Extreme Stochastic Variational Inference: Distributed Inference for Large Scale Mixture Models

Jiong Zhang*
zhangjiong724@utexas.edu
UT Austin

Parameswaran Raman*
params@ucsc.edu
UC Santa Cruz

Shihao Ji
sji@cs.gsu.edu
Georgia State University

Hsiang-Fu Yu †
rofu.yu@gmail.com
Amazon

S.V.N. Vishwanathan
vishy@amazon.com
Amazon

Inderjit S. Dhillon †
inderjit@cs.utexas.edu
UT Austin & Amazon

Abstract

Mixture of exponential family models are among the most fundamental and widely used statistical models. Stochastic variational inference (SVI), the state-of-the-art algorithm for parameter estimation in such models is inherently serial. Moreover, it requires the parameters to fit in the memory of a single processor; this poses serious limitations on scalability when the number of parameters is in billions. In this paper, we present extreme stochastic variational inference (ESVI), a distributed, asynchronous and lock-free algorithm to perform variational inference for mixture models on massive real world datasets. ESVI overcomes the limitations of SVI by requiring that each processor only access a subset of the data and a subset of the parameters, thus providing data and model parallelism simultaneously. Our empirical study demonstrates that ESVI not only outperforms VI and SVI in wallclock-time, but also achieves a better quality solution. To further speed up computation and save memory when fitting large number of topics, we propose a variant ESVI-TOPK which maintains only the top-k important topics. Empirically, we found that using top 25% topics suffices to achieve the same accuracy as storing all the topics.

† This work was done while the authors were at UT Austin. ∗ denotes equal contribution.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

1 Introduction

Mixture of exponential family models generalize a wide collection of popular latent variable models such as *Latent Dirichlet Allocation (LDA)*, *Gaussian Mixture Models (GMM)*, and *Mixed Membership Stochastic Block Models (MM-SBM)*. In recent years, variational inference (VI) has emerged as a powerful technique for parameter estimation in these Bayesian models [14], [3]. One attractive property of VI is that it reduces parameter estimation to the task of optimizing a objective function, often with a well defined "structure". This opens up the possibility of bringing to bear mature tools from optimization to tackle massive problems. Traditionally, VI in mixture models involves alternating between updating *global variables* and *local variables*. Both these operations involve accessing all the data points. Large datasets are usually stored on disk, and the cost of accessing every datapoint to perform updates is prohibitively high.

The first approach to tackle this, is to divide the data across multiple machines and use a distributed framework such as map-reduce to aggregate the computations [11]. The second approach, is to exploit the underlying structure of the optimization problem to reduce the number of iterations (and therefore the corresponding data access) [7]. The key observation here is that the optimization problem corresponding to the local variables is *separable*, that is, it can be written as a sum of functions, where each function only depends on one data point. Therefore, one can use stochastic optimization to update the local variables. Moreover, in Stochastic Variational Inference (SVI) [7], even before one pass through the dataset, the global variables are updated multiple times, and therefore the model parameters converge rapidly towards their final values. The argument is similar in spirit to how stochastic op-

timization outperforms batch algorithms for maximum aposteriori (MAP) estimation [4].

With the advent of the big-data era, we now routinely deal with industry-scale problems involving billions of documents and tokens. Such massive datasets pose another challenge, which, unfortunately neither VI nor SVI are able to address; namely, the set of parameters is so large that *all the parameters do not fit* on a single processor[1]. For instance, if we have D dimensional data and K mixture components, then the parameter size is $O(DK)$. If D is of the order of millions and K is in the 100's or 1000's, modest numbers by todays standards, the parameter size is a few 100s of GB (see our experiments in Section 5).

In this paper, we propose a new framework, Extreme Stochastic Variational Inference (ESVI) to address these challenges. The main contributions of this paper are:

1. We develop a novel approach to achieve *simultaneous data and model parallelism* in *mixture models* by exploiting the following key idea: instead of updating all the K coordinates of a local variable and then updating all K global variables, we propose updating a small subset of the local variables and the corresponding global variables (see Theorem 1 for proof of correctness). The global variables are exchanged across the processors, and this ensures mixing (see Section 4.2 for more details). This seemingly simple idea has some powerful consequences. It allows multiple processors to *simultaneously perform parameter updates independently.*
2. Using a classic owner-computes paradigm, we make ESVI *asynchronous* and *lock-free*, and thus avoid expensive bulk synchronization between processors. We present an extensive empirical study to evaluate the performance of ESVI by applying it to GMM and LDA models on several large real-world datasets. We find that ESVI *outperforms VI and SVI both in terms of time as well as the quality of solutions obtained.* For practitioners, ESVI offers the advantage of *not having to tune a learning rate*, since it makes closed form parameter updates unlike SVI.
3. We develop a variant ESVI-LDA-TOPK to *speed up computation and save memory when fitting large number of topics.* Empirically, we found that using the most frequent 25% of the topics was enough to obtain the same level of accuracy as storing all the topics.

[1]The discussion in this paper applies to the shared memory, distributed memory, as well as hybrid settings, and therefore we will use the term processor to denote either a thread or a machine.

To the best of our knowledge there is no existing algorithm for VI that sports these desirable properties.

The rest of the paper is structured as follows: We present an exhaustive study of related work in Section 2. We briefly review VI and SVI in Section 3. We present our new algorithm ESVI in Section 4, and discuss its advantages. Empirical evaluation is presented in Section 5 and Section 6 concludes the paper.

2 Related Work

Recent research on VI has focused on extending it to non-conjugate models [15] and developing variants that can scale to large datasets such as SVI [7]. Other than the fact that SVI is inherently serial, it also suffers from another drawback: storage of the entire $D \times K$ matrix θ on a single machine. On the other hand, our method, ESVI, exhibits model parallelism; each processor only needs to store $1/P$ fraction of θ. Black-box variational inference (BBVI) [12] generalizes SVI beyond conditionally conjugate models. The paper proposes a more generic framework by observing that the expectation in the ELBO can be exploited directly to perform stochastic optimization. We view this line of work as complementary to our research. It would be interesting to verify if an ESVI like scheme can also be applied to BBVI.

There has been a flurry of work in the past few years in developing data-parallel distributed methods for Approximate Bayesian Inference. One such popular work includes a classic Map-Reduce style inference algorithm [11], where the data is divided across several worker nodes and each of them perform VI updates in parallel until a final synchronization step during which the parameters from the slaves are combined to produce the final result. This method suffers from the well-known *curse of the last reducer*, that is, a single slow machine can dramatically slow down the performance. *ESVI does not suffer from this problem*, because our asynchronous and lock-free updates avoid bulk synchronization altogether.

[5] presents an algorithm that applies VI to the streaming setting by performing asynchronous Bayesian updates to the posterior as batches of data arrive continuously, which is similar in spirit to Hogwild [13]. Their approach uses a parameter server to enable asynchronous local updates. Unlike ESVI, their work cannot guarantee that - (a) each worker works on the latest parameters, (b) the global parameters are all parallely updated. In [1] the authors present *Incremental Variational Inference* which is also a distributed variational inference algorithm, however it is also only data-parallel. Besides, it requires tuning of a step-size and sequential access of global parameters. *ESVI avoids these*

drawbacks.

