mn}) + \beta_v - 1}{\phi_{kv}} - \lambda_k \\
\lambda_k \phi_{kv} &= \sum_{m=1}^{M} \sum_{n=1}^{N_m} q_{mnk} \delta(v - w_{mn}) + \beta_v - 1
\end{aligned}
\tag{50}
$$

After solving for λ_k, we finally obtain

$$
\phi_{kv} = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N_m} q_{mnk} \delta(v - w_{mn}) + \beta_v - 1}{\sum_{v'=1}^{V} \sum_{m=1}^{M} \sum_{n=1}^{N_m} \delta(v' - w_{mn}) + \beta_{v'} - 1}
\tag{51}
$$

Note that $\sum_{v'=1}^{V} \delta(v' - w_{mn}) = 1$, we reach at the optimizer:

$$
\boxed{\phi_{kv} = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N_m} q_{mnk} \delta(v - w_{mn}) + \beta_v - 1}{\sum_{m=1}^{M} \sum_{n=1}^{N_m} q_{mnk} + \sum(\beta_{v'} - 1)}}
\tag{52}
$$

B.3 Introducing Stochasticity

After performing the E-step, we add an extra simulation step, i.e. we draw and impute the values for the latent variables from its distribution conditioned on data and current estimate of the parameters. This means basically q_mnk gets transformed into $\delta(z_{mn} - \tilde{k})$ where $\tilde{k}$ is value drawn from the conditional distribution. Then we proceed to perform the M-step, which is even simpler now. To summarize SEM for LDA will have following steps:

E-step : in parallel compute the conditional distribution locally:

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{ij}}} \tag{53}$$

S-step : in parallel draw z_{mn} from the categorical distribution:

$$z_{mn} \sim \mathsf{Categorical}(q_{mn1}, ..., q_{mnK}) \tag{54}$$

M-step : in parallel compute the new parameter estimates:

$$\begin{aligned} \theta_{mk} &= \frac{D_{mk} + \alpha_k - 1}{N_m + \sum(\alpha_{k'} - 1)} \\ \phi_{kv} &= \frac{W_{kv} + \beta_v - 1}{T_k + \sum(\beta_{v'} - 1)} \end{aligned} \tag{55}$$

where $D_{mk} = \Big|\big\{\, z_{mn} \mid z_{mn} = k \,\big\}\Big|$,
$W_{kv} = \Big|\big\{\, z_{mn} \mid w_{mn} = v,\, z_{mn} = k \,\big\}\Big|$, and
$T_k = \Big|\big\{\, z_{mn} \mid z_{mn} = k \,\big\}\Big| = \sum_{v=1}^{V} W_{kv}$.

C Equivalency between (S)EM and (S)GD for LDA

We study the equivalency between (S)EM and (S)GD for LDA.

C.1 EM for LDA

EM for LDA can be summarized by follows:

E-Step

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} \tag{56}$$

M-Step

$$\begin{aligned} \theta_{mk} &= \frac{1}{N_m + \sum(\alpha_{k'}-1)}\left(\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1\right) \\ \phi_{kv} &= \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v-w_{mn}) + \beta_v - 1}{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk} + \sum(\beta_{v'}-1)} \end{aligned} \tag{57}$$

C.2 GD for LDA

The joint probability density can be expressed as:

$$\begin{aligned} p(W,Z,\theta,\phi|\alpha,\beta) &= \left[\prod_{k=1}^{K} p(\phi_k|\beta)\right]\left[\prod_{m=1}^{M} p(\theta_m|\alpha)\prod_{n=1}^{N_m} p(z_{mn}|\theta_m)p(w_{mn}|\phi_{z_{mn}})\right] \\ &\propto \left[\prod_{k=1}^{K}\prod_{v=1}^{V}\phi_{kv}^{\beta-1}\right]\left[\prod_{m=1}^{M}\left(\prod_{k=1}^{K}\theta_{mk}^{\alpha-1}\right)\prod_{n=1}^{N_m}\theta_{mz_{mn}}\phi_{z_{mn}w_{mn}}\right] \end{aligned} \tag{58}$$

