Streaming Variational Bayes

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, Michael I. Jordan

UC Berkeley
Discussion led by Miao Liu

September 13, 2013
Introduction

The SDA-Bayes Framework
  Streaming Bayesian updating
  Distributed Bayesian updating
  Asynchronous Bayesian updating

Experiments
The advantages of the Bayesian paradigm: hierarchical modeling, coherent treatment of uncertainty

These advantages seem out of reach in the setting of Big Data

An exception: variational Bayes (VB)

- stochastic variational inference (SVI) = VB + stochastic gradient descent [Hoffman et. al. 2012]
- the objective function: the variational lower bound on the marginal likelihood
- the stochastic gradient is computed for a single data point (document) at a time

The problem of SVI

- The objective is based on the conceptual existence of a full data set involving $D$ data points (documents)
  - $D$ is a fixed number and must be specified in advance
  - does not apply for the streaming setting

Development in computer architectures, which permit distributed and asynchronous computations
Introduction

- The aim of this paper
  - develop a **scalable** approximate Bayesian inference algorithm in the real streaming setting

- Methodology
  - A recursive application of Bayes theorem provides a sequence of posteriors, not a sequence of approximations to a fixed posterior
  - The posteriors are approximated by VB

- Similar methods
  - density filtering [Opper, 1999]
  - expectation propagation [Minka, 2001]

- Related work
  - MCMC approximations [Canini et. al. AISTATS09]
  - VB and VB-like approximations [Honkela and Valpola, 2003, Luts et al. preprint arXiv]
Streaming Bayesian updating

- Consider data $x_1, x_2, \cdots i.i.d.$ $p(x|\Theta)$, the collection of $S$ data points, $C_1 := (x_1, \cdots, x_S)$
- The posterior after the $b$th mini batch

$$p(\Theta|C_1, \cdots, C_b) \propto p(C_b|\Theta)p(\Theta|C_1, \cdots, C_{b-1}) \quad (1)$$

- repeated application of Eqn (1) is streaming
- automatically yields the new posterior without needing to revisit old data points
- An approximation algorithm $\mathcal{A}$ for computing an approximate posterior $q : q(\Theta) = \mathcal{A}(C, p(\Theta))$
- setting $q_0(\Theta) = p(\Theta)$, then recursively calculate an approximation to the posterior

$$p(\Theta|C_1, \cdots, C_b) \approx q_b(\Theta) = \mathcal{A}(C, q_{b-1}(\Theta)) \quad (2)$$
Distributed Bayesian updating

- Parallelizing computations increases algorithm throughput
- Calculate individual mini batch posteriors $p(\Theta|C_b)$ (perhaps in parallel), and then combine them to find the full posterior

\[ p(\Theta|C_1, \cdots, C_B) \propto \left[ \prod_{b=1}^{B} p(C_b|\Theta) \right] p(\Theta) \propto \left[ \prod_{b=1}^{B} [p(\Theta|C_b)p(\Theta)^{-1}] \right] p(\Theta) \tag{3} \]

- The corresponding approximate update

\[ p(\Theta|C_1, \cdots, C_B) \approx q(\Theta) \propto \left[ \prod_{b=1}^{B} A(C_b, p(\Theta))p(\Theta)^{-1} \right] p(\Theta), \tag{4} \]

- Assumptions

1. $p(\Theta) \propto \exp\{\xi_0 \cdot T(\Theta)\}$ is an exponential family distribution for $\Theta$ with sufficient statistic $T(\Theta)$ and natural parameter $\xi_0$
2. Further assume $A$ returns a distribution in the same exponential family $q_b(\Theta) \propto \exp\{\xi_b \cdot T(\Theta)\}$

- The update in Eq.(4) becomes

\[ p(\Theta|C_1, \cdots, C_B) \approx q(\Theta) \propto \exp \left\{ \left[ \xi_0 + \sum_{b=1}^{B}(\xi_b - \xi_0) \right] \cdot T(\Theta) \right\} \tag{5} \]

- It is not necessary to assume any conjugacy
Asynchronous Bayesian updating

- Asynchronous algorithms can decrease downtime in the system
- The proposed asynchronous algorithm is in the spirit of Hogwild! [Niu et al. NIPS2011]
- An asynchronous scheme (conceptual stepping stone, not used in practice)
  - Workers (processors that solve a subproblem)
    1. collect a new minibatch $\mathcal{C}$.
    2. compute the local approximate posterior $\xi \leftarrow A(\mathcal{C}, \xi_0)$
    3. return $\Delta \xi := \xi - \xi_0$ to the master
  - The master
    - starts by assigning the posterior to equal the prior: $\xi^{post} \leftarrow \xi_0$
    - each time the master receives a quality $\Delta \xi$ from any worker, it updates the posterior synchronously: $\xi^{post} \leftarrow \xi^{post} + \Delta \xi$
The master initializes its posterior estimate to the prior:
\[ \xi^{\text{(post)}} \leftarrow x_0 \]