A number of data-parallel approaches exist in the Exact Bayesian Inference literature as well. [6] is a distributed MCMC based approach where workers perform MCMC updates locally and these are aggregated by maintaining a posterior server. [17] proposed a distributed asynchronous algorithm for parameter estimation in LDA [2]. However, the algorithm is specialized to collapsed Gibbs sampling for LDA, and it is unclear how to extend it to other, more general, mixture models. *ESVI in contrast is a purely VI based method and provides model-parallelism in addition to data-parallelism.*

Somewhat close to our ESVI-TOPK approach is *Memoized Online Variational Inference for DP Mixture Models* [8]. This paper describes the application of Expectation Truncation to mixture models. In their L-sparse method, unused dimensions are set to zero and used dimensions are shifted. In ESVI-TOPK, unused dimensions are averaged (1-sum of used dimensions).

Another related line of work is *Sparse EM* [10]. There are some high-level similarities to ESVI in that both the methods update a subset of latent variables at any given time while keeping others frozen. However there are some crucial differences: (a) Sparse EM is not a parallel algorithm while ESVI is, (b) Sparse EM needs to iterate between sparse EM update and full EM update (to select active dimensions occasionally) while each ESVI worker's job queue will continuously distribute Z's dimensions to ensure a good mixing, (c) Sparse EM selects active dimensions based on values of Z, while ESVI is designed to ensure the active dimensions of each worker is an unbiased sample of all dimensions.

Since the coordinate-ascent algorithm in VI can be formulated as a message passing scheme applied to general graphical models, we believe ESVI is also related to Variational Message Passing [16]. This connection could be made more concrete if we assume a Mixture Model setup in both cases. Both the d-VMP algorithm (Algorithm 2 in [9]) and ESVI de-couple the global parameters to make the updates scalable, however they differ in some fundamental aspects. d-VMP defines a disjoint partitioning of the global parameters based on their markov-blankets. In contrast, ESVI completely decentralizes the global parameter updates by requiring that the local variables (or assignment vector z_i) need to only satisfy local summation constraints (as discussed in Theorem 1 in Section 4). As a side-note, the local updates in d-VMP algorithm do not seem to be de-coupled across the mixture components, whereas this holds true in the case of ESVI.

3 Parameter Estimation for Mixture of Exponential Families

3.1 Generative Model

The following data generation scheme underlies a mixture of exponential family model (Table 1 defines the notations):

Prior:

$$p(\pi|\alpha) = \text{Dirichlet}(\alpha) \tag{1}$$

$$p(\theta_k|n_k,\nu_k) = \exp\left(\langle n_k \cdot \nu_k, \theta_k\rangle - n_k \cdot g(\theta_k) - h(n_k,\nu_k)\right) \tag{2}$$

where, n_k and ν_k are the parameters of the *conjugate prior* $p(\theta_k|n_k,\nu_k)$.

Likelihood:

$$p(z_i|\pi) = \text{Multinomial}(\pi) \tag{3}$$

$$p(x_i|z_i,\theta) = \exp\left(\langle \phi(x_i,z_i), \theta_{z_i}\rangle - g(\theta_{z_i})\right) \tag{4}$$

where, ϕ denotes the sufficient statistics. Observe that $p(\theta_k|n_k,\nu_k)$ is *conjugate* to $p(x_i|z_i=k,\theta_k)$, while $p(\pi|\alpha)$ is *conjugate* to $p(z_i|\pi)$.

Joint Distribution:

$$p(x,\pi,z,\theta|\alpha,n,\nu) = p(\pi|\alpha)\cdot\prod_{k=1}^{K} p(\theta_k|n_k,\nu_k)\cdot \prod_{i=1}^{N} p(z_i|\pi)\cdot p(x_i|z_i,\theta) \tag{5}$$

3.2 Variational Inference and Stochastic Variational Inference

The goal of inference is to estimate $p(\pi,z,\theta|x,\alpha,n,\nu)$. This however, involves an intractable marginalization. Therefore, variational inference [3] approximates this distribution with a fully factorized distribution [2]:

$$q\left(\pi,z,\theta|\tilde{\pi},\tilde{z},\tilde{\theta}\right) = q(\pi|\tilde{\pi})\cdot\prod_{i=1}^{N} q(z_i|\tilde{z}_i)\cdot\prod_{k=1}^{K} q\left(\theta_k|\tilde{\theta}_k\right). \tag{6}$$

Note that $\tilde{z}_i \in \Delta_K$ and $z_{i,k} = q(z_i=k|\tilde{z}_i)$. Moreover, each of the factors in the variational distribution is assumed to belong to the same exponential family as their full conditional counterparts in (5). The variational parameters are estimated by maximizing the

[2] A $\sim$ over a symbol is used to denote that it is a parameter of the variational distribution. See Table 1.

Symbol	Definition
N	total number of observations
D	total number of dimensions
K	total number of mixture components
Δ_K	K-dimensional simplex
$x = \{x_1, \ldots, x_N\}$, $x_i \in \mathbb{R}^D$	observations
$z = \{z_1, \ldots, z_N\}$, $z_i \in \Delta_K$	the component data point x_i was drawn from (local variable)
$\theta = \{\theta_1, \ldots, \theta_K\}$, $\theta_k \in \mathbb{R}^D$	sufficient statistics of the exponential family distribution (global variable)
$\pi \in \Delta_K$	mixing coefficients (global variable)
$\tilde{z} = \{\tilde{z}_1, \ldots, \tilde{z}_N\}$, $\tilde{z}_i \in \Delta_K$	variational parameter for z (local variable)
$\tilde{\theta} = \left\{\tilde{\theta}_1, \ldots, \tilde{\theta}_K\right\}$, $\tilde{\theta}_k \in \mathbb{R}^D$	variational parameter for θ (global variable)
$\tilde{\pi} \in \Delta_K$	variational parameter for π (global variable)

Table 1: Notations for Mixture of Exponential Family Model. $\sim$ denotes variational parameters.

following evidence lower-bound (ELBO) [3]:

$$\mathcal{L}\left(\tilde{\pi}, \tilde{z}, \tilde{\theta}\right) = \mathbb{E}_{q\left(\pi,z,\theta|\tilde{\pi},\tilde{z},\tilde{\theta}\right)}\left[\log p\left(x, \pi, z, \theta|\alpha, n, \nu\right)\right] - \mathbb{E}_{q\left(\pi,z,\theta|\tilde{\pi},\tilde{z},\tilde{\theta}\right)}\left[\log q\left(\pi, z, \theta|\tilde{\pi}, \tilde{z}, \tilde{\theta}\right)\right] \tag{7}$$

VI performs coordinate ascent updates on $\mathcal{L}$ by optimizing each set of variables, one at a time.

Update for $\tilde{\pi}_k$:

$$\tilde{\pi}_k = \alpha + \sum_{i=1}^{N} \tilde{z}_{i,k} \tag{8}$$

Update for $\tilde{\theta}_k$: The components of $\tilde{\theta}_k$ namely $\tilde{n}_k$ and $\tilde{\nu}_k$ are updated as follows:

$$\tilde{n}_k = n_k + N_k \tag{9}$$

$$\tilde{\nu}_k = n_k \cdot \nu_k + N_k \cdot \bar{x}_k \tag{10}$$

where $N_k = \sum_{i=1}^{N} \tilde{z}_{i,k}$ and $\bar{x}_k = \frac{1}{N_k} \sum_{i=1}^{N} \tilde{z}_{i,k} \cdot \phi\left(x_i, k\right)$.