The log-probability of joint model with Z marginalized can be written as:

$$\begin{aligned} \log p(W,\theta,\phi|\alpha,\beta) &= \log\sum_{Z} p(W,Z,\theta,\phi|\alpha,\beta) \\ &= \sum_{m=1}^{M}\sum_{n=1}^{N_m}\log\sum_{k=1}^{K} p(z_{mn}=k|\theta_m)p(w_{mn}|\phi_k) \\ &\qquad + \sum_{k=1}^{K}\log p(\phi_k|\beta) + \sum_{m=1}^{M}\log p(\theta_m|\alpha) \\ &= \sum_{m=1}^{M}\sum_{n=1}^{N_m}\log\sum_{k=1}^{K}\theta_{mk}\phi_{kw_{mn}} \\ &\qquad + \sum_{m=1}^{M}\sum_{k=1}^{K}(\alpha_k-1)\log\theta_{mk} + \sum_{k=1}^{K}\sum_{v=1}^{V}(\beta_v-1)\log\phi_{kv} \end{aligned} \tag{59}$$

Gradient for topic per document Now take derivative with respect to θ_{mk}:

$$\begin{aligned} \frac{\partial\log p}{\partial\theta_{mk}} &= \sum_{j=1}^{N_m}\frac{\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} + \frac{\alpha_k-1}{\theta_{mk}} \\ &= \frac{1}{\theta_{mk}}\left(\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1\right) \end{aligned} \tag{60}$$

Gradient for word per topic Now take derivative with respect to ϕ_{kv}:

$$\begin{aligned}\frac{\partial \log p}{\partial \phi_{kv}} &= \sum_{m=1}^{M}\sum_{n=1}^{N_m}\frac{\theta_{mk}\delta(v-w_{mn})}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}}+\frac{\beta_v-1}{\phi_{kv}} \\ &= \frac{1}{\phi_{kv}}\left(\sum_{m=1}^{M}\sum_{n=1}^{N_m}q_{mnk}\delta(v-w_{mn})+\beta_v-1\right)\end{aligned} \tag{61}$$

C.3 Equivalency

If we look at one step of EM:

For topic per document

$$\begin{aligned}\theta_{mk}^{+} &= \frac{1}{N_m+\sum(\alpha_{k'}-1)}\left(\sum_{n=1}^{N_m}q_{mnk}+\alpha_k-1\right) \\ &= \frac{\theta_{mk}}{N_m+\sum(\alpha_{k'}-1)}\frac{\partial \log p}{\partial \theta_{mk}}\end{aligned}$$

Vectorize and can be re-written as:

$$\theta_m^{+} = \theta_m + \frac{1}{N_m+\sum(\alpha_{k'}-1)}\left[\text{diag}(\theta_m)-\theta_m\theta_m^T\right]\frac{\partial \log p}{\partial \theta_m} \tag{62}$$

For word per topic

$$\begin{aligned}\phi_{kv}^{+} &= \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m}q_{mnk}\delta(v-w_{mn})+\beta_v-1}{\sum_{m=1}^{M}\sum_{n=1}^{N_m}q_{mnk}+\sum(\beta_{v'}-1)} \\ &= \frac{\phi_{kv}}{\sum_{m=1}^{M}\sum_{n=1}^{N_m}q_{mnk}+\sum(\beta_{v'}-1)}\frac{\partial \log p}{\partial \phi_{kv}}\end{aligned}$$

Vectorize and can be re-written as:

$$\theta_m^{+} = \theta_m + \frac{1}{N_m+\sum(\alpha_{k'}-1)}\left[\text{diag}(\theta_m)-\theta_m\theta_m^T\right]\frac{\partial \log p}{\partial \theta_m} \tag{63}$$

