Overview

• Large, streaming data sets are increasingly the norm
• Inference for Big Data has generally been non-Bayesian
• Advantages of Bayes: complex models, coherent treatment of uncertainty, etc.

We deliver:
• SDA-Bayes, a framework for Streaming, Distributed, Asynchronous Bayesian inference
• Experiments demonstrating streaming topic discovery with comparable predictive performance to non-streaming algorithms
  – Corporuses used are Wikipedia (3.6M documents) and the scientific journal Nature (350K documents)

Background

• Posterior: adjusted belief about unknowns \( \theta \) after observing data \( x \)
• Variational Bayes (VB): finds approximate posterior by solving an optimization problem (minimize Kullback-Liebler divergence)
• Batch VB: solves a VB optimization problem using coordinate descent
  – Requires passing over the data multiple times
• Stochastic Variational Inference (SVI): solves a VB optimization problem using stochastic gradient descent
  – Requires specifying the data size in advance (so not streaming)
  – Generally much better predictive performance after a single data pass than batch VB

SDA-Bayes: Streaming

• Can iteratively update posterior after new data using Bayes’ theorem
  \[ p(\theta \mid x_1, \ldots, x_T) \propto p(s_T \mid x_T) \cdot p(s_T \mid x_1, \ldots, x_T) \]
• Choose any batch approximation \( \mathcal{A} \) to the posterior

\[ \begin{array}{c}
\text{data} \\
p(\theta) \\
\text{q}(\theta) = p(\theta \mid x)
\end{array} \begin{array}{c}
\mathcal{A} \\
\text{batch alg}
\end{array} \begin{array}{c}
p(\theta \mid x_1) \\
q_1(\theta) \\
p(\theta \mid x_1, x_2)
\end{array}
\]

• Can iterate as long as approximation has same form as prior

SDA-Bayes: Distributed

• Posteriors calculated in parallel can be combined using Bayes’ rule:
  \[ p(\theta \mid x_1, \ldots, x_T) \propto \prod_{t=1}^{T} p(x_t \mid \theta) \cdot p(\theta) \propto \prod_{t=1}^{T} p(x_t \mid \theta) \cdot p(\theta)^{-1} \cdot p(\theta) \]
• Can combine approximated posteriors in similar fashion
• If the prior and approximate posterior are in the same exponential family, the update is simply vector addition
  – Sufficient statistic \( T(\theta) \), prior parameter \( \zeta_0 \), \( n \)th approximate posterior parameter \( \zeta_n \)
  \[ p(\theta \mid x_1, \ldots, x_T) = q(\theta) \propto \exp \left\{ \zeta_0 + \sum_{n=1}^{N} (\zeta_n - \zeta_0) \cdot T(\theta) \right\} \]

SDA-Bayes: Asynchronous

• Each worker iterates the following steps.
  1. Collect a new data point \( x \).
  2. Copy the master posterior parameter locally: \( \xi(\text{local}) \rightarrow \xi(\text{post}) \)
  3. Compute the local approximate posterior parameter \( \xi(\text{local}) \) as the prior parameter
  4. Return \( \Delta \xi \) from a worker, it updates synchronously: \( \xi(\text{post}) \leftarrow \xi(\text{post}) + \Delta \xi \)

Case Study: Latent Dirichlet Allocation (LDA)

• LDA: a model for the content of documents
• Topic: a theme potentially shared by multiple documents
• (Unsupervised) inference problem: discover the topics and identify which topics occur in which documents

Experimental Setup

• We compare SDA-Bayes with a batch VB primitive for \( \mathcal{A} \) ("Streaming Variational Bayes") to SVI
• All algorithms learn topics using an LDA model with 100 topics
• Data: 3.6M Wikipedia and 350K Nature documents for training; 10K Wikipedia and 1K Nature documents for testing.
• Documents seen in minibatches (small groups) rather than one by one
• Log predictive probability: on held-out words in held-out testing documents
  – We use an approximation of this as our performance measure in experiments; higher is better

Results

• SDA performs at least as well as SVI, an algorithm not designed for the streaming setting (32 threads and 1 thread depicted in table)

<table>
<thead>
<tr>
<th>Wikipedia</th>
<th>Nature</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-SDA</td>
<td>1-SDA</td>
</tr>
<tr>
<td>Log pred prob</td>
<td>−7.31</td>
</tr>
<tr>
<td>Time (hours)</td>
<td>2.09</td>
</tr>
</tbody>
</table>

• Using more threads in SDA improves performance and runtime

References