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.

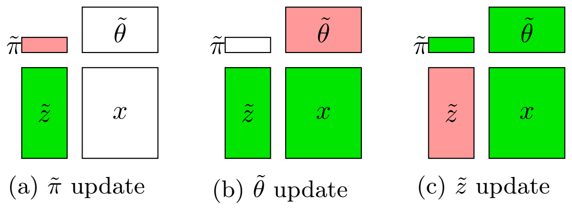

Figure 9: Access pattern of variables during Variational Inference (VI) updates. Green indicates that the variable or data point is being read, while red indicates that the variable is being updated.

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).

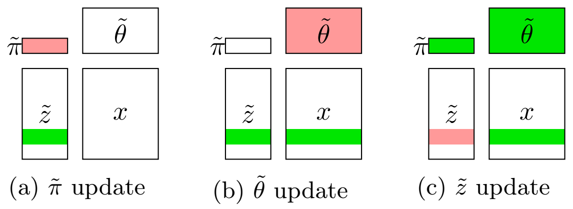

Figure 10: Access pattern of variables during Stochastic Variational Inference (SVI) updates. Green indicates that the variable or data point is being read, while red indicates that the variable is being updated.

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 $\{d_1 \dots d_D\}$ 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 $\lambda^v$ do
      $push\left(Q_t, \lambda^v\right)$ for some $t$
    end while
  end if
  if sender thread then
    while not $q_s.empty\left(\right)$ do
      send $q_s.pop()$ to next random machine
    end while
  end if
  if worker thread $t$ then
    pop from $Q_t$: $\lambda^v$,
    for all local word token s.t. $w_{dn} = v$ do
      for $k = 1 \dots K$ do
        $\phi_{dn}^k \propto \exp\left(\psi\left(\gamma_d^k\right) + \psi\left(\lambda_k^{w_{dn}}\right) - \psi\left(\sum_v \lambda_k^v\right)\right)$
      end for
      for $k = 1 \dots K$ do
        $\gamma_d^k += \phi_{dn}^k - \phi_{dn}^k(old)$
        $\lambda_k^{w_{dn}} += \phi_{dn}^k - \phi_{dn}^k(old)$
      end for
      Update global $\sum_v \lambda_k^v$
    end for
    $q_s.push\left(\lambda^v\right)$
  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 $x = (x_1, \dots, x_N)$ 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, $z = (z_1, \dots, z_N), z_i \in \Delta_k$ 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:

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

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

Update rules for local variables:

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

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:

$$\begin{aligned}\beta_k&=\beta_0+N_k\\ m_k&=\frac{1}{\beta_k}\left(\beta_0\cdot m_0+N_k\cdot\bar{x}_k\right)\\ W_k^{-1}&={W_0}^{-1}+N_k\cdot S_k+\frac{\beta_0N_k}{\beta_0+N_k}\left(\bar{x}_k-m_0\right)\left(\bar{x}_k\right)-m_0^{\top}\\ \nu_k&=\nu_0+N_k\end{aligned}$$

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.