C.4 SEM for LDA

We summarize our SEM derivation for LDA as follows:

E-Step

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} \tag{64}$$

S-step

$$z_{mn} \sim \mathsf{Categorical}(q_{mn1},...,q_{mnK}) \tag{65}$$

M-step

$$\begin{aligned}\theta_{mk} &= \frac{D_{mk}+\alpha_k-1}{N_m+\sum(\alpha_{k'}-1)} \\ \phi_{kv} &= \frac{W_{kv}+\beta_v-1}{T_k+\sum(\beta_{v'}-1)}\end{aligned} \tag{66}$$

C.5 Equivalency

In case of LDA, let us consider only θ for the purpose of illustration. Now consider the case of stochastic EM, the update over one step is:

$$\theta_{ik}^{+} = \frac{n_{ik}}{N_i} = \frac{1}{N_i}\sum_{j=1}^{N_i}\delta(z_{ij}=k)$$

Again vectorizing and re-writing as earlier:

$$\theta_i^+ = \theta_i + Mg$$

where $M = \frac{1}{N_i}\left[\mathrm{diag}(\theta_i) - \theta_i\theta_i^T\right]$ and $g = \frac{1}{\theta_{ik}}\sum_{j=1}^{N_i}\delta(z_{ij} = k)$. The vector g can be shown to be an unbiased noisy estimate of the gradient, i.e.

$$\begin{aligned}\mathbb{E}[g] &= \frac{1}{\theta_{ik}}\sum_{j=1}^{N_i}\mathbb{E}[\delta(z_{ij} = k)] \\ &= \frac{1}{\theta_{ik}}\sum_{j=1}^{N_i} q_{ijk} \qquad\qquad = \frac{\partial \log p}{\partial \theta_{ik}}\end{aligned}$$

Thus, it is SGD with constraints. However, note that stochasticity does not arise from sub-sampling data as usually in SGD, rather from the randomness introduced in the S-step.

D Non-singularity of Fisher Information for Mixture Models

Let us consider a general mixture model:

$$p(x|\theta, \phi) = \sum_{k=1}^{K} \theta_k f(x|\phi_k) \tag{67}$$

Then the log-likelihood can be written as:

$$\log p(x|\theta, \phi) = \log \left(\sum_{k=1}^{K} \theta_k f(x|\phi_k) \right) \tag{68}$$

The Fisher Information is given by:

$$\begin{aligned} I(\theta, \phi) &= \mathbb{E}\left[(\nabla \log p(x|\theta,\phi))(\nabla \log p(x|\theta,\phi))^T \right] \\ &= \left[\begin{array}{c} \frac{\partial}{\partial \theta} \log p(x|\theta,\phi) \\ \frac{\partial}{\partial \phi} \log p(x|\theta,\phi) \end{array} \right] \left[\begin{array}{c} \frac{\partial}{\partial \theta} \log p(x|\theta,\phi) \\ \frac{\partial}{\partial \phi} \log p(x|\theta,\phi) \end{array} \right]^T \end{aligned}$$

These derivatives can be computed as follows:

$$\begin{aligned} \frac{\partial}{\partial \theta_k} \log p(x|\theta, \phi) &= \frac{\partial}{\partial \theta_k} \log \left((\sum_{k=1}^{K} \theta_k f(x|\phi_k) \right) \\ &= \frac{f(x|\phi_k)}{\sum_{k'=1}^{K} \theta_{k'} f(x|\phi_{k'})} \\ \frac{\partial}{\partial \phi_k} \log p(x|\theta, \phi) &= \frac{\partial}{\partial \phi_k} \log \left((\sum_{k=1}^{K} \theta_k f(x|\phi_k) \right) \\ &= \frac{\theta_k \frac{\partial}{\partial \phi_k} f(x|\phi_k)}{\sum_{k'=1}^{K} \theta_{k'} f(x|\phi_{k'})} \end{aligned} \tag{69}$$