Update for $\tilde{z}_i$: Let u_i be a K dimensional vector whose k-th component is given by

$$u_{i,k} = \psi\left(\tilde{\pi}_k\right) - \psi\left(\sum_{k'=1}^{K} \tilde{\pi}_{k'}\right) + \left\langle \phi\left(x_i, k\right), \mathbb{E}_{q\left(\theta_k|\tilde{\theta}_k\right)}\left[\theta_k\right] \right\rangle - \mathbb{E}_{q\left(\theta_k|\tilde{\theta}_k\right)}\left[g\left(\theta_k\right)\right] \tag{11}$$

$$\tilde{z}_{i,k} = \frac{\exp\left(u_{i,k}\right)}{\sum_{k'=1}^{K} \exp\left(u_{i,k'}\right)} \tag{12}$$

It has to be noted that the summation term $\psi\left(\sum_{k'=1}^{K} \tilde{\pi}_{k'}\right)$ cancels out during the $\tilde{z}_{i,k}$ update in (12).

The VI algorithm [14, 3] iteratively updates all the local variables $\tilde{z}_i$ before updating the global variables $\tilde{\pi}_k$ and $\tilde{\theta}_k$ (see Algorithm 1). In contrast, SVI [7], updates $\tilde{z}_i$ corresponding to one data point x_i, followed by updating the global parameters $\tilde{\pi}$ and $\tilde{\theta}$ (see Algorithm 2).

Algorithm 1 VI

for $i = 1, \ldots, N$ **do**
  Update $\tilde{z}_i$ using (12)
end for
for $k = 1, \ldots K$ **do**
  Update $\tilde{\pi}_k$ using (8)
  Update $\tilde{\theta}_k$ using (9) and (10)
end for

Algorithm 2 SVI

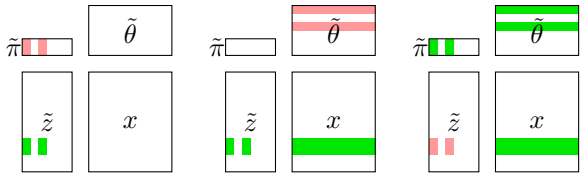
Generate step size sequence $\eta_t \in (0, 1)$
Pick an $i \in \{1, \ldots, N\}$ uniformly at random
  Update $\tilde{z}_i$ using (12)
  for $k = 1, \ldots K$ **do**
    Update $\tilde{\pi}_k \leftarrow (1 - \eta_t)\tilde{\pi}_k + \eta_t\left(\alpha + N \cdot \tilde{z}_{i,k}\right)$
    $\hat{\tilde{\theta}}_k = \{n_k + N \cdot \tilde{z}_i,\ n_k \cdot \nu_k + N \cdot \tilde{z}_{i,k} \cdot \phi\left(x_i, k\right)\}$
    Update $\tilde{\theta}_k \leftarrow (1 - \eta_t)\tilde{\theta}_k + \eta_t \hat{\tilde{\theta}}_k$
  end for

4 Extreme Stochastic Variational Inference (ESVI)

Algorithm 3 ESVI

Sample $i \in \{1, \ldots, N\}$
  Select $\mathcal{K} \subset \{1, \ldots, K\}$
    Update $\tilde{z}_{i,k}$ for all $k \in \mathcal{K}$ (see below)
    Update $\tilde{\pi}_k$ for all $k \in \mathcal{K}$ using (8)
    Update $\tilde{\theta}_k$ for all $k \in \mathcal{K}$ using (9), (10)

Algorithm 3 illustrates our proposed updates in ESVI. Before we discuss why this update is advantageous for parallelization, let us first study why updating a subset of coordinates of $\tilde{z}_i$ is justified. Plugging



(a) $\tilde{\pi}$ update (b) $\tilde{\theta}$ update (c) $\tilde{z}$ update

Figure 1: Access pattern during ESVI updates. Green indicates the variable or data point being read, while Red indicates it being updated.

in (5) and (6) into (7), and restricting our attention to the terms in the above equation which depend on z_i, substitute (4), $\tilde{z}_{i,k} = q(z_i = k|\tilde{z}_i)$ and $\mathbb{E}_{q(\pi|\tilde{\pi})}[\log p(z_i = k|\pi)] = \psi(\tilde{\pi}_k) - \psi\left(\sum_{k'=1}^{K} \tilde{\pi}_{k'}\right)$, to obtain the following objective function,

$$\mathcal{L}\left(\tilde{z}_i|\tilde{\pi},\tilde{\theta}\right) = \sum_{k=1}^{K} \tilde{z}_{i,k} \cdot \left(\psi(\tilde{\pi}_k) - \psi\left(\sum_{k'=1}^{K} \tilde{\pi}_{k'}\right)\right) + \sum_{k=1}^{K} \tilde{z}_{i,k} \cdot \left(\left\langle \phi(x_i,k), \mathbb{E}_{q(\theta_k|\tilde{\theta}_k)}[\theta_k] \right\rangle - \mathbb{E}_{q(\theta_k|\tilde{\theta}_k)}[g(\theta_k)] - \log \tilde{z}_{i,k}\right). \quad (13)$$

Now using the definition of $u_{i,k}$ in (11), one can compactly rewrite the above objective function as

$$\mathcal{L}\left(\tilde{z}_i|\tilde{\pi},\tilde{\theta}\right) = \sum_{k=1}^{K} \tilde{z}_{i,k} \cdot (u_{i,k} - \log \tilde{z}_{i,k}). \quad (14)$$

Moreover, to ensure that $\tilde{z}_{i,k}$ is a valid distribution, one needs to enforce the following constraints:

$$\sum_k \tilde{z}_{i,k} = 1, \quad 0 \le \tilde{z}_{i,k} \le 1. \quad (15)$$

Theorem 1 shows that one can find a closed form solution to maximizing (14) even if we restrict our attention to a subset of coordinates.

Theorem 1 *For $2 \le K' \le K$, let $\mathcal{K} \subset \{1,\dots,K\}$ be s.t. $|\mathcal{K}| = K'$. For any $C > 0$, the problem*

$$\max_{z_i \in \mathbb{R}^{K'}} \mathcal{L}_{\mathcal{K}} = \sum_{k\in\mathcal{K}} \tilde{z}_{i,k} \cdot u_{i,k} - \tilde{z}_{i,k} \cdot \log \tilde{z}_{i,k} \quad s.t. \quad \sum_{k\in\mathcal{K}} \tilde{z}_{i,k} = C \quad and \quad 0 \le \tilde{z}_{i,k}, \quad (16)$$

has the closed form solution:

$$\tilde{z}^*_{i,k} = C\frac{\exp(u_{i,k})}{\sum_{k'\in\mathcal{K}} \exp(u_{i,k'})}, \ for\ k \in \mathcal{K}. \quad (17)$$

Proof Proof is given in Appendix A.

Theorem 1 suggests the following strategy: start with a feasible $\tilde{z}_i$, pick, say, a pair of coordinates $\tilde{z}_{i,k}$ and $\tilde{z}_{i,k'}$ and let $\tilde{z}_{i,k} + \tilde{z}_{i,k'} = C$. Solve (16), which has the closed form solution (17). Clearly, if $\tilde{z}_i$ satisfied constraints (15) before the update, it will continue to satisfy the constraints even after the update. On the other hand, the conditional ELBO (14) increases as a result of the update. Therefore, ESVI is a valid coordinate ascent algorithm for improving the ELBO (7).

