Extreme Stochastic Variational Inference: Distributed and Asynchronous

Jiong Zhang ¹ 1  Parameswaran Raman ¹ ²  Shihao Ji ³  Hsiang-Fu Yu ⁴ ⁵  S.V.N. Vishwanathan ⁶  Inderjit S. Dhillon ⁴

Abstract

We propose extreme stochastic variational inference (ESVI), an asynchronous and lock-free algorithm to perform variational inference on massive real world datasets. Stochastic variational inference (SVI), the state-of-the-art algorithm for scaling variational inference to large-datasets, is inherently serial. Moreover, it requires the parameters to fit in the memory of a single processor; this is problematic when the number of parameters is in billions. ESVI overcomes these limitations 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. We demonstrate the effectiveness of ESVI by running Latent Dirichlet Allocation (LDA) on UMBC-3B, a dataset that has a 3 million words in the vocabulary and 3 billion tokens. To best of our knowledge, this is an order of magnitude larger than the largest dataset on which results using variational inference have been reported in literature. In our experiments, we found that ESVI outperforms VI and SVI, and also achieves a better quality solution. In addition, we propose a strategy to speed up computation and save memory when fitting large number of topics.

1. Introduction

In recent years, variational inference (VI) has emerged as a powerful technique for parameter estimation in a wide variety of Bayesian models (Wainwright and Jordan, 2008), (Blei et al., 2016). 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.

In this paper we will primarily focus on mixture models, a large and important class of models in machine learning. 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. Consequently, application of Bayesian methods was limited to small and medium sized datasets.

In the literature, there are two main approaches to tackle the above problem. The first is to use a divide and conquer strategy to distribute the computation, and the second is to exploit the underlying structure of the optimization problem to reduce the number of iterations (and therefore the corresponding data access). For instance, one can divide the data across multiple machines and use a map-reduce based framework to aggregate the computations. See, for instance, Neiswanger et al. (2015) for an example of this approach. An instance of the algorithmic approach is the paper by Hoffman et al. (2013). 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, the Stochastic Variational Inference (SVI) algorithm of Hoffman et al. (2013) interleaves global variable updates with local variable updates. The intuition here is that even before one pass through the dataset, the global variables have been updated multiple times, and therefore the model parameters converge rapidly towards their final values. The argument is similar in spirit to how stochastic optimization outperforms batch algorithms for maximum aposteriori (MAP) estimation (Bottou and Bousquet, 2011). Consequently, SVI enabled applying variational inference to datasets consisting of millions of documents such as Nature and NewYork Times (Hoffman et al., 2013) which could not be handled before.

| what fits in memory | data, parameters | parameters | data | none |
|---------------------|-----------------|------------|------|------|
| Distributed-VI      | ✓               |            |      |      |
| SVI                 | ✓               | ✓          |      |      |
| ESVI                | ✓               | ✓          | ✓    | ✓    |

Table 1. Applicability of the three algorithms to common scenarios in distributed machine learning

With the advent of the big-data era, now we routinely deal with industry-scale problems which involve billions of doc-
ments and tokens. Such massive datasets pose another challenge, which, unfortunately SVI is unable to address; the set of parameters is so large that it does not fit on a single processor\(^1\). For instance, if we have \(K\) mixture components and \(D\) is the dimension of the features, then the parameter size is \(D \times K\). If \(D\) is of the order of millions and \(K\) is in the 100’s or 1000’s (see our experiments in Section 6), the parameter size is a few 100s of GB.

In this paper we propose a new framework, Extreme Stochastic Variational Inference (ESVI), in order to scale VI to datasets which are an order of magnitude larger than those that can be handled by SVI. An attractive property of ESVI is that it is both data and model parallel. What this means is that, unlike SVI, one can distribute the computation across multiple processors and each processor only needs to store and update a fraction of the parameters. Of course, the algorithm is stochastic just like SVI. However, unlike existing data parallel approaches which require bulk synchronization after every iteration, the updates of ESVI are asynchronous and lock-free. To the best of our knowledge there is not existing algorithm for VI that sports these desirable properties.

Table 1 outlines the various scenarios in scaling to large datasets and the applicability of the three algorithms. We categorize them based on whether the data and model parameters fit in memory on a single machine or not. We stress upon the fact that although in principle ESVI is applicable to all the regimes mentioned, its advantages are truly realized in the case where both data and parameters do not fit in memory on a single machine.