For any $u, v \in \mathbb{R}^K$ (with at least one nonzero), then the Fisher Information is positive definite as:

$$\begin{aligned} (u^T\ v^T) I \left(\begin{array}{c} u \\ v \end{array} \right) &= (u^T\ v^T) \mathbb{E} \left[\left[\begin{array}{c} \frac{\partial}{\partial \theta} \log \left(\sum_{k=1}^{K} \theta_k f(X|\phi_k) \right) \\ \frac{\partial}{\partial \phi} \log \left(\sum_{k=1}^{K} \theta k f(X|\phi_k) \right) \end{array} \right] \left[\begin{array}{c} \frac{\partial}{\partial \theta} \log \left(\sum_{k=1}^{K} \theta_k f(X|\phi_k) \right) \\ \frac{\partial}{\partial \phi} \log \left(\sum_{i=1}^{K} \theta_k f(X|\phi_k) \right) \end{array} \right]^T \right] \left(\begin{array}{c} u \\ v \end{array} \right) \\ &= \mathbb{E} \left[\left(u^T \frac{\partial}{\partial \theta} \log \left(\sum_{k=1}^{K} \theta_k f(X|\phi_k) \right) + v^T \frac{\partial}{\partial \theta} \log \left(\sum_{i=1}^{K} \theta_k f(X|\phi_i) \right) \right)^2 \right] \\ &= \mathbb{E} \left[\left(\frac{\sum_{k=1}^{K} u_k f(X|\phi_k) + v_k \theta_k \frac{\partial}{\partial \phi_k} f(X|\phi_k)}{\sum_{k=1}^{K} \theta_k f(X|\phi_k)} \right)^2 \right] \end{aligned}$$

This can be 0 if and only if

$$\sum_{k=1}^{K} u_k f(x|\phi_i) + v_k \theta_k \frac{\partial}{\partial \phi_k} f(x; \phi_k) = 0 \quad \forall x. \tag{70}$$

In case of exponential family emission models this cannot hold if all components are unique and all $\theta_k > 0$. Thus, if we assume all components are unique and every component has been observed at least once, the Fisher information matrix becomes non-singular.

E Alias Sampling Method

The alias sampling method is an efficient method for drawing samples from a K outcome discrete distribution in $O(1)$ amortized time and we describe it here for completeness. Denote by p_i for $i \in \{1 \ldots K\}$ the probabilities of a distribution over K outcomes from which we would like to sample. If p were the uniform distribution, i.e. $p_i = K^{-1}$, then sampling would be trivial. For the general case, we must pre-process the distribution p into a table of K triples of the form (i, j, π_i) as follows:

- Partition the indices $\{1 \ldots K\}$ into sets U and L where $p_i > K^{-1}$ for $i \in U$ and $p_i \leq K^{-1}$ for $i \in L$.
- Remove any i from L and j from U and add (i, j, p_i) to the table.
- Update $p_j = p_i + p_j - K^{-1}$ and if $p_j > K^{-1}$ then add j to U, else to L.

By construction the algorithm terminates after K steps; moreover, all probability mass is preserved either in the form of π_i associated with i or in the form of $K^{-1} - \pi_i$ associated with j. Hence, sampling from p can now be accomplished in constant time:

- Draw (i, j, π_i) uniformly from the set of k triples in K.
- With probability $K\pi_i$ emit i, else emit j.

Hence, if we need to draw from p at least K times, sampling can be accomplished in amortized $O(1)$ time.