4.1 Access Patterns

In this section we compare the access patterns of variables in the three algorithms to gain a better understanding of their abilities to be parallelized efficiently. In VI, the updates for $\tilde{\pi}$ and $\tilde{\theta}$ requires access to all $\tilde{z}_i$, while update to $\tilde{z}_i$ requires access to $\tilde{\pi}$ and all $\tilde{\theta}_k$. On the other hand, in case of SVI, the access pattern is somewhat different. The updates for $\tilde{\pi}$ and $\tilde{\theta}$ require access to only the $\tilde{z}_i$ that was updated, however the update to $\tilde{z}_i$ still requires access to $\tilde{\pi}$ and all the $\tilde{\theta}_k$. This is a crucial bottleneck to model parallelism. Refer to Figure 9 and Figure 10 in the Appendix for a visual illustration.

In contrast, the following access pattern of ESVI allows multiple processors to access and update mutually exclusive subsets of coordinates $\mathcal{K}$ independently (See Figure 1 for an illustration):

- The update for $\tilde{\pi}$ (8) requires access to the coordinates $\tilde{z}_{i,k}$ for $k \in \mathcal{K}$.
- The update for $\tilde{\theta}$ (9) and (10) requires access to $\tilde{z}_{i,k}$ for $k \in \mathcal{K}$.
- The update to $\tilde{z}_{i,k}$ for $k \in \mathcal{K}$ requires access to $\tilde{\pi}_k$ and $\tilde{\theta}_k$ for $k \in \mathcal{K}$.

4.2 Parallelization

In this sub-section, we describe the parallel asynchronous algorithm of ESVI. Let P denote the number of processors, and let $\mathcal{I}_p \subset \{1,\dots,N\}$ denote indices of the data points owned by processor p. $\tilde{z}_i$ for $i \in \mathcal{I}_p$ are local variables assigned to processor p. The global variables are split across the processors. Let $\mathcal{K}_p \subset \{1,\dots,K\}$ denote the indices of the rows of $\tilde{\theta}$ currently residing in processor p. Then processor p can update any $\tilde{z}_{i,k}$ for $i \in \mathcal{I}_p$ and $k \in \mathcal{K}_p$. Finally, we need to address the issue of how to communicate $\tilde{\theta}_k$ across processors. For this, we follow the asynchronous communication scheme outlined by [19] and [18]. Figure 2 is an illustration of how this works pictorially. We partition the data and the corresponding $\tilde{z}_{1:N}$ variables across the processors. Each processors maintains its own queue. Once partitioned, the $\tilde{z}$ variables never

Algorithm 4 Parallel-ESVI Algorithm

P: total number of workers, T: maximum computing time
$\mathcal{I}_p$: data points owned by worker p,
$\mathcal{K}_p$: global parameters owned by worker p (concurrent queue)
Initialize global parameters $\tilde{\theta}^0$, $\tilde{\pi}^0$
for worker $p = 1 \dots P$ asynchronously **do**
  while Stop criteria not satisfied **do**
    Pick a subset $\mathbf{k}_s \subset \mathcal{K}_p$
    for All data point $i \in \mathcal{I}_p$ **do**
      for $k \in \mathbf{k}_s$ **do**
        Compute $\tilde{z}^*_{ik}$ using (17)
        $\tilde{\pi}_k += \tilde{z}^*_{ik} - \tilde{z}_{ik}$
        $\tilde{n}_k += \tilde{z}^*_{ik} - \tilde{z}_{ik}$
        $\tilde{\nu}_k += (\tilde{z}^*_{ik} - \tilde{z}_{ik}) \times \phi(x_i, k)$
        $\tilde{z}_{ik} \leftarrow \tilde{z}^*_{ik}$
        Pick a random worker p' and send $\tilde{\pi}_k$ and $\tilde{\theta}_k$, push k to $\mathcal{K}_{p'}$
      end for
    end for
  end while
end for

move. On the other hand, the $\tilde{\theta}$ variables move nomadically[3] between processors. Each processor performs ESVI updates using the current subset of $\tilde{\theta}$ variables that it currently holds. Then the variables are passed on to the queue of another randomly chosen processor as shown in the second sub-figure in Figure 2. It is this nomadic movement of the $\tilde{\theta}$ variables that ensures proper mixing and convergence. The complete algorithm for parallel-ESVI is outlined in Algorithm 4.

4.3 Comparison and Complexity

ESVI updates are stochastic w.r.t. the coordinates, however the update in each coordinate is exact using (17). In contrast, SVI stochastically samples the data and performs inexact or noisy updates and does not guarantee each step to be an ascent step. Moreover, given a $N \times D$ dataset and fixing K clusters, by simple calculation we can see that to update all $\tilde{z}_{ik}$ once, VI requires $O(DK)$ updates on $\tilde{\theta}$, while SVI needs $O(NDK)$ and parallel ESVI needs $O(PDK)$.

5 Experiments

In our experiments[4], we compare our proposed **ESVI-GMM** and **ESVI-LDA** methods against VI and SVI.

[3]Nomadic movement [19] refers to the distributed setup, where the ownership of parameters rapidly keeps changing after every update. Figure 2 illustrates this.

[4]Our code and scripts will be made publicly available.

To handle large number of topics in LDA, we also implemented a more efficient version **ESVI-LDA-TOPK**. We use real-world datasets of varying scale as described in Table 2. We used a large-scale parallel computing platform with node configuration of 20 Intel Xeon E5-2680 CPUs and 256 GB memory. We implemented ESVI-LDA in C++ using MPICH, OpenMP and Intel TBB. For Distributed-VI and SVI implementations, we modified the authors' original code in C[5]. More details on the parameter settings and update equations are available in Appendix C, D, E.

	# documents	# vocabulary	#words
AP-DATA	2,246	10,473	912,732
NIPS	1,312	12,149	1,658,309
Enron	37,861	28,102	6,238,796
Ny Times	298,000	102,660	98,793,316
PubMed	8,200,000	141,043	737,869,083
UMBC-3B	40,599,164	3,431,260	3,013,004,127

Table 2: Data Characteristics

5.1 ESVI-GMM

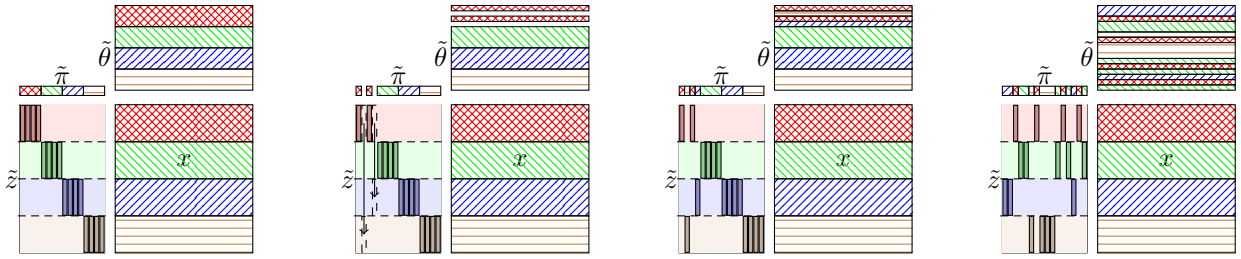
We first compare ESVI-GMM with SVI and VI in the Single Machine Single thread setting. We use a TOY dataset which consists of $N = 29,983$ data points, $D = 128$ dimensions and the AP-DATA dataset which consists of $N = 2,246$ data points, $D = 10,473$ dimensions. In both cases, we set the number components $K = 256$. We plot the performance of the methods (ELBO) as a function of time. ESVI-GMM outperforms SVI and VI by quite some margin. For Multi Machine case, we use the NIPS and NY Times datasets and only compare against VI (SVI does not apply; it needs to update all its K global parameters which is infeasible when K is large). Although typically these datasets do not demand running on multiple machines, out intention here is to demonstrate scalability to very large number of components ($K = 1024$) and dimensions, which is typically the case in large scale text datasets with millions of word count features. Traditionally, GMM inference methods have not been able to handle such a scale. Figure 3 clearly indicates that ESVI-GMM is able to outperform VI by a good margin.