In a nutshell, our idea is as follows: instead of updating all the \(K\) coordinates of a local variable and then moving onto updating a global variable, we will only update a small subset of these coordinates and update the global variables. This seemingly simple idea has some powerful consequences. It allows multiple processors to simultaneously perform parameter updates. Moreover, using a classic owner-computes paradigm, one can make the resulting algorithm asynchronous and lock-free. In our experiments, we show that our new algorithm outperforms a straightforward strategy for parallelizing variational inference, which requires bulk synchronization after every iteration.

The rest of the paper is structured as follows: we briefly review variational and stochastic variational inference in Section 2. We present our new algorithm in Section 3, and discuss its advantages. Next in Section 4, we show how to apply ESVI to Latent Dirichlet Allocation (ESVI-LDA) which is a specific instantiation of the mixture of exponential family models. Related work is discussed in Section

\(^1\)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.

5. Empirical evaluation is presented in Section 6, and the paper concludes with a discussion in Section 7.

2. Background

In this paper we focus on parameter estimation for the mixture of exponential families model which generalizes a wide collection of well-known latent variable models such as latent dirichlet allocation (LDA), gaussian mixture models (GMM), matrix factorization, mixed membership stochastic block models (MMB) and dirichlet process mixtures. Hence, our ESVI algorithm is fairly generic and can be adapted to any of these models once their specific updates have been derived.

2.1. Notation

Given \(N\) observations \(x = \{x_1, \ldots, x_N\}\), with each \(x_i \in \mathbb{R}^D\), we wish to model \(p(x)\) as a mixture of \(K\) distributions from the exponential family. Towards this end, let \(z = \{z_1, \ldots, z_N\}\), with each \(z_i \in \{1, \ldots, K\}\), denote local latent variables; intuitively, \(z_i\)'s denote which component the current data point was drawn from. Moreover, let \(\theta = \{\theta_1, \ldots, \theta_K\}\) denote global latent variables; each \(\theta_k\) represents the sufficient statistics of an exponential family distribution. Finally, we denote the \(K\)-dimensional simplex by \(\Delta_K\), and let \(\pi \in \Delta_K\) be the mixing coefficients of the mixture model. Note that \(\pi\) is also a global latent variable.

Although we use a general description in this section, we refer the readers to keep GMM in mind as a running example. In GMM, the parameters \(\theta_{1:k}\) represent the mean and covariance of the \(K\) gaussians, and \(\pi\) represents their mixing proportions. \(z_{1:N}\) is the soft-assignment of a particular data point to one of the \(K\) components.

2.2. Mixture Model

The following data generation scheme underlies a mixture of exponential family model:

\[
p(\pi|\alpha) = \text{Dirichlet} (\alpha) \quad (1)
\]

For \(k = 1, \ldots, K\)

\[
p(\theta_k|n_k, \nu_k) = \exp \left( \langle n_k \cdot \nu_k, \theta_k \rangle - n_k \cdot g(\theta_k) - h(n_k, \nu_k) \right) \quad (2)
\]

for \(i = 1, \ldots, N\)

\[
p(z_i|\pi) = \text{Multinomial} (\pi) \quad (3)
\]

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

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 \(p(z_i|\pi)\). The joint distribu-
tion of the data and latent variables can be written as
\[ p(x, \pi, z, \theta | \alpha, n, \nu) = p(\pi | \alpha) \cdot \prod_{k=1}^{K} p(\theta_k | n_k, \nu_k) \cdot \prod_{i=1}^{N} p(z_i | \pi) \cdot p(x_i | z_i, \theta). \] (5)

### 2.3. Variational Inference

The goal of inference is to estimate \( p(\pi, z, \theta | x, \alpha, n, \nu) \). However, computing this distribution requires marginalization over \( x \), which is typically intractable. Therefore, variational inference (Blei et al., 2016) approximates this distribution with a fully, factorized distribution of the following form:
\[ q(\pi, z, \theta | \tilde{\pi}, \tilde{z}, \tilde{\theta}) = q(\pi | \tilde{\pi}) \cdot \prod_{i=1}^{N} q(z_i | \tilde{z}_i) \cdot \prod_{k=1}^{K} q(\theta_k | \tilde{\theta}_k). \] (6)