F Applicability of ESCA

We begin with a simple Gaussian mixture model (GMM) with K components. Let $x_1, ..., x_n$ be *i.i.d.* observations, $z_1, ..., z_n$ be hidden component assignment variable and $\eta = \eta(\theta_1, ..., \theta_K, \mu_1, \Sigma_1, \mu_2, \Sigma_2, ..., \mu_K, \Sigma_K)$ be the parameters. Then the GMM fits into ESCA with sufficient statistics given by:

$$\begin{aligned} T(x_i, z_i) = [&\mathbb{1}\{z_i = 1\}, ..., \mathbb{1}\{z_i = K\}, \\ &x_i\mathbb{1}\{z_i = 1\}, ..., x_i\mathbb{1}\{z_i = K\}, \\ &x_i x_i^T\mathbb{1}\{z_i = 1\}, ..., x_i x_i^T\mathbb{1}\{z_i = K\}]. \end{aligned} \tag{71}$$

The conditional distribution for the E-step is:

$$p(z_i = k|x_i; \eta) \propto \theta_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \tag{72}$$

In the S-step we draw from this conditional distribution and the M-step, through inversion of link function, is:

$$\begin{aligned} \tilde{\theta}_k &= \frac{1}{n + K\alpha - K}\sum_{i=1}^{n}(\mathbb{1}\{z_i = k\} + \alpha - 1) \\ \tilde{\mu}_k &= \frac{\kappa_0\mu_0 + \sum_{i=1}^{n} x_i\mathbb{1}\{z_i = k\}}{\kappa_0 + \sum_{i=1}^{n}\mathbb{1}\{z_i = k\}} \\ \tilde{\Sigma}_k &= \frac{\Psi_0 + \kappa_0\mu_0\mu_0^T + \sum_{i=1}^{n} x_i x_i^T\mathbb{1}\{z_i = k\} - (\kappa_0 + \sum_{i=1}^{n}\mathbb{1}\{z_i = k\})\tilde{\mu}_k\tilde{\mu}_k^T}{\nu_0 + d + 2 + \sum_{i=1}^{n}\mathbb{1}\{z_i = k\}} \end{aligned} \tag{73}$$

and is only function of the sufficient statistics.

Next, we provide more details on how to employ ESCA for any conditional exponential family mixture model; i.e., in which n random variables x_i, $i = 1, \ldots, n$ correspond to observations, each distributed according to a mixture of K components, with each component belonging to the same exponential family of distributions (e.g., all normal, all multinomial, etc.), but with different parameters:

$$p(x_i|\phi) = \exp(\langle\psi(x_i), \phi\rangle - g(\phi)). \tag{74}$$

The model also has n latent variables z_i that specify the identity of the mixture component of each observation x_i, each distributed according to a K-dimensional categorical distribution. A set of K mixture weights θ_k, $k = 1, \ldots, K$, each of which is a probability (a real number between 0 and 1 inclusive) and collectively sum to one. A Dirichlet prior on the mixture weights with hyper-parameters α. A set of K parameters ϕ_k, $k = 1, \ldots, K$, each specifying the parameter of the corresponding mixture component. For example, observations distributed according to a mixture of one-dimensional Gaussian distributions will have a mean and variance for each component. Observations distributed according to a mixture of V-dimensional categorical distributions (e.g., when each observation is a word from a vocabulary of size V) will have a vector of V probabilities, collectively summing to 1. Moreover, we put a shared conjugate prior on these parameters:

$$p(\phi; n_0, \psi_0) = \exp\left(\langle\psi_0, \phi\rangle - n_0 g(\phi) - h(m_0, \psi_0)\right). \tag{75}$$

Then joint sufficient statistics would be given by:

$$\begin{aligned} T(z_i, x_i) = [&\mathbb{1}\{z_i = 1\}, ..., \mathbb{1}\{z_i = K\}, \\ &\psi(x_i)\mathbb{1}\{z_i = 1\}, ..., \psi(x_i)\mathbb{1}\{z_i = K\}] \end{aligned} \tag{76}$$

In the E-step of t^{th} iteration, we derive the conditional distribution $p(z_i|x_i, \eta)$, namely

$$\begin{aligned} p(z_i = k|x_i, \eta) &\propto p(x_i|\phi_k^{t-1}, z_i = k)p(z_i = k|\theta^{t-1}) \\ &= \frac{\theta_k^{t-1}p(x_i|\phi_k^{t-1})}{\sum_{k'}\theta_{k'}^{t-1}p(x_i|\phi_{k'}^{t-1})} \end{aligned} \tag{77}$$

