Distributed Algorithms for Topic Models
and
Distributed Stochastic Gradient MCMC

+David Newman, Arthur Asuncion, Padhraic Smyth, Max Welling

*Sungjin Ahn, Babk Shahbaba, Max Welling

Presented by Changyou Chen
March 6, 2015
Outline

1. Distributed Algorithms for Topic Models
   - Distributed LDA and HDP
   - Experiments

2. Distributed Stochastic Gradient MCMC
   - The Algorithm
   - Experiments
Distributed Algorithms for Topic Models

LDA and HDP

\(\phi_k\): topic-word distributions; \(\theta_j\): topic proportion; \(X_{ij}\): words

LDA (left):

\[\begin{align*}
\phi_k &\sim \text{Dir}(\beta) \\
\theta_j &\sim \text{Dir}(\alpha) \\
Z_{ij} &\sim \text{Cat}(\theta_j) \\
X_{ij} &\sim \text{Cat}(\phi_{Z_{ij}})
\end{align*}\]

HDP (right):

\[(\alpha_k, \phi_k) \sim \text{DP}(\gamma, \text{Dir}(\beta))\]

\[\begin{align*}
\theta_j &\sim \text{DP}(\eta, \alpha) \\
Z_{ij} &\sim \text{Cat}(\theta_j) \\
X_{ij} &\sim \text{Cat}(\phi_{Z_{ij}})
\end{align*}\]
Collapsed posterior sampling

- Marginalize out $\phi_k$ and $\theta_j$.

- LDA:

$$p(Z_{ij} = k|Z^{-ij}, X, \alpha, \beta) \propto \frac{N_{w_k}^{-ij} + \beta}{\sum_w N_{w_k}^{-ij} + \beta} \left( N_{kj}^{-ij} + \alpha \right)$$  \tag{1}

- HDP:

$$p(Z_{ij} = k|Z^{-ij}, X, \alpha, \beta) \propto \begin{cases} \frac{N_{w_k}^{-ij} + \beta}{\sum_w N_{w_k}^{-ij} + \beta} \left( N_{kj}^{-ij} + \eta \alpha_k \right), & \text{if } k \text{ exists} \\ \eta \alpha_k W, & \text{if } k \text{ new} \end{cases}$$  \tag{2}
Collapsed & uncollapsed posterior sampling

- Collapsed sampler works better.
- Focus on building distributed algorithms for the collapsed sampler.
Approximate distributed LDA (AD-LDA)

- Need $N_{wk}$ (global) and $N_{kj}$ (local) in the sampling.
- In each iteration:
  - copy global counts $N_{wk}$ to each processor $p$
  - each processor sample the local counts $N_{kj}$ independently
  - synchronize

**Algorithm 1 AD-LDA**

repeat
  for each processor $p$ in parallel do
    Copy global counts: $N_{wkp} \leftarrow N_{wk}$
    Sample $z_p$ locally: LDA-Gibbs-Iteration($x_p, z_p, N_{kjp}, N_{wkp}, \alpha, \beta$)
  end for
  Synchronize
  Update global counts: $N_{wk} \leftarrow N_{wk} + \sum_p (N_{wkp} - N_{wk})$
until termination criterion satisfied
Hierarchical distributed LDA (HD-LDA)

- Instead of copying $N_{wk}$ to each processor, define a hierarchical dependency between $\phi_{wk}$ and its local copies $\varphi_{kp}$:

  $\varphi_{kp} \sim \text{Dir}(\beta_k \phi_k)$ \hspace{1cm} (3)

  $\beta_k \sim \text{Gamma}(a, b)$ \hspace{1cm} (4)

- Works a little bit better than AD-LDA in general.
Hierarchical distributed LDA (HD-LDA)

**Algorithm 2 HD-LDA**

repeat
  for each processor $p$ in parallel do
    Sample $z_p$ locally: LDA-Gibbs-Iteration($x_p, z_p, N_{kjp}, N_{wkp}, \alpha_p, \beta_k \Phi_k$)
    Sample $\alpha_p$ locally
  end for
  Synchronize
  Sample: $\beta_k, \Phi_k$
  Broadcast: $\beta_k, \Phi_k$
until termination criterion satisfied
Different from AD-LDA in that new topics are to be born

strategy: merge

**Algorithm 3 AD-HDP**

```
repeat
  for each processor \( p \) in parallel do
    Sample \( z_p \) locally: HDP-Gibbs-Iteration(\( x_p, z_p, N_{kjp}, N_{wkp}, \alpha_{kp}, \beta, \gamma, \eta \))
    Report \( N_{wkp}, \alpha_{kp} \) to master node
  end for
  Synchronize
  Update global counts (and merge new topics): \( N_{wk} \leftarrow N_{wk} + \sum_p (N_{wkp} - N_{wk}) \)
  \( \alpha_k \leftarrow \frac{(\sum_p \alpha_{kp})}{P} \)
  Sample: \( \eta, \alpha_k, \gamma \)
  Broadcast: \( N_{wk}, \alpha_k, \gamma, \eta \)
until termination criterion satisfied
```
Three heuristic ways to merge new topics

- Merge by matching integer topic label.