A \( \sim \) over a symbol is used to denote that it is a parameter of the variational distribution. 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 the full conditionals of their counterparts in (5). The parameters of the variational inference are estimated by maximizing the following evidence based lower-bound (ELBO) (Blei et al., 2016):
\[ \mathcal{L}(\tilde{\pi}, \tilde{z}, \tilde{\theta}) = \mathbb{E}_{q(\pi, z, \theta | \tilde{\pi}, \tilde{z}, \tilde{\theta})} \left[ \log p(x, \pi, z, \theta | \alpha, n, \nu) \right] - \mathbb{E}_{q(\pi, z, \theta | \tilde{\pi}, \tilde{z}, \tilde{\theta})} \left[ \log q(\pi, z, \theta | \tilde{\pi}, \tilde{z}, \tilde{\theta}) \right]. \] (7)

Variational inference algorithms perform coordinate descent updates on \( \mathcal{L} \) by optimizing each set of variables, one at a time.

**Update for \( \tilde{\pi} \)**
\[ \tilde{\pi}_k = \alpha + \sum_{i=1}^{N} \tilde{z}_{i,k} \] (8)

**Update for \( \tilde{\theta}_k \)** The components of \( \tilde{\theta}_k \) namely \( \tilde{n}_k \) and \( \tilde{\nu}_k \) are updated as follows:
\[ \tilde{n}_k = n_k + N_k \] (9)
\[ \tilde{\nu}_k = n_k \cdot \nu_k + N_k \cdot \bar{x}_k \] (10)

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

**Update for \( \tilde{z}_i \)** Let \( u_i \) be a \( K \) dimensional vector whose \( k \)-th component is given by
\[ u_{i,k} = \psi(\tilde{\pi}_k) - \psi \left( \sum_{k'=1}^{K} \tilde{\pi}_{k'} \right) + \left( \phi(x_i, k), \mathbb{E}_{q(\theta_{i,k} | \tilde{\theta}_k)} [\theta_k] \right) - \mathbb{E}_{q(\theta_{i,k} | \tilde{\theta}_k)} [\phi(\theta_k)] \] (11)
\[ \tilde{z}_{i,k} = \frac{\exp(u_{i,k})}{\sum_{k'=1}^{K} \exp(u_{i,k'})} \] (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). We discuss in the next section, how this benefits our formulation. See e.g., Blei et al. (2016) for details on how these updates are derived. The corresponding update equations for GMM can be found in (Bishop, 2006).

Therefore, the variational inference algorithm (Wainwright and Jordan, 2008; Blei et al., 2016) iteratively performs the following sequence of updates until convergence:

1. For \( i = 1, \ldots, N \) update \( \tilde{z}_i \) using (12)
2. Update \( \tilde{\pi} \) using (8)
3. For \( k = 1, \ldots, K \) update \( \tilde{\theta}_k \) using (9) and (10).

### 2.4. Stochastic Variational Inference

The stochastic variational inference algorithm of Hoffman et al. (2013), on the other hand, performs the following sequence of updates:

- Pick an \( i \in \{1, \ldots, N\} \) uniformly at random
  1. Update \( \tilde{z}_i \) using (12)
  2. Update \( \tilde{\pi} \) using (8)
  3. For \( k = 1, \ldots, K \) update \( \tilde{\theta}_k \) using (9) and (10).

In contrast to variational inference where all the local variables are updated before updating the global variables, here \( \tilde{z}_i \) corresponding to one data point \( x_i \) is updated, and then immediately the global parameters \( \tilde{\pi} \) and \( \tilde{\theta} \) are updated.

### 3. Extreme Stochastic Variational Inference (ESVI)

In this paper we propose the following sequence of updates:

- Sample \( i \in \{1, \ldots, N\} \)
  - Select \( K \subset \{1, \ldots, K\} \)
    1. Update \( \tilde{z}_{i,k} \) for all \( k \in K \) (see below)
    2. Update \( \tilde{\pi}_k \) for all \( k \in K \) using (8)
    3. Update \( \tilde{\theta}_k \) for all \( k \in K \) using (9) and (10).
Before we discuss why this update is advantageous for parallelization, let us first study how one can update a subset of coordinates of $\tilde{z}_i$ efficiently. Towards this end, plugging 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 $\mathbb{E}_{q(\pi | z)} \log p(z_i = k | \pi) = \psi(\tilde{\pi}_k) - \psi\left(\sum_{k' = 1}^{K} \tilde{\pi}_{k'}\right)$, to obtain the following objective function

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

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

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

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

$$
\sum_{k} \tilde{z}_{i,k} = 1, \quad 0 \leq \tilde{z}_{i,k} \leq 1.
$$

The following lemma shows that one can find a closed form solution to maximizing (14) even if we restrict our attention to a subset of variables.