In the S-step we draw z_i^t from this conditional distribution and the M-step through inversion of the link function yields:

$$\begin{aligned}
\nabla g(\tilde{\phi}_k) &= \frac{\phi_0 + \sum_i \psi(x_i)\mathbb{1}\{z_i = k\})}{n_0 + \sum_i \mathbb{1}\{z_i = k\}} \\
\text{or} \qquad \tilde{\phi}_k &= \xi^{-1}\left(\frac{\psi_0 + \sum_i \psi(x_i)\mathbb{1}\{z_i = k\}}{n_0 + \sum_i \mathbb{1}\{z_i = k\}}\right) \\
\tilde{\theta}_k &= \frac{\sum_i \mathbb{1}\{z_i = k\} + \alpha_k - 1}{n + \sum_k \alpha_k - k} \ .
\end{aligned} \tag{78}$$

This encompasses most of the popular mixture models (and with slight more work all the mixed membership or admixture models) with Binomial, multinomial, or Gaussian emission model, e.g. beta-binomials for identification, Dirichlet-multinomial for text or Gauss-Wishart for images as listed in Table 1.

Note further, ESCA is applicable to models such as restricted Boltzmann machines (RBMs) as well which are also in the exponential family. For example, if the data were a collection of images, each cell could independently compute the S-step for its respective image. For RBMs the cell would flip a biased coin for each latent variable, and for deep Boltzmann machines, the cells could perform Gibbs sampling.

To elabortate, consider 2-layer RBM (1 observed, 1 latent), then ESCA should work as it is. That is, we sample latent variables conditioned on data and weights. Then optimize weights, given latent variables and observed data. Now if we have deep RBM, i.e. one with many hidden layers. Then ESCA will have similar problem as Ising model. But there is a quick fix borrowing ideas from chromatic samplers.

```
for each iteration
    1. Sample all odd layers of the RBM
    2. Optimize for weights
    3. Sample all even layers of the RBM
    4. Optimize for weights
end for
```

We save a precise derivation and empirical evaluation for future work.

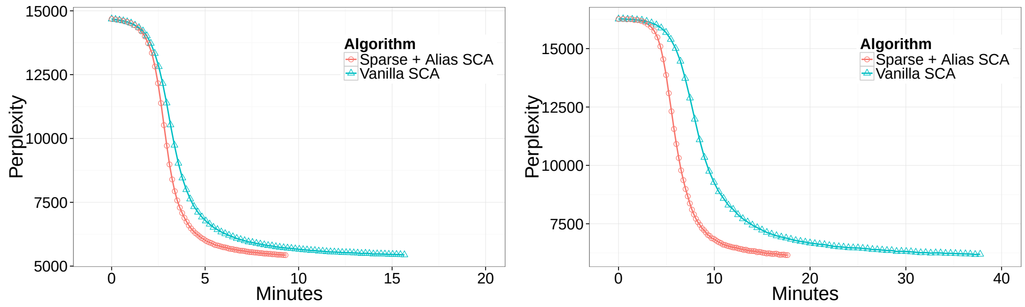
G More experimental results

In addition to the experiments reported in main paper, we perform another set of experiments. As before, to evaluate the strength and weaknesses of our algorithm, we compare against parallel and distributed implementations of CGS and CVB0.

Software & hardware All three algorithms were first implemented in the **Java** programming language. (We later switched to C++ for achieving better performance and those results are reported in the main paper.) To achieve good performance in the Java programming language, we use only arrays of primitive types and pre-allocate all of the necessary structures before the learning starts. We implement multithreaded parallelization within a node using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the Java binding to OpenMPI. We also implemented a version of SCA with a sparse representation for the array D of counts of topics per documents and Vose's alias method to draw from discrete distributions. We run our experiments on a small cluster of 16 nodes connected through 10Gb/s Ethernet. Each node has two 8-core Intel Xeon E5 processors (some nodes have Ivy Bridge processors while others have Sandy Bridge processors) for a total of 32 hardware threads per node and 256GB of memory.