5.2 ESVI-LDA

5.2.1 Single Machine Single thread

We compare serial versions of the methods on Enron and NY Times datasets which are medium sized and fit on a single-machine. On both datasets, we run with single machine and single thread. For Enron, we set #

[5]http://www.cs.princeton.edu/~blei/lda-c/. Distributed-VI was implemented in Map-Reduce style



(a) Initial assignment of $\tilde{\theta}$ and x. We plot diagonal initialization while in real case random initialization is used.

(b) Worker 1 finishes processing $\{2,4\} \in \mathcal{K}_1$, it sends them over to a random worker. Here, $\tilde{\theta}_2$ is sent from worker 1 to 4 and $\tilde{\theta}_4$ from 1 to 3.

(c) Upon receipt, the column is processed by the new worker. Here, worker 4 can now operate on $\tilde{\theta}_2$ and 3 on $\tilde{\theta}_4$

(d) During the execution of the algorithm, the ownership of the global parameters $\tilde{\theta}_k$ changes.

Figure 2: Illustration of the communication pattern in ESVI (asynchronous) algorithm. Parameters of same color are in memory of the same worker. Horizontal and Vertical lines indicate the two directions of partitioning data and parameters. Data x is partitioned horizontally along N and vertically along D. Local parameter $\tilde{z}$ is partitioned horizontally along N and vertically along K. Global parameters - $\tilde{\pi}$ is partitioned vertically along K, and $\tilde{\theta}$ is partitioned horizontally along K and vertically along D. $\tilde{\pi}$ and $\tilde{\theta}$ are nomadically exchanged.

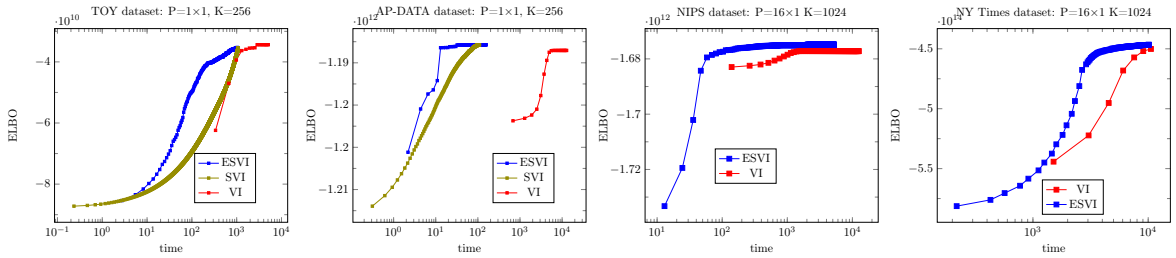


Figure 3: Comparison of ESVI-GMM, SVI and VI. $P = N \times n$ denotes N machines each with n threads.

of topics $K = 8, 16, 20, 32, 64, 128, 256$. For NY Times, we set $K = 8, 16, 32, 64$. To keep the plots concise, we only show results with $K = 16, 64, 128$ in Figure 4 (two left-most plots). ESVI-LDA performs better than VI and SVI in both the datasets for all values of K.

5.2.2 Single Machine Multi Core

We evaluate the performance of distributed ESVI-LDA against a map-reduce based distributed implementation of VI, and the streaming SVI method [5]. We vary the number of cores as $4, 8, 16$. This is shown in Figure 4 (two right-most plots). For Enron dataset, we use $K = 128$ and for NY Times dataset, we use $K = 64$. ESVI-LDA outperforms VI and SVI consistently in all scenarios. In addition, we observe that both the methods benefit reasonably when we provide more cores to the computation. We observe that ESVI-LDA-TOPK, which stores only top 1/4-th of K topics performs the best on both datasets.

5.2.3 Multi Machine Multi Core

We stretch the limits of ESVI-LDA method and compare it against distributed VI on large datasets: PubMed and UMBC-3B. UMBC-3B is a massive dataset with 3 billion tokens and a vocabulary of 3 million. We use 32 nodes and 16 cores, and fit $K = 128$ topics. As the results in Figure 5 demonstrate, ESVI-LDA achieves a better solution than distributed VI in all cases. On the largest dataset UMBC-3B, ESVI-LDA is also much faster than VI. In PubMed, VI has a slight initial advantage, however eventually ESVI progresses much faster towards a better ELBO. ESVI-LDA-TOPK is particularly better than the other two methods on both the datasets, especially on PubMed.

5.2.4 Predictive Performance

We evaluate the predictive performance of ESVI-LDA comparing against distributed VI on Enron and NY Times datasets on multiple cores. As shown in Figure 6, ESVI typically reaches comparable test perplexity scores as VI but in much shorter wallclock time.

5.3 Handling large # of topics in ESVI-LDA

In VI for LDA, the linear dependence of the model size on K prevents scaling to large K due to memory limitations. Our **ESVI-LDA-TOPK** approach addresses this: instead of storing all K components of the assignment parameter, we only store the most im-

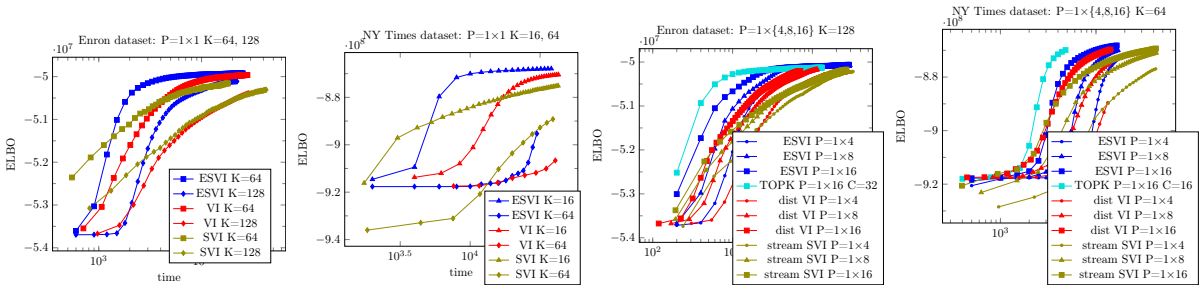


Figure 4: Single Machine experiments for ESVI-LDA (Single and Multi core). TOPK refers to our ESVI-TOPK method. $P = N \times n$ denotes N machines each with n threads.

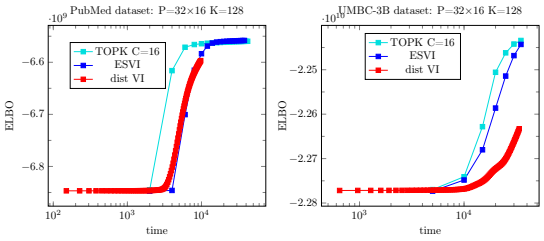


Figure 5: Multi Machine Multi Core experiments for ESVI-LDA. TOPK is our ESVI-TOPK method

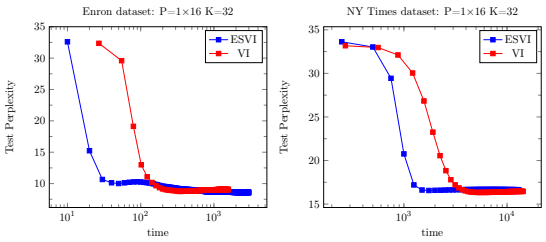


Figure 6: Predictive Performance of ESVI-LDA

portant top-k topics, which we denote using C. Using a min-heap of size C, we maintain only $C \ll K$ topics and we get performance very close to storing all the topics with much lesser memory footprint.