**Lemma 1** For $2 \leq K' \leq K$ let $\mathcal{K} \subset \{1, \ldots, K\}$ be such that $|\mathcal{K}| = K'$. For any $C > 0$, the following problem

$$
\max_{\tilde{z}_i \in \mathbb{R}^{K'}} \mathcal{L}_\mathcal{K} := \sum_{k \in \mathcal{K}} \tilde{z}_{i,k} \cdot u_{i,k} - \tilde{z}_{i,k} \cdot \log \tilde{z}_{i,k}
$$

s.t. $\sum_{k \in \mathcal{K}} \tilde{z}_{i,k} = C$ and $0 \leq \tilde{z}_{i,k}$,

has the closed form solution:

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

**Proof** Proof of this Lemma is available in Appendix A.

The above lemma 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, the ESVI strategy we proposed earlier in this section is a valid coordinate ascent algorithm for improving the ELBO (7).

### 3.1. 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.

The access pattern of variational inference is as follows (see Figure 9 in Appendix for a visual illustration):

- The update for $\tilde{\pi}$ (8) requires access to all $\tilde{z}_i$.
- The update for $\tilde{\theta}$ (9) and (10) requires access to all $\tilde{z}_i$.
- In order to update $\tilde{z}_i$ using (12), we need to first compute $u_{i,k}$ (11) for all $k$. This in turn requires access to $\tilde{\pi}$ and all $\tilde{\theta}_k$.

The straightforward way to distribute computation is to use a master-slave/map-reduce architecture. The data is distributed across the slaves. At the beginning of each iteration, the master collects all the local $\tilde{z}_i$ values from the slaves and updates $\tilde{\theta}$ and $\tilde{\pi}$. These variables are then transmitted to the slaves, which in turn update their local $\tilde{z}_i$ variables. There are two key problems with this scheme: First, we need to bulk-synchronize after every iteration. Second, the parameters need to be replicated on each slave (in other words, this scheme exhibits data-parallelism but not model-parallelism). Moreover, we still suffer from the slow convergence of batch variational inference.

On the other hand, the access pattern of stochastic variational inference is somewhat different (see Figure 10 in Appendix for an illustration). Since only one $\tilde{z}_i$ is updated at every step, it follows that

- The update for $\tilde{\pi}$ (8) requires access to only the $\tilde{z}_i$ that was updated.
- The update for $\tilde{\theta}$ (9) and (10) requires 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 $\tilde{\theta}_k$.

It is not straightforward to parallelize stochastic variational inference. The key bottleneck is the $\tilde{z}_i$ update, which needs access to all the global parameters $\tilde{\pi}$ and $\tilde{\theta}$, and hence cannot be performed in parallel.

In contrast, the access pattern of extreme stochastic variational inference is as follows (recall that $\mathcal{K}$ denotes the subset of coordinates of $\tilde{z}_i$ that were updated):

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

---

**References:**

1. Figure 9 in Appendix for a visual illustration.
2. Figure 10 in Appendix for an illustration.
3. For $2 \leq K' \leq K$ let $\mathcal{K} \subset \{1, \ldots, K\}$ be such that $|\mathcal{K}| = K'$. For any $C > 0$, the following problem

---

**Appendix A:**

Proof of this Lemma is available in Appendix A.

The above lemma 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, the ESVI strategy we proposed earlier in this section is a valid coordinate ascent algorithm for improving the ELBO (7).
• The update for $\tilde{\theta}$ (9) and (10) requires access to $\tilde{z}_{i,k}$ for $k \in K$.
• The update to $\tilde{z}_{i,k}$ for $k \in K$ requires access to $\tilde{\pi}$ and $\tilde{\theta}_k$ for $k \in K$.

See Figure 1 for an illustration. In particular, what this means is that, multiple processors can access and update mutually exclusive subsets of coordinates $K$ independently.