Each worker continuously iterates between four steps
1. collect a new minibatch \( C \).
2. copy the master posterior value locally \( \xi^{\text{(local)}} \leftarrow \xi^{\text{(post)}} \)
3. compute the local approximate posterior \( \xi \leftarrow A(C, \xi^{\text{(local)}}) \)
4. return \( \Delta \xi := \xi - \xi_0 \) to the master

Each time the master receives a quality \( \Delta \xi \) from any worker, it updates the posterior synchronously: \( \xi^{\text{post}} \leftarrow \xi^{\text{post}} + \Delta \xi \)

The key difference: the latest posterior is used as a prior in the second frameworks:

The latter framework introduces a new layer of approximation, but works better in practice
Case study: latent Dirichlet allocation (Blei et al., 2003)

Notations

- $\alpha$: the parameter of Dirichlet prior on the per-document topic distribution
- $\beta$: the parameter of Dirichlet prior on the per-topic word distribution
- $\theta_i$: the topic distribution for document $i$
- $\phi_k$: the word distribution for topic $k$
- $z_{ij}$: the topic for the $j$th word in document $i$
- $w_{ij}$: the specific word

The generative process

$$
\theta_i \sim \text{Dir}(\alpha), \text{ where } i \in \{1, \cdots, M\} \tag{6}
$$

$$
\phi_k \sim \text{Dir}(\beta), \text{ where } k \in \{1, \cdots, K\} \tag{7}
$$

For each of the words $w_{ij}$, where $j \in \{1, \cdots, N_i\}$

$$
z_{ij} \sim \text{Categorical}(\theta_i), w_{ij} \sim \text{Categorical}(\phi_{z_{ij}}) \tag{9}
$$
The posterior of LDA

\[ p(\beta, \theta, z|C, \eta, \alpha) = \left[ \prod_{k=1}^{K} \text{Dirchlet}(\beta_k|\eta_k) \right] \cdot \left[ \prod_{d=1}^{D} \text{Dirchlet}(\theta_d|\alpha) \right] \cdot \left[ \prod_{d=1}^{D} \prod_{n=1}^{N_d} \theta_{zn} \beta_{zn, w_{dn}} \right] \]

Posterior-approximation algorithms

- Mean-field variational Bayesian: \( \text{KL}(q||p) \)
- Expectation propagation [Minka, 2001]: \( \text{KL}(p||q) \)

Other single-pass algorithms for approximate LDA posteriors

- Stochastic variational inference
- Sufficient statistics
Algorithm 1: VB for LDA

Input: Data \((n_d)_{d=1}^D\); hyperparameters \(\eta, \alpha\)
Output: \(\lambda\)
Initialize \(\lambda\)
while \((\lambda, \gamma, \phi)\) not converged do
  for \(d = 1, \ldots, D\) do
    \((\gamma_d, \phi_d) \leftarrow \text{LocalVB}(d, \lambda)\)
    \(\forall (k, v), \lambda_{kv} \leftarrow \eta_{kv} + \sum_{d=1}^D \phi_{dek} n_{dv}\)
  \end{algorithm}

Subroutine \text{LocalVB}(d, \lambda)

Output: \((\gamma_d, \phi_d)\)
Initialize \(\gamma_d\)
while \((\gamma_d, \phi_d)\) not converged do
  \(\forall (k, v), \text{set } \phi_{dek} \propto \exp(\mathbb{E}_q[\log \theta_{dk}] + \mathbb{E}_q[\log \beta_{kv}] )\) (normalized across \(k\))
  \(\forall k, \gamma_{dk} \leftarrow \alpha_k + \sum_{v=1}^V \phi_{dek} n_{dv}\)
\end{algorithm}

Algorithm 2: SVI for LDA

Input: Hyperparameters \(\eta, \alpha, D, (\rho_t)_{t=1}^T\)
Output: \(\lambda\)
Initialize \(\lambda\)
for \(t = 1, \ldots, T\) do
  Collect new data minibatch \(C\)
  foreach document indexed \(d\) in \(C\) do
    \((\gamma_d, \phi_d) \leftarrow \text{LocalVB}(d, \lambda)\)
    \(\forall (k, v), \lambda_{kv} \leftarrow \eta_{kv} + \frac{D}{|C|} \sum_{d \in C} \phi_{dek} n_{dv}\)
    \(\forall (k, v), \lambda_{kv} \leftarrow (1 - \rho_t) \lambda_{kv} + \rho_t \lambda_{kv}\)
\end{algorithm}