5.3.1 Vary C (cutoff for K), Fix K

While the approximation of ELBO must get more accurate as $C \to K$, there might exist a choice of $C << K$, which gives a good enough approximation. This will give us a significant boost in speed. On Enron dataset, we varied C as $1, 8, 32, 64, 128$ with true $K = 128$ as our baseline. On NY Times dataset, we varied C as $1, 4, 16, 32, 64$ with the true $K = 64$ as baseline. As we expected, setting the cutoff to a value too low leads to very slow convergence. However, it is interesting to note that at a cut off value of roughly $\frac{K}{4}$ (32 on Enron and 16 on NY Times), we get a good result on par with the baseline. On the larger datasets - PubMed and UMBC-3B, setting $C = 16$ was enough to achieve a similar ELBO as the baseline. (See Figure 7).

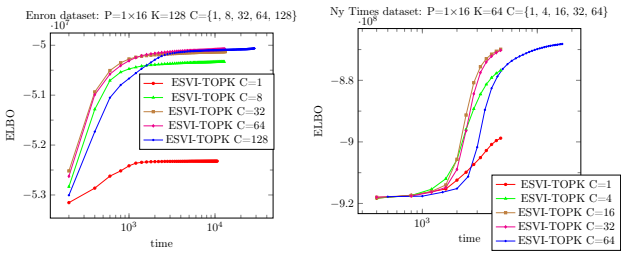


Figure 7: Effect of varying C in ESVI-LDA-TOPK

5.3.2 Fix C (cutoff for K), Scale to large K

By tuning C, ESVI-LDA-TOPK can be run on large number of topics such as $K = 256$ and $K = 512$ on the largest dataset UMBC-3B (See Figure 8).

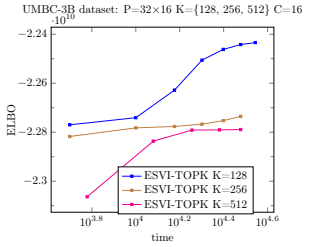


Figure 8: Effect of varying K by fixing C

6 Conclusion

In this paper, we propose Extreme Stochastic Variational Inference (ESVI), a distributed, asynchronous and lock-free algorithm to perform inference for mixture of exponential families. ESVI exhibits simultaneous model and data parallelism, allowing us to handle real-world datasets with large number of documents as well as learn sufficiently large number of parameters. For practitioners, we show how to use ESVI to fit GMM and LDA models on large scale real-world datasets consisting of millions of terms and billions of documents. In our empirical study, ESVI outperforms VI and SVI, and in most cases achieves a better quality solution. ESVI framework is very general and can be extended to several other latent variable models such as Stochastic Block Models.

Acknowledgments: We thank the Texas Advanced Computing Center for infrastructure and support. This research was supported by NSF grants CCF-1564000, IIS-1546452 and IIS-1546459.

References

[1] C. Archambeau and B. Ermis. Incremental variational inference for latent dirichlet allocation. *arXiv preprint arXiv:1507.05016*, 2015.

[2] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Jan. 2003.

[3] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *arXiv preprint arXiv:1601.00670*, 2016.

[4] L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351, 2011.

[5] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan. Streaming variational bayes. In *Advances in Neural Information Processing Systems*, pages 1727–1735, 2013.

[6] L. Hasenclever, S. Webb, T. Lienart, S. Vollmer, B. Lakshminarayanan, C. Blundell, and Y. W. Teh. Distributed bayesian learning with stochastic natural gradient expectation propagation and the posterior server. *Journal of Machine Learning Research*, 18(106):1–37, 2017.

[7] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[8] M. C. Hughes and E. Sudderth. Memoized online variational inference for dirichlet process mixture models. In *Advances in Neural Information Processing Systems*, pages 1133–1141, 2013.

[9] A. R. Masegosa, A. M. Martinez, H. Langseth, T. D. Nielsen, A. Salmerón, D. Ramos-López, and A. L. Madsen. Scaling up bayesian variational inference using distributed computing clusters. *International Journal of Approximate Reasoning*, 88:435–451, 2017.

[10] R. M. Neal and G. 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.

[11] W. Neiswanger, C. Wang, and E. Xing. Embarrassingly parallel variational inference in nonconjugate models. *arXiv preprint arXiv:1510.04163*, 2015.

[12] R. Ranganath, S. Gerrish, and D. M. Blei. Black box variational inference. In *aistats*, 2014.

[13] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In P. Bartlett, F. Pereira, R. Zemel, J. Shawe-Taylor, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701, 2011.

[14] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1 – 2):1–305, 2008.

[15] C. Wang and D. M. Blei. Variational inference in nonconjugate models. *Journal of Machine Learning Research*, 14(Apr):1005–1031, 2013.

[16] J. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694, 2005.

[17] H.-F. Yu, C.-J. Hsieh, H. Yun, S. Vishwanathan, and I. 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.

[18] H.-F. Yu, C.-J. Hsieh, H. Yun, S. Vishwanathan, and I. S. Dhillon. A scalable asynchronous distributed algorithm for topic modeling. In *WWW*, 2015.

[19] H. Yun, H.-F. Yu, C.-J. Hsieh, S. Vishwanathan, and I. Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *Proceedings of the VLDB Endowment*, 7(11):975–986, 2014.

A Proof of Theorem 1

Proof We prove that $\tilde{z}_i^*$ is a stationary point by checking the KKT conditions for (16). Let $h\left(\tilde{z}_i\right) = \left(\sum_{k\in\mathcal{K}} \tilde{z}_{i,k}\right) - C$ and $g_k\left(\tilde{z}_i\right) = -z_{i,k}$. It is clear that $\tilde{z}_i^*$ satisfies the primal feasibility. Now consider KKT multipliers:

$$\lambda = \log \frac{C}{\sum_{k'\in\mathcal{K}} \exp\left(u_{i,k'}\right)}, \text{ and } \mu_k = 0.$$

We have

$$\begin{aligned}
\nabla_k \mathcal{L}_{\mathcal{K}}\left(\tilde{z}_i^*\right) &= u_{i,k} - \log(\tilde{z}_{i,k}^*) - 1 \\
&= u_{i,k} - \left(u_{i,k} + \log \frac{C}{\sum_{k'\in\mathcal{K}} \exp(u_{i,k'})}\right) \\
&= \log \frac{C}{\sum_{k'\in\mathcal{K}} \exp(u_{i,k'})} \\
\lambda \nabla_k h(\tilde{z}_i^*) &= \log \frac{C}{\sum_{k'\in\mathcal{K}} \exp(u_{i,k'})} \\
\mu_k \nabla_k g_{k'}(\tilde{z}_i^*) &= 0.
\end{aligned}$$

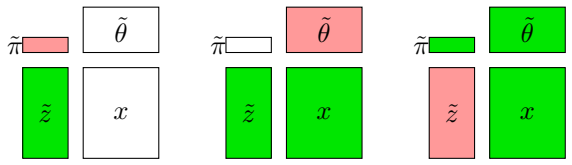
Then it is easy to verify that $\nabla_k \mathcal{L}_{\mathcal{K}}(\tilde{z}_i^*) = \lambda = \lambda \nabla_k h(\tilde{z}_i^*)$. Thus, $\tilde{z}_i^*$ satisfies the stationarity condition:

$$\nabla \mathcal{L}_{\mathcal{K}}(\tilde{z}_i^*) = \lambda \nabla h(\tilde{z}_i^*) + \sum_{k=1}^{K} \mu_k \nabla g_k(\tilde{z}_i^*).$$

Due to choice of $\mu_k = 0$, complementary slackness and dual feasibility are also satisfied. Thus, $\tilde{z}_i^*$ is the optimal solution to (16). ■

B Access Patterns

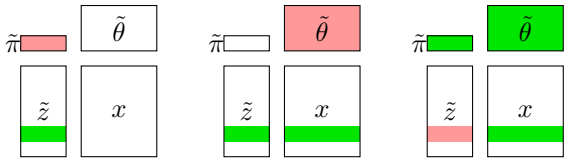
In this section, we outline the access patterns of VI and SVI in Figure 9 and Figure 10 respectively.