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

Figure 1. Access pattern of variables during Extreme Stochastic Variational Inference (ESVI) updates. Green indicates that the variable or data point is being read, while red indicates that the variable is being updated.

3.2. Parallelization

In this sub-section, we describe the parallel asynchronous algorithm of ESVI. Let $P$ denote the number of processors, and let $I_p \subset \{1, \ldots, N\}$ denote the indices of the data points owned by processor $p$. The corresponding local variables $\tilde{z}_i$ for $i \in I_p$ are also assigned to processor $p$. The global variables are split across the processors. Let $K_{p,t} \subset \{1, \ldots, K\}$ denote the indices of the rows of $\tilde{\theta}$ residing in processor $p$ at time instance $t$. Then processor $p$ can update any $\tilde{z}_{i,k}$ for $i \in I_p$ and $k \in K_{p,t}$.

To complete the description of our scheme, 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 (Yun et al., 2014) and Yu et al. (2015b).

Figure 2 is an illustration of how this works pictorially. The horizontal blocks denote the partitioning of data We partition the data across the processors. As a result the corresponding $\tilde{z}_{1:N}$ variables also get partitioned across processors. Once partitioned, the $\tilde{z}$ variables never move. On the other hand, the $\tilde{\theta}$ variables move nomadically 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 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 1. Each processor maintains its own worker queue and picks a subset of global variables $\tilde{\theta}$ from the queue to work on. Once the updates are performed, this subset of $\tilde{\theta}$ is pushed into the queue of a random neighbor. It can be seen here that the nature of this communication is asynchronous and nonblocking. As long as there are items in the queue, the computations remain fully asynchronous and decentralized. Furthermore, all workers are symmetric, that is, there is no designated master or slave.

4. ESVI-LDA

We now show how to apply ESVI to Latent Dirichlet Allocation (LDA), which is a popular latent variable model used to understand the document structure in large text corpora. Recall the standard LDA model by Blei et al. (Blei et al., 2003). 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 every word $w \in [N]$
    * Draw topic assignment $z_{in} \sim \text{Multi}(\theta_i)$
    * Draw word $w_{in} \sim \text{Multi}(\beta_{z_{in}})$
We denote the assignment of word $LDA$.

tions to develop extreme stochastic variational inference for estimate the posterior. Here we use the idea in previous sec-
to compute, many methods have been developed to approx-
distribution $\alpha$ parameter for $\beta$ and $\gamma$. The update rules for these three variational parameters are:

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 $\theta_i$. The global hidden variable are the topics $\beta_k$. Given these, we can formulate the complete conditional of the topics $\beta_k \theta_i$ and $z_{in}$ as:

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

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

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

We denote multinomial parameter for $z_{in}^k$ as $\phi_{in}^k$, Dirichlet parameter for $\beta_k$ and $\theta_i$ as $\lambda_k$ and $\gamma_i$. The update rules for these three variational parameters are:

\[
\begin{align*}
\lambda_k &= \eta + \sum_{i=1}^{D} \sum_{n=1}^{N} z_{in}^k w_{in} \\
\gamma_i &= \alpha + \sum_{n=1}^{N} z_{in} \\
\phi_{in}^k &\propto \exp\{\Psi(\gamma_i^k) + \Psi(\lambda_k w_{in}) - \Psi(\sum_{v=1}^{V} \lambda_v)\}
\end{align*}
\]

where $\Psi$ is the digamma function and we denote $\pi_k = \sum_{v=1}^{V} \lambda_v^k$. Traditional VI algorithms infer all the local variables $\theta, z$ and then update the global variable $\beta$. This is very inefficient and not scalable. Notice that when updating $\phi_{in}$ we only need to access $\gamma_i^k, \lambda_k w_{in}$ and $\pi_k$. And similarly, once $\phi_{in}$ is modified, the parameters that need to be updated are $\beta_{1:k}, \lambda_{1:k} w_{in}$ and $\pi_k$. Therefore, as long as $\pi_k$ can be accessed, the updates to these parameters can be parallelized. Based on the ideas we introduced in Section 3, we propose an asynchronous distributed method ESVI-LDA, which is outlined in Algorithm 2. 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 Yu et al. (2015b), 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 $\pi$ using a round robin fashion. Since $\pi$ does not change rapidly, one can tolerate some staleness without adversely affecting convergence.