- Merge by bipartite matching:
  - Hungarian algorithm, expensive, no better than merge by matching integer topic label

- Merge by greedy matching schema:
  - sequentially compared new topics with a global set of new topics
  - if similarity larger than a threshold, merge them
  - otherwise, create a new global new topic
Datasets and models

- WIKIPEDIA and PUBMED used for testing speedups, no testing data.

<table>
<thead>
<tr>
<th></th>
<th>KOS</th>
<th>NIPS</th>
<th>WIKIPEDIA</th>
<th>PUBMED</th>
<th>NEWSGROUPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>3,000</td>
<td>1,500</td>
<td>2,051,929</td>
<td>8,200,000</td>
<td>19500</td>
</tr>
<tr>
<td>$W$</td>
<td>6,906</td>
<td>12,419</td>
<td>120,927</td>
<td>141,043</td>
<td>27,059</td>
</tr>
<tr>
<td>$N$</td>
<td>467,714</td>
<td>2,166,058</td>
<td>344,941,756</td>
<td>737,869,083</td>
<td>2,057,207</td>
</tr>
<tr>
<td>$D_{\text{test}}$</td>
<td>430</td>
<td>184</td>
<td>-</td>
<td>-</td>
<td>498</td>
</tr>
</tbody>
</table>

- LDA, AD-LDA, HD-LDA
- HDP, AD-HDP (with and without matching)
HDP (AD-HDP) is better.

HD-LDA marginally better (if not comparable) than AD-LDA.
AD-HDP works almost the same with HDP on perplexities.
AD-HDP generate less topics.
Matching improves on the rate of convergence.
Speedups on large datasets

- Use the supercomputer DataStar
- MPI implementation

![Graph showing speedup vs. number of processors for different datasets and models.](image)
Analysis of AD-LDA with synthetic datasets

- Generate LDA data with vocabulary words $W = 3$ and $K = 2$, did not report #words generated.
- Run with $P = 2$.
- AD-LDA very close to LDA.

**Figure**: Left: $L_1$ distance to the true mode; Right: 50 samples of $\phi$ (projected onto the topic simplex) from the equilibrium distribution.
Analysis of AD-LDA with synthetic datasets

- AD-LDA converges closer to those learning by LDA than the true value.

**Figure**: Average $L_1$ error in word-topic distribution versus $P$. 
When does AD-LDA fail?

- Roughly speaking, when not doing synchronization.
- Use KOS data with $P = 2$ and $K = 16$. 

![Graph showing perplexity over iterations for LDA and AD-LDA with different parameters.](image)
Outline

1. Distributed Algorithms for Topic Models
   - Distributed LDA and HDP
   - Experiments

2. Distributed Stochastic Gradient MCMC
   - The Algorithm
   - Experiments
Setup

- Given data \(X = \{x_1, \cdots, x_N\}\), a generative model 
  \(p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta)\) and prior \(p(\theta)\), we want to compute 
  the posterior \(\pi(\theta) \triangleq p(\theta|X) \propto p(X|\theta)p(\theta)\).
- Define \(\bar{g}(X|\theta) \triangleq \frac{1}{|X|} \sum_{x \in X} \nabla_{\theta} \log p(x|\theta)\).
- Stochastic gradient Langevin dynamics (SGLD):
  \[
  \theta_{t+1} \leftarrow \theta_t + \frac{\epsilon_t}{2} \{\nabla_{\theta} \log p(\theta_t) + N\bar{g}(X^n_t|\theta)\} + \nu_t , \tag{5}
  \]
  where \(X^n_t\) is a minibatch of size \(n\) in iteration \(t\), \(\nu_t \sim N(0, \epsilon)\).
- Goal is to develop a distributed/parallel version of the SGLD for large scale learning.
In distributed system \( \bar{g}(X_{i}^{n}|\theta) \) is hard to compute because the whole data cannot be accessed in a local machine.

Define a unbiased estimator \( f(\theta,Z;X) \) for \( \bar{g}(X|\theta) \) where \( Z \) is a set of auxiliary random variables, such that

\[
\mathbb{E}_{Z} [f(\theta,Z;X)] = \bar{g}(\theta;X) .
\]  

(6)

Use it in SGLD instead of \( \bar{g}(\theta;X) \):

\[
\theta_{t+1} \leftarrow \theta_{t} + \frac{\epsilon_{t}}{2} \{ \nabla_{\theta} \log p(\theta_{t}) + N f(X_{t}^{n}|\theta) \} + \nu_{t} ,
\]  

(7)
Distributed Stochastic Gradient MCMC

The Algorithm

Claim

\[ \theta_{t+1} \leftarrow \theta_t + \frac{\epsilon_t}{2} \{ \nabla_{\theta} \log p(\theta_t) + Nf(X_t^n|\theta) \} + \nu_t , \quad (8) \]