(a) $\tilde{\pi}$ update (b) $\tilde{\theta}$ update (c) $\tilde{z}$ update

Figure 9: Access pattern of variables during Variational Inference (VI) updates. Green indicates that the variable or data point is being read, while red indicates that the variable is being updated.

Bottleneck to Model Parallelism: The local variable $\tilde{z}_i$ needs to be normalized in order to be maintained on the k-dimensional simplex Δ_k after the update (12).



(a) $\tilde{\pi}$ update (b) $\tilde{\theta}$ update (c) $\tilde{z}$ update

Figure 10: Access pattern of variables during Stochastic Variational Inference (SVI) updates. Green indicates that the variable or data point is being read, while red indicates that the variable is being updated.

This is the primary bottleneck to model parallelism in both VI and SVI, since this requires access to all K components. In ESVI, we propose a novel way to overcome this barrier, leading to completely independent local and global variable updates.

C ESVI-LDA

In this section, we show how to apply ESVI to Latent Dirichlet Allocation (LDA). Recall the standard LDA model by Blei et al.[2]. Each topic $\beta_k, k \in [K]$ is a distribution over the vocabulary with size V and each document is a combination of K topics. The generative process is:

- Draw topic weights $\beta_k \sim \text{Dirichlet}(\eta)$, $k = 1 \ldots K$
- For every document $d_i \in \{d_1, d_2 \ldots d_D\}$:
  - Draw $\theta_i \sim \text{Dirichlet}(\alpha)$
  - For each word $n \in [N]$:
    - Draw topic assignment $z_{in} \sim \text{Multi}(\theta_i)$
    - Draw word $w_{in} \sim \text{Multi}(\beta_{z_{in}})$

where $\alpha \in \mathbb{R}^K$ and $\eta \in \mathbb{R}^V$ are symmetric Dirichlet priors. The inference task for LDA is to characterize the posterior distribution $p(\beta, \theta, z|w)$. While the posterior is intractable to compute, many methods have been developed to approximate the posterior. Here we use the idea in previous sections to develop extreme stochastic variational inference for LDA.

We denote the assignment of word n in document d_i as z_{in} where $z_i \in \mathbb{R}^K$. Also w_{in} denotes the n-th word in i-th document. Thus in LDA, the local hidden variables for a word is the word assignment vector z_{in} and local hidden variable for a document is z_i and the topic mixture θ_i. The global hidden variable are the topics β_k. Given these, we can formulate the complete

conditional of the topics β_k θ_i and z_{in} as:

$$p(\beta_k|z,w) = \text{Dirichlet}(\eta + \sum_{i=1}^{D}\sum_{n=1}^{N} z_{in}^k w_{in})$$

$$p(\theta_i|z_i) = \text{Dirichlet}(\alpha + \sum_{n=1}^{N} z_{in})$$

$$p(z_{in}^k = 1|\theta_i, \beta_{1:K}, w_{in}) \propto \exp\left(\log\theta_{ik} + \log\beta_k^{w_{in}}\right)$$

We denote multinomial parameter for z_{in}^k as ϕ_{in}^k, Dirichlet parameter for β_k and θ_i as λ_k and γ_i. The update rules for these three variational parameters are:

$$\gamma_i = \alpha + \sum_{n=1}^{N} z_{in}$$

$$\lambda_k = \eta + \sum_{i=1}^{D}\sum_{n=1}^{N} z_{in}^k w_{in},$$

$$\phi_{in}^k \propto \exp\left(\Psi(\gamma_i^k) + \Psi(\lambda_k^{w_{in}}) - \Psi(\sum_{v=1}^{V}\lambda_k^v)\right)$$

where Ψ is the digamma function and we denote $\pi_k = \sum_{v=1}^{V}\lambda_k^v$. Traditional VI algorithms infer all the local variables θ, z and then update the global variable β. This is very inefficient and not scalable. Notice that when updating ϕ_{in}^k we only need to access γ_i^k, $\lambda_k^{w_{in}}$ and π_k. And similarly, once ϕ_{in}^k is modified, the parameters that need to be updated are γ_i^k, $\lambda_k^{w_{in}}$ and π_k. Therefore, as long as π_k can be accessed, the updates to these parameters can be parallelized. Based on the ideas we introduced in Section 4, we propose an asynchronous distributed method ESVI-LDA, which is outlined in Algorithm 5. Besides working threads, each machine also has a sender thread and a receiver thread, which enables the non-locking send/recv of parameters. One key issue here is how to keep $\pi_{1:K}$ up-to-date across multiple processors. For this, we follow [18], who present a scheme for keeping a slowly changing K dimensional vector, approximately synchronized across multiple machines. Succinctly, the idea is to communicate the changes in π using a round robin fashion. Since π does not change rapidly, one can tolerate some staleness without adversely affecting convergence.

In order to update ϕ_{in}^k we need only to access γ_i^k $\lambda_k^{w_{in}}$ and π_k. And similarly, once ϕ_{in}^k is modified, only parameters γ_i^k, $\lambda_k^{w_{in}}$ and π_k need to be updated. Following that, for each word token, these parameters can be updated independently. In our setting, each machine loads its own chunk of the data, and also has local model parameters γ and ϕ. Each machine maintains a local job queue that stores global parameters λ that is now owned by this machine. After updating with each $\lambda_{1:K}^v$, the machine sends it to another machine while pushing v into the job queue of that machine. This leads to a fully asynchronous and non-locking distributed algorithm.

Algorithm 5 ESVI-LDA Algorithm

```
Load {d_1 ... d_D} into P machines
Initialize φ, γ, λ using priors α, η
Initialize job queue Q: distribute λ^{1:V} in P machines
Initialize sender queue q_s
for every machine asynchronously do
  if receiver thread then
    while receive λ^v do
      push (Q_t, λ^v) for some t
    end while
  end if
  if sender thread then
    while not q_s.empty () do
      send q_s.pop() to next random machine
    end while
  end if
  if worker thread t then
    pop from Q_t: λ^v,
    for all local word token s.t. w_dn = v do
      for k = 1 ... K do
        φ^k_dn ∝ exp(ψ(γ^k_d) + ψ(λ^{w_dn}_k) − ψ(Σ_v λ^v_k))
      end for
      for k = 1 ... K do
        γ^k_d += φ^k_dn − φ^k_dn(old)
        λ^{w_dn}_k += φ^k_dn − φ^k_dn(old)
      end for
      Update global Σ_v λ^v_k
    end for
    q_s.push (λ^v)
  end if
end for
```

D ESVI-GMM

Since the distribution of computation in ESVI-GMM method also works in a similar manner as ESVI-LDA Algorithm 5 (only the local and global updates need to be replaced), in this section, we only present the update rules for the local and global variational parameters for Gaussian Mixture Models (GMM).