In order to update $\phi_{in}$ we only need to access $\gamma_i^k, \lambda_{k} w_{in}$ and $\pi_k$. And similarly, once $\phi_{in}$ is modified, only parameters $\gamma_i^k, \lambda_{k} w_{in}$ and $\pi_k$ 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 $\gamma$ and $\phi$. Each machine maintains a local job queue that stores global parameters $\lambda$ that is now owned by this machine. After updating with each $\lambda_{k} w_{in}$, 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.

Handling large number of topics: In variational inference LDA, the linear dependence of the model size on $K$ prevents scaling to large $K$ due to memory limitations. We propose a strategy to address this which we call the top-k approach. This basic idea is simple: instead of storing all $K$ components of the assignment parameter, we only store the most important top $k$ topics. Such a top-k can be maintained using a min-heap of size $k$. Based on our empirical study, we find that this strategy indeed works very well. By
Algorithm 2 ESVI-LDA Algorithm

Load \{d_1 \ldots d_D\} into P machines
Initialize \(\phi, \gamma, \lambda\) using priors \(\alpha, \eta\)
Initialize job queue \(Q\): distribute \(\lambda^1 : V\) in \(P\) machines
Initialize sender queue \(q_s\)

for every machine asynchronously do
  if receiver thread then
    while receive \(\lambda^t\) do
      push \((Q_t, \lambda^t)\) for some \(t\)
  if sender thread then
    while not \(q_s.empty()\) do
      send \(q_s.pop()\) to next random machine
  if worker thread \(t\) then
    pop from \(Q_t: \lambda^t\),
    for all local word tokens s.t. \(w_{dn} = v\) do
      for \(k = 1 \ldots K\) do
        \(\phi_{dn}^k \propto \exp \left( \psi(\gamma_d^k) + \psi(\lambda_{w^d}^k) - \psi(\sum_v \lambda_v^k) \right)\)
      for \(k = 1 \ldots K\) do
        \(\gamma_{d}^{k+} = \phi_{dn}^k - \phi_{dn}^k\) (old)
        \(\lambda_{w^d}^{k+} = \phi_{dn}^k - \phi_{dn}^k\) (old)
        Update global \(\sum_v \lambda_v^k\)
        \(q_s.push(\lambda^t)\)

only maintaining \(C \ll K\) top values we get performance very close to storing all the values without exceeding the maximum memory limit.

5. Related Work

Recent research on variational inference has focused on two aspects: 1) Extending variational inference to a wider set of models e.g., mixtures of more complicated models such as HMMs (Foti et al., 2014) or to models which do not satisfy conditional conjugacy. 2) Developing variants that can scale to large datasets. As we already discussed Stochastic Variational Inference (SVI) proposed by Hoffman et al. (2013) was an important step in this direction. Other than the fact that SVI is inherently serial, it also suffers from another drawback. One needs to store the \(D \times K\) matrix \(\theta\) on a single machine. For certain massive datasets, where both \(D\) and \(K\) are large, this can become prohibitively expensive. On the other hand, our method, ESVI, exhibits model parallelism; each processor only needs to store \(1/P\) fraction of \(\theta\).

Ranganath et al. (2014) proposed black-box variational inference, which generalize SVI beyond conditionally conjugate models. They propose 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 black-box variational inference.

Neiswanger et al. (2015) presents a classic Map-Reduce style inference algorithm, 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. Our approach in contrast does not suffer from this problem, because our asynchronous and lock-free updates avoid bulk synchronization altogether.

Yu et al. (2015a) proposed a distributed asynchronous algorithm for parameter estimation in Latent Dirichlet Allocation (Blei et al., 2003), which is a mixture of multinomials. 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.

There also has been some work to apply Variational Inference to the streaming setting where SVI does not work. (Broderick et al., 2013) presents an algorithm that applies VI to the streaming setting by performing asynchronous Bayesian updates to the posterior as batches of data arrive continuously. For this they follow the popular master-slave architecture. This is similar in spirit to Hogwild (Recht et al., 2011).

6. Experiments

In our experiments, we evaluate the effectiveness of our proposed ESVI-GMM and ESVI-LDA methods against the other popular approaches: VI and SVI.