Datasets We experiment on two datasets, both of which are cleaned by removing stop words and rare words: Reuters RCV1 and English Wikipedia. Our Reuters dataset is composed of 806,791 documents comprising 105,989,213 tokens with a vocabulary of 43,962 vocabulary words. Our Wikipedia dataset is composed of 6,749,797 documents comprising 6,749,797 tokens with a vocabulary of 291,561 words. (Note this Wikipedia dump was collected at a different time than the main paper, hence different numbers.) We also apply the SCA algorithm to a third larger dataset composed of more than 3 billion documents comprising more than 171 billion tokens with a vocabulary of about 140,000 words.

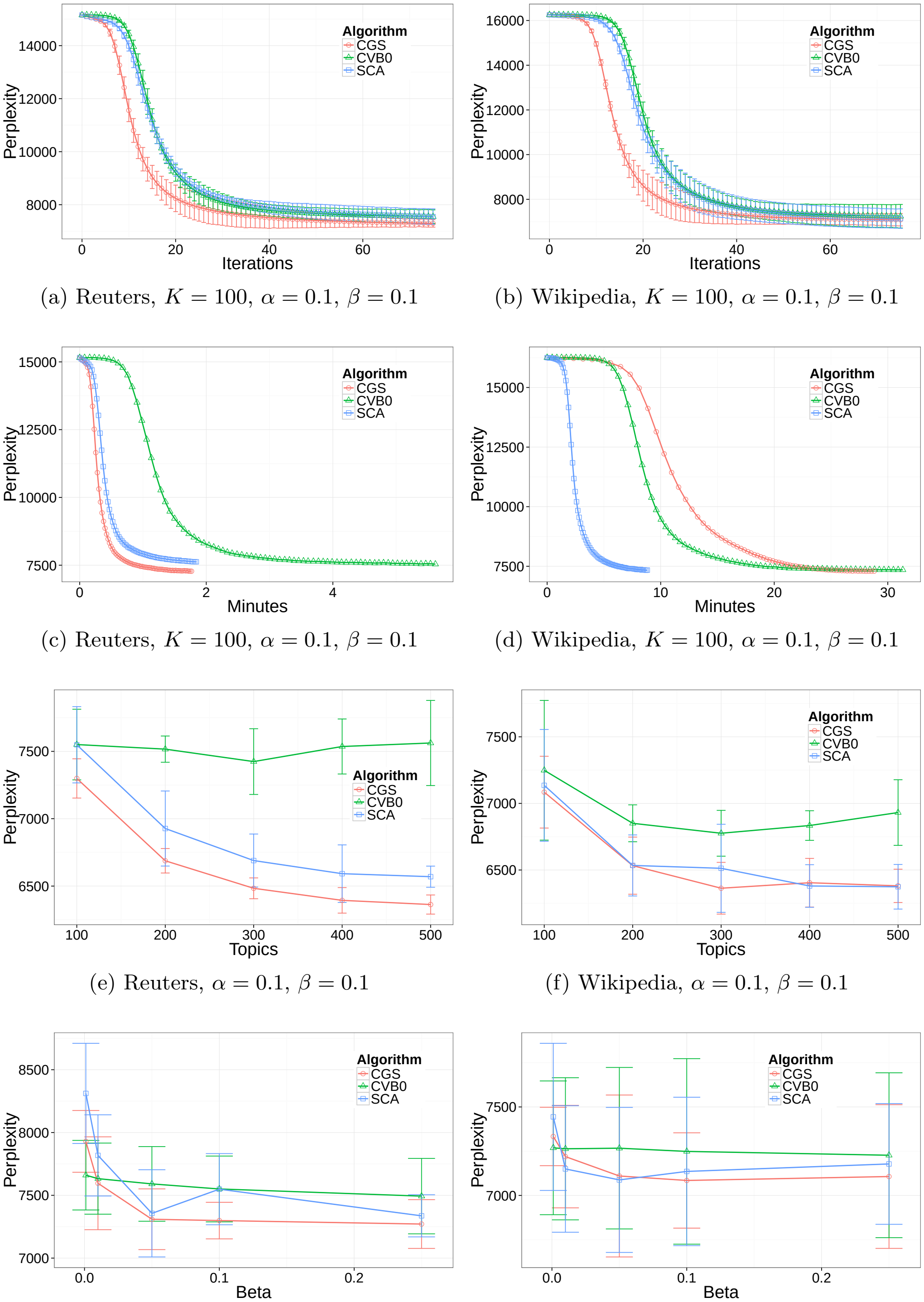
Protocol We use perplexity on held-out documents to compare the algorithms. When comparing algorithms trained on Wikipedia, we compute the perplexity of 10,000 Reuters documents. Vice versa, when comparing algorithms trained on Reuters, we compute the perplexity of 10,000 Wikipedia documents. We run four sets of experiment on each dataset: (1) how perplexity evolves for some numbers of training iterations (100 topics); (2) how perplexity evolves over time (100 topics); (3) perplexity as a function of the number of topics (75 iterations); and (4) perplexity as a function of the value of β (100 topics, 75 iterations). With the exception of the second experiment, we ran all experiments five times with five different seeds, and report the mean and standard deviation of these runs. The results are presented in Figure 6. We also ran an experiment to compare vanilla SCA and its improved version that uses a sparse representation and Vose's alias method for discrete sampling. The results are presented in Figure 5.