Algorithm 3: SSU for LDA

Input: Hyperparameters \(\eta, \alpha\)
Output: A sequence \(\lambda^{(1)}, \lambda^{(2)}, \ldots\)
Initialize \(\forall (k, v), \lambda_{kv}^{(0)} \leftarrow \eta_{kv}\)
for \(b = 1, 2, \ldots\) do
  Collect new data minibatch \(C\)
  foreach document indexed \(d\) in \(C\) do
    \((\gamma_d, \phi_d) \leftarrow \text{LocalVB}(d, \lambda)\)
    \(\forall (k, v), \lambda_{kv}^{(b)} \leftarrow \lambda_{kv}^{(b-1)} + \sum_{d \in C} \phi_{dek} n_{dv}\)
\end{algorithm}

Algorithm 4: EP for LDA

Input: Data \((w_d)_{d=1}^D\); hyperparameters \(\eta, \alpha\)
Output: \(\lambda\)
Initialize \(\forall (k, d, v), \chi_{kde} \leftarrow 0\) and \(\zeta_{dv} \leftarrow 0\)
while \((\chi, \zeta)\) not converged do
  foreach \((d, v)\) with \(n_{dv} \geq 1\) do
    /* Variational distribution without the word token \((d, v)\) */
    \(\forall k, \lambda_{k(d,v)} \leftarrow \eta_k + (n_{dv} - 1) \chi_{kde} + \sum_{(d',v') \neq (d,v)} n_{d'v'} \chi_{kde} \)
    \(\gamma_{d(v)} \leftarrow \alpha + (n_{dv} - 1) \zeta_{dv} + \sum_{v' \neq v} n_{dv} \zeta_{dv} \)
    If any of \(\lambda_{k(d,v)}\) or \(\gamma_{d(v)}\) are non-positive, skip updating this \((d, v)\)
    /* Variational parameters from moment-matching */
    \(\forall (k, u), \text{compute } \lambda_{ku} \text{ from Eq. (12)}\)
    \(\forall k, \text{compute } \gamma_{dk} \text{ from Eq. (13)}\)
    /* Type-level updates to parameter values */
    \(\forall k, \chi_{kde} \leftarrow n_{de}^{-1} \left( \lambda_k - \lambda_{k(d,v)} \right) + (1 - n_{dv}^{-1}) \chi_{kde} \)
    \(\zeta_{dv} \leftarrow n_{dv}^{-1} \left( \gamma_d - \gamma_{d(v)} \right) + (1 - n_{dv}^{-1}) \zeta_{dv} \)
    Other \(\chi, \zeta\) remain unchanged
    /* Global variational parameters */
    \(\forall k, \lambda_k \leftarrow \eta_k + \sum_{d=1}^D \sum_{v=1}^V n_{dv} \chi_{kde} \)
\end{algorithm}
Experiments

▪ Data set
  ▪ the full Wikipedia corpus (3,611, 558 for training, 10,000 for testing)
  ▪ the Nature corpus (351, 525 for training, 1,024 for testing)

▪ Experiment setup
  ▪ Hold out 10,000/1,024 Wikipedia/Nature documents for testing
  ▪ For all cases, fit an LDA model with $K = 100$ topics
  ▪ Chose hyperparameters as $\forall k, \alpha_k = 1/K$, $\forall (k, v), \eta_{k,v} = 1$
<table>
<thead>
<tr>
<th></th>
<th>Wikipedia</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32-SDA</td>
<td>1-SDA</td>
<td>SVI</td>
<td>SSU</td>
</tr>
<tr>
<td>Log pred prob</td>
<td>-7.30</td>
<td>-7.38</td>
<td>-7.39</td>
<td>-7.94</td>
</tr>
<tr>
<td>Time (hours)</td>
<td>2.07</td>
<td>25.37</td>
<td>6.56</td>
<td>10.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nature</td>
<td>32-SDA</td>
<td>1-SDA</td>
<td>SVI</td>
</tr>
<tr>
<td>Log pred prob</td>
<td>-7.07</td>
<td>-7.13</td>
<td>-7.14</td>
<td>-7.89</td>
</tr>
<tr>
<td>Time (hours)</td>
<td>0.31</td>
<td>7.00</td>
<td>1.73</td>
<td>1.99</td>
</tr>
</tbody>
</table>

Table 1: A comparison of (1) log predictive probability of held-out data and (2) running time of four algorithms: SDA-Bayes with 32 threads, SDA-Bayes with 1 thread, SVI, and SSU.
Figure 2: SDA-Bayes log predictive probability (two upper) and run time (two lower) as a function of number of threads.
Figure 3: Sensitivity of SVI and SDA-Bayes to some respective parameters. Legends have the same top-to-bottom order as the rightmost curve points.