Datasets: We use real-world datasets of varying scale. The specific characteristics are described in Table 2.

|        | # Documents | # Vocabulary | # Words      |
|--------|-------------|--------------|--------------|
| 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

Hardware and Implementation: The experiments were performed on a large-scale parallel computing platform at the anonymous center, where each node consists of 20 Intel Xeon E5-2680 CPUs and 256 GB memory. We implemented ESVI-LDA in C++ using MPICH for inter-machine communication, OpenMP for multi-threading and Intel Threaded Building blocks (TBB) for concurrent queues. For Distributed-VI and SVI implementations, we
used the authors’ original code in C\(^2\) and modified it\(^3\).

6.1. ESVI-GMM

In this section, we first compare ESVI-GMM with SVI and VI in the Single Machine Single thread setting. We use the ALOI dataset which is a color image collection of one-thousand small objects. It consists of \(N = 108,000\) data points, \(D = 128\) dimensions and \(K = 256\) components. We plot the performance of the methods (ELBO) both as a function of the total number of coordinates \(z_{ik}\) updated by each algorithm (this is a proxy for the amount of work done to update the local and global parameters), as well as time. In both the plots we can observe that ESVI-GMM does outperform SVI by a little bit (more so in terms of time). When compared against VI, ESVI-GMM wins by a lot.

For Multi Machine case, we use the NIPS and NY Times datasets and only compare against VI (SVI does not apply as it needs to update all its global parameters and when \(K\) is large, this is not feasible). Although typically these datasets do not demand running on Multiple Machines, we do so to demonstrate the ability to scale to very large number of components \((K = 1024)\) and very high dimensions, which is typically the case with large scale text datasets consisting of millions of word count features. Traditionally, GMM inference methods have not been able to handle such a scale. The results are shown in Figure 3 and it is evident that ESVI-GMM is able to outperform VI by a clear margin.

6.2. ESVI-LDA

In these experiments, we compare our proposed ESVI-LDA and ESVI-LDA-TOPK approaches with VI and SVI.

6.2.1. Single Machine Single thread

We compare serial versions of the methods on Enron and NY Times datasets which are medium sized and can be easily fit in memory on a single-machine. On both datasets, we run with single machine and single thread. For Enron, we set the number 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 the results with \(K = 64\) and 128. Figure 4 shows the result of our experiments. In our experiments, x-axis is in log-scale. ESVI-LDA performs better than VI and SVI in both the datasets for all corresponding values of \(K\). Infact, the ESVI-LDA-TOPK performs despite storing just the top 32 topics out of 128.

6.2.2. Single Machine Multi Core

In this section, we evaluate the performance of distributed ESVI-LDA against a map-reduce based distributed implementation of VI. We present results varying the number of cores as 4, 8, 16. This is shown in Figure 4 (two left-most plots). For all the cores, the number of topics \(K\) is set to a constant value. For Enron dataset, we use \(K = 128\) and for NY Times dataset, we use \(K = 64\). Here again, we observe that ESVI-LDA is able to outperform VI consistently for each setting of the number of cores. In addition, we observe that both the methods benefit reasonably when we provide more cores to the computation. Here again, we observe the same trend that ESVI-LDA-TOPK performs the best by storing only the top 16 topics instead of 64.

6.2.3. Multi Machine Multi Core

In this section, we stretch the limits of our distributed 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. To the best of our knowledge such large scale datasets have not been handled before to perform Variational Inference. For all the experiments in this section, we make use of 32 nodes and 16 cores and fit number of topics \(K = 128\). As the results in Figure 5 (two right-most plots) demonstrate, ESVI-LDA gets to 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 an slight initial advantage, however eventually ESVI progresses much faster towards a better ELBO. ESVI-LDA-TOPK is particularly better than the other two in both the datasets. On PubMed especially, the difference in speed we gain by using the top-k approach is substantial.