- It is claimed without proof that (8) generates corrected posterior samples if:
  1. \( f(X^n_t|\theta) \) is an unbiased estimate of \( \bar{g}(X|\theta) \)
  2. \( \nu_t \) decreases to 0 as \( t \to \infty \)

- Not necessarily true in theory:
  1. 1 is true when \( \nu_t \) is fixed to a small enough value (not decreasing) [Vollmer et al., 2015]
  2. not obviously be true with a decreasing \( \nu_t \)
  3. the condition for a decreasing \( \nu_t \) to be true relates to the Liapunov function [Teh et al., 2014], though I think these two are closely related
  - a fixed step size is used in experiments

- For now we assume the claim is true.
Algorithm:

1. Each machine loads a partition of the whole data, with size $N_s$
2. Sample a machine with probability $q_s$ ($\sum_s q_s = 1$) and pass the previous $\theta_t$ to this machine
3. Sample the SGLD equation with estimator (9):

$$f_d(\theta; X^n_s) \triangleq \frac{N_s}{Nq_s} \bar{g}(\theta; X^n_s)$$ (9)

Theorem

SGLD with the estimator (9) is a valid estimator and generates correct posterior samples.
Distributed SGLD with parallel chains

1. Store the data on a distributed file system to minimize the transition cost.
2. Store the parameter $\theta$ with distributed caches to minimize the communication cost.
3. Start several parallel MCMC chains, each evolves via the SGLD.
4. Jump to other machines via a scheduler $h(Q)$, e.g., uniformly scheduling.
5. Run a few mini-batches before jumping to other machines.
6. Same estimator as the above case.

Theorem

SGLD with parallel chains and a scheduler generates correct posterior samples.
Adaptive load balance

- The above method becomes a problem when some machines are much slower.
- Propose to let the chain in the fast machine do more samples before jumping to another machine.

**Theorem**

*SGLD with adaptive load balance generates correct posterior samples.*
The variance of samples from different chains is too large.

Propose to reduce variance by averaging samples from a set of chains, e.g.,

\[
\theta_{t+1} = \frac{1}{R} \sum_{r=1}^{R} \theta^r_{t+1}
\]

\[
\theta_t + \frac{\varepsilon}{2} \left\{ \nabla_{\theta} \log p(\theta_t) + \frac{N}{nR} \sum_{x \in \cup X^n_{t,r}} g(\theta_t; x) \right\} + \bar{\nu}_t,
\]

where \( \bar{\nu}_t \sim N(0, \varepsilon_t) \).

Need to add a corrected noise \( \eta_t \sim N(0, \frac{R-1}{R} \varepsilon) \).
Multivariate Gaussian demonstration

\[ x_i \sim N(\mu_x, \Sigma_x), \quad \mu_x \sim N(\mu_0, \Sigma_0) \quad (12) \]

- 20 machines
- \( N = 20,000, N_s = 500 \sim 1500 \)
- Step size \( \varepsilon = 1e^{-7} \), mini-batch size 300
- Bias correction (red circuit is the true posterior):
Multivariate Gaussian demostration

\[ x_i \sim N(\mu_x, \Sigma_x), \quad \mu_x \sim N(\mu_0, \Sigma_0) \]  

- 4 machines
- \( N = 8,000, \ N_s = 2,000 \)
- Step size \( \epsilon = 2e^{-6} \), mini-batch size 300
- Effect of trajectory lengths (red circuit is the true posterior):
Distributed LDA

Comparison:
- D-SGLD: this method
  - D-CC: complete coupling chains
  - D-CI: complete independent chains
  - D-Hybrid: partially coupling chains
- AD-LDA
- Async-LDA: AD-LDA without synchronization
- SGRLD: stochastic gradient Riemannian Langevin dynamics

Datasets:
- Wikipedia: 4.6M docs, 811M tokens, 7702 vocabulary size
- PubMed Abstract: 8.2 docs, 730M tokens, 39,987 vocabulary
Table: Required time to reach the perplexity that AD-LDA obtains after running 27.7 hr.

Q: how can SGRLD run faster on a single machine than AD-LDA on 20 machines?
Figure: Group size and # groups effects on Wikipedia (left) and Pubmed (right). Generally, for fixed #groups, the larger the group size, the better; For fixed group size, the more groups, the better.
Distributed Stochastic Gradient MCMC

Experiments

On different data sizes

- D-SGLD less affected by data sizes.
- D-SGLD does worse in small sub-datasets in Pubmed (right).
Load balance

- Enforce unbalance of computational speed by dummy delay.
- D-SGLD overcomes this by running more iterations in the faster machines.

**Figure**: Load balance. \((a, b)\) means make the machine \(b\) times slower.
Figure: #topics. Top: Wikipedia; Bottom: Pubmed. Right: Perplexity after $10^4$ updates.
Thanks for your attention!!!