(a) Wikipedia, $K = 200$, $\alpha = 0.1$, $\beta = 0.1$

(b) Wikipedia, $K = 500$, $\alpha = 0.1$, $\beta = 0.1$

Figure 5: Evolution of perplexity over time for plain SCA and a sparse one using the alias method.



(a) Reuters, $K = 100$, $\alpha = 0.1$, $\beta = 0.1$

(b) Wikipedia, $K = 100$, $\alpha = 0.1$, $\beta = 0.1$

(c) Reuters, $K = 100$, $\alpha = 0.1$, $\beta = 0.1$

(d) Wikipedia, $K = 100$, $\alpha = 0.1$, $\beta = 0.1$

(e) Reuters, $\alpha = 0.1$, $\beta = 0.1$

(f) Wikipedia, $\alpha = 0.1$, $\beta = 0.1$

(g) Reuters, $K = 100$, $\alpha = 0.1$

(h) Wikipedia, $K = 100$, $\alpha = 0.1$

Figure 6: Evolution of perplexity on Wikipedia and Reuters over number of iterations, time, number of topics, value of β. Here SCA does not use alias method or sparsity and hence slower.

Topics

Here are the first five topics inferred via ESCA on LDA from both PubMed and Wikipedia:

PubMed				
Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
seizures	data	local	gene	state
epilepsy	information	block	transcript	change
seizure	available	lidocaine	exon	transition
epileptic	provide	anethesia	genes	states
temporal_lobe	regarding	anethetic	expression	occur
anticonvulsant	sources	acupuncture	region	process
convulsion	literature	bupivacaine	mrna	shift
kindling	concerning	anaesthesia	mouse	condition
partial	limited	under	expressed	changed
generalized	provided	anaesthetic	human	dynamic

Wikipedia				
Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
hockey	medical	von	boy	music
ice	medicine	german	youth	music
league	hospital	karl	boys	pop
played	physician	carl	camp	music
junior	doctor	friedrich	girl	artists
nhl	clinical	wilhelm	scout	electronic
professional	md	johann	girls	duo
games	physicians	ludwig	guide	genre
playing	doctors	prussian	scouts	genres
national	surgeon	heinrich	scouting	musicians