6.2.4. Predictive Performance

In order to evaluate the predictive performance of ESVI-LDA, we examine its perplexity on a held-out test set. We compare against distributed VI on Enron and NY Times datasets. This experiment is performed using multiple cores. As shown in Figure 6, ESVI typically reaches comparable perplexity scores as VI but in a much shorter wall-clock time. For instance, on the Enron dataset VI reaches a perplexity score of 9.052773 after a time of 1576.32 secs. ESVI on the other hand, matches this perplexity in just 440 secs, and goes down further to a final perplexity score of 8.58929 in 2991 secs. On NY Times, VI begins at 33.171374 and reaches 16.460541 in 14309 secs. ESVI matches this perplexity in roughly 2000 secs.
7. Conclusion

In this paper, we proposed Extreme Stochastic Variational Inference (ESVI), a distributed, asynchronous and lock-free algorithm to perform large-scale inference for the mixture of exponential families. ESVI exhibits both model as well as data parallelism simultaneously, allowing us to handle real-world datasets with large number of documents as well as learn sufficiently large number of parameters. To demonstrate its practical applicability, 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. Our extensive empirical study is strongly suggestive that ESVI outperforms VI and SVI, and in most cases achieves a better quality solution. We also want to note that even though we focus on the inference for exponential families in this paper, the idea is actually widely applicable. Same framework could be applied to various problems such as k-means clustering and Forward-backward algorithm. Thus ESVI framework promises several future directions of work. An obvious next line of work is to explore other latent variable models such as Stochastic Block Models and Bayesian Matrix Factorization.
References

Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

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

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *arXiv preprint arXiv:1601.00670*, 2016.

Léon Bottou and Olivier Bousquet. The tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351, 2011.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. In *Advances in Neural Information Processing Systems*, pages 1727–1735, 2013.

Nicholas Foti, Jason Xu, Dillon Laird, and Emily Fox. Stochastic variational inference for hidden markov models. In *Advances in Neural Information Processing Systems*, pages 3599–3607, 2014.

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

Willie Neiswanger, Chong Wang, and Eric Xing. Emarrassingly parallel variational inference in nonconjugate models. *arXiv preprint arXiv:1510.04163*, 2015.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *aistats*, 2014.

B. Recht, C. Re, S.J. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Peter Bartlett, Fernando Pereira, Richard Zemel, John Shawe-Taylor, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701, 2011. URL http://books.nips.cc/nips24.html.

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.

Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, SVN Vishwanathan, and Inderjit S Dhillon. A scalable asynchronous distributed algorithm for topic modeling. In *WWW*, 2015b.

Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, SVN Vishwanathan, and Inderjit 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 Lemma 1

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

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

We have

$$\nabla_k L_K(\hat{z}_i^*) = u_{i,k} - \log(\hat{z}_{i,k}^*) - 1 = u_{i,k} - \left( u_{i,k} + \log \left( \frac{C}{\sum_{k' \in K} \exp(u_{i,k'})} \right) \right) = \log \frac{C}{\sum_{k' \in K} \exp(u_{i,k'})} \lambda \nabla_k h(\hat{z}_i^*) = \lambda \nabla_k h(\hat{z}_i^*) = \mu_k \nabla_k g_k(\hat{z}_i^*) = 0.$$

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

$$\nabla L_K(\hat{z}_i^*) = \lambda \nabla h(\hat{z}_i^*) + \sum_{k=1}^{K} \mu_k \nabla g_k(\hat{z}_i^*).$$

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

B. Effect of using the top-$k$ strategy

B.0.1. Effect of varying $C$ (cutoff for $K$) in ESVI-LDA-TOPK

In this sub-section, we investigate the effect of varying the value of cutoff $C$ in the ESVI-LDA-TOPK method. While the approximation of evidence lower bound (ELBO) must get more accurate as $C \to K$, there might exist a choice of $C \ll K$, which still gives a reasonably good enough approximation. This will give us a significant boost in speed. Figure 7 shows the result of our experiment. On Enron dataset, we varied $C$ as 1, 8, 32, 64, 128 with the 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 $K = 32$ on Enron and 16 on NY Times, we get a good result on par with using the baseline. On the larger datasets, PubMed and UMBC-3B also, we observed a similar behavior where setting $C = 16$ was enough to achieve a similar ELBO as the baseline. Figure 7 shows the results of this experiment.

B.0.2. Fixing $C$ (cutoff for $K$) and Scaling to Large $K$ in ESVI-LDA-TOPK

In this sub-section, we present results showing how, once we have picked a suitable cutoff $C$ for the our ESVI-LDA-TOPK method, we can scale our algorithm to very large number of topics such as $K = 256$ and $K = 512$ on the largest dataset: UMBC-3B. Figure 8 below demonstrates this.

C. Access Patterns

In this section, we outline the access patterns of VI and SVI.
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.

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.