D.1 VI updates for GMM

The generative process for this model assumes that data $x = (x_1, \ldots, x_N)$ is generated by a mixture of K gaussian distributions whose mean and precision are given by $\mu = \{\mu_k\}$ and $\Lambda = \{\Lambda_k\}$. $\pi \in \Delta_k$ denotes the mixing coefficient, where Δ_k is defined to be the K-dimensional simplex. These are the *global variables*. As usual, $z = (z_1, \ldots, z_N), z_i \in \Delta_k$ denotes the latent

variable to keep track of the component assignments to the data points. These are the *local variables*.

The conditional distributions for the data x and z (likelihood) can be written as:

$$p\left(x|z,\mu,\Lambda\right)=\prod_{i=1}^{N}\prod_{k=1}^{K}\mathcal{N}\left(x_i|\mu_k,\Lambda_k^{-1}\right)^{z_{ik}}$$
$$p\left(z|\pi\right)=\prod_{i=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{ik}}$$

We now introduce the following conjugate priors to simplify the bayesian inference.

$$\begin{aligned}p\left(\pi\right)&=\text{Dirichlet}\left(\pi|\alpha_0\right)\\ p\left(\mu,\Lambda\right)&=p\left(\mu|\Lambda\right)\cdot p\left(\Lambda\right)\\ &=\prod_{k=1}^{K}\underbrace{\mathcal{N}\left(\mu_k|m_0,\left(\beta_0\Lambda_k\right)^{-1}\mathcal{W}\left(\Lambda_k|W_0,\nu_0\right)\right)}_{\text{Gaussian-Wishart}}\end{aligned}$$

where, $m_0,\alpha_0,\beta_0,\nu_0,W_0$ are hyper-parameters that can be initialized to some suitable value.

Given this setup, we can express the joint distribution of all our random variables as:

$$p\left(x,z,\pi,\mu,\Lambda\right)=\underbrace{p\left(x|z,\mu,\Lambda\right)\cdot p\left(\mu,\Lambda\right)}_{\text{conjugate pair}}\cdot\underbrace{p\left(z|\pi\right)\cdot p\left(\pi\right)}_{\text{conjugate pair}}$$

Clearly, the corresponding posterior distribution $p\left(z,\pi,\mu,\Lambda|x\right)$ involves computing expensive high-dimensional integrals and therefore a simpler variational distribution q is used as an approximation:

$$\begin{aligned}q\left(z,\pi,\mu,\Lambda\right)&=q\left(z\right)\cdot q\left(\pi\right)\cdot\prod_{k=1}^{K}q\left(\mu_k,\Lambda_k\right)\\ &=q\left(z\right)\cdot\underbrace{q\left(\pi|\alpha\right)}_{\text{Dirichlet}}\cdot\prod_{k=1}^{K}\underbrace{q\left(\mu_k|m_k,\left(\beta_k\Lambda_k\right)^{-1}\right)}_{\text{Gaussian}}\\ &\cdot\underbrace{q\left(\Lambda_k|W_k,\nu_k\right)}_{\text{Wishart}}\end{aligned}$$

Optimizing the ELBO, leads to the following local and global variable updates.

Update rules for local variables:

$$\rho_{i,k}=\exp\left(\underbrace{\mathbb{E}\left[\log\pi_k\right]}_{t_1}+\frac{1}{2}\underbrace{\mathbb{E}\left[\log\left|\Lambda_k\right|\right]}_{t_2}-\frac{D}{2}\log 2\pi-\frac{1}{2}\underbrace{\mathbb{E}_{\mu_k,\Lambda_k}\left[\left(x_i-\mu_k\right)^\top\Lambda_k\left(x_i-\mu_k\right)\right]}_{t_3}\right)$$

where, the terms t_1, t_2 and t_3 are given by:

$$\begin{aligned}t_1&=\psi\left(\alpha_k\right)-\psi\left(\sum_{k=1}^{K}\alpha_k\right)\\ t_2&=\sum_{j=1}^{D}\psi\left(\frac{\nu_k+1-j}{2}\right)+D\log 2+\log\left|W_k\right|\\ t_3&=D{\beta_k}^{-1}+\nu_k\left(x_i-\mu_k\right)^\top W_k\left(x_i-\mu_k\right)\end{aligned}$$

Using these, the local updates can be written as:

$$\tilde{z}_{i,k}=\frac{\rho_{i,k}}{\sum_{k'=1}^{K}\rho_{i,k'}}$$

Update rules for global variables:
For these, first we define some intermediate quantities that are used in the global updates.

$$\begin{aligned}N_k&=\sum_{i=1}^{N}\tilde{z}_{i,k}\\ \bar{x}_k&=\frac{1}{N_k}\sum_{i=1}^{N}\tilde{z}_{i,k}\cdot x_i\\ S_k&=\frac{1}{N_k}\sum_{i=1}^{N}\tilde{z}_{i,k}\left(x_i-\bar{x}_k\right)\left(x_i-\bar{x}_k\right)^\top\end{aligned}$$

Using these, the global updates can be written as:

$$\begin{aligned}q\left(\Lambda_k\right)&\sim\mathcal{W}\left(\Lambda_k|W_k,\nu_k\right)\\ q\left(\mu_k|\Lambda_k\right)&\sim\mathcal{N}\left(\mu_k|\left(\beta_k\Lambda_k\right)^{-1}\right)\\ q\left(\pi|\alpha\right)&\sim\text{Dirichlet}\left(\pi|\alpha\right)\end{aligned}$$

where the above parameters are given by:

$$\begin{aligned}\beta_k&=\beta_0+N_k\\ m_k&=\frac{1}{\beta_k}\left(\beta_0\cdot m_0+N_k\cdot\bar{x}_k\right)\\ W_k^{-1}&={W_0}^{-1}+N_k\cdot S_k+\frac{\beta_0 N_k}{\beta_0+N_k}\left(\bar{x}_k-m_0\right)\left(\bar{x}_k\right)-m_0^\top\\ \nu_k&=\nu_0+N_k\end{aligned}$$

D.2 Scaling to large dimensions

When the dimensions D are large, GMM becomes computationally heavy since it involves the storage and inversion of a large $O(D\times D)$ matrix. To overcome this problem, we make the assumption of diagonal covariance matrix $\Sigma_k=diag\left(\sigma_1^2,\ldots,\sigma_D^2\right)$ for each component k, which intuitively means that the dimensions are independent within each mixture component. This lets us run ESVI-GMM on larger datasets.

E Parameter Settings used in the empirical study

We used the following hyper-parameter settings:

- In Gaussian Mixture Models (GMM) experiments, we used $\alpha_0 = 5$, $\beta_0 = 1$, $m_0 = 0$. ν_0 and W_0 were turned per dataset based on the best performance. We set $\{\nu_0, W_0\}$ as $\{300000, 0.1\}$ for TOY, $\{300000, 0.1\}$ for AP-DATA, $\{500000, 0.5\}$ for NIPS and $\{500000, 0.5\}$ for NYTIMES datasets.
- For the SVI methods, we used a batch size of 100 (we tried various batch sizes and picked the value we found to provide the best results). The step-size in SVI was decayed following the recommendation in [7], namely, $\eta_t = \frac{\eta_0}{(1+t)}$, where η_0 was carefully tuned and set to 0.1. Here, t denotes the iteration index.