Scalable Bayes via Double-Parallel Computing

Faming Liang

University of Florida

December 17, 2016
We propose an efficient algorithm for Bayesian analysis of big data, which consists of a few steps: (i) divide the big dataset into some smaller subsets, (ii) for each subset, simulate MCMC samples from an approximate posterior distribution, (iii) shift the mean of each subset posterior distribution to the global mean. We show that the full data posterior can be consistently approximated by the mixture of the mean-shifted subset posterior distributions. To further speed up computation, we implement the Pop-SAMC algorithm, a parallel MCMC algorithm, to draw samples from each subset posterior. Since parallel computing is conducted both within and across subsets, we call the proposed algorithm a double-parallel algorithm. The proposed algorithm is illustrated using both simulated and real data examples.
Motivation-I: The integration of computer technology into science and daily life has enabled scientists to collect a vast amount of data, e.g., climate data, multi-omics data, mobile health data, electronic medical record, web transaction logs, and credit card records.

- High-dimensional, while the sample size can be small (small-$n$-large-$p$)
- BIG (volume, velocity, variety)

We need efficient methods for making statistical inference for high-dimensional and big data!
Motivation-II With growing data size typically comes a growing complexity of data structure, of the patterns in the data, and of the models needed to account for the patterns.

Markov chain Monte Carlo (MCMC) methods have proven to be a powerful tool for analyzing data of complex structures. However, their computer-intensive nature, which typically require a large number of iterations and a complete scan of the full dataset for each iteration, precludes their use for big data problems.

We need new developments of the MCMC methodology to meet the challenges of big data!
Strategy I: subsampling for big data

- Maximum mean log-likelihood estimation (MMLE): Liang et al. (2013)
- Little bag of bootstrap: Kleiner et al. (2014).
- Monte Carlo simulation: Bootstrap Metropolis-Hastings (Liang et al., 2016); Stochastic gradient Hamiltonian Monte Carlo (Chen, Fox and Guestrin, 2014); stochastic gradient Langevin dynamics (Welling and Teh, 2011)
Strategy II: split and merge for big data

The embarrassingly parallel structure and the theoretical guarantee make the methods extremely attractive.

- Variable selection: Chen and Xie (2014, split in ”n”); Song and Liang (2015, split in ”p”).
- Single value decomposition (Liang et al., 2016)
- Monte Carlo Simulation: Consensus Monte Carlo (Scott et al., 2016); WASP Monte Carlo (Srivastava et al., 2015).
Consensus Monte Carlo

- Split the dataset into $k$ subsets, each with $m = n/k$ samples.
- For each subset, simulate samples from the posterior

$$
\pi(\theta|X_{[j]}) \propto \int_{i=1}^{m} f(X_{ji}|\theta) \pi^{1/k}(\theta).
$$

- Aggregate the subset posterior samples by setting

$$
\hat{\theta}_i = \sum_{j=1}^{k} w_j \theta_{ji}, \quad i = 1, 2, \ldots,
$$

where $\theta_{j1}, \theta_{j2}, \ldots$ denote the samples from the $j$th subset posterior, and $w_j = \Sigma_j^{-1}$ denotes the inverse of the covariance matrix of the $j$th subset posterior.
WASP (WASserstein Posterior) Monte Carlo

- Split the dataset into $k$ subsets, each with $m = n/k$ samples.
- For each subset, simulate samples from the posterior

$$\tilde{\pi}(\theta|X_{[j]}) \propto \int_{i=1}^{m} f^k(X_{ji} | \theta)\pi(\theta).$$

- Averaging subset posteriors through WASP:
  - Approximate each subset posterior as an empirical measure with simulated atoms, i.e., set

$$\hat{\pi}(\theta|X_{[j]}) = \sum_{i=1}^{N_j} \frac{1}{N_j} \delta_{\theta_{ji}}(\cdot).$$

- Estimate the full data posterior by

$$\hat{\pi}(\theta|X) = \sum_{j=1}^{k} \sum_{i=1}^{N_j} a_{ji} \delta_{\theta_{ji}}(\cdot),$$

where $a_{ji}$ are determined by calculating the barycenter with respect to a Wasserstein distance between $\hat{\pi}(\theta|X_{[j]})$’s. The barycenter can be calculated by linear programming.
Data Parallel

- Split the dataset into $k$ subsets, each containing $m = n/k$ samples.
- For each subset, simulate samples from the posterior
  \[
  \tilde{\pi}(\theta|X_{[j]}) \propto \int_{i=1}^{m} f^k(X_{ji}|\theta)\pi(\theta).
  \]
- Let $\mu_1, \ldots, \mu_k$ denote the means of the subset posteriors, let $\hat{\mu} = \sum_{j=1}^{k} \mu_j / k$, and merge the subset posteriors by setting
  \[
  \tilde{\pi}(\theta|X) = \frac{1}{k} \sum_{j=1}^{k} \tilde{\pi}(\theta + \hat{\mu} - \mu_j|X_{[j]}).
  \]
Posterior Consistency

Assuming that the observations are a random sample from some fixed measure $P_0$. The posterior is said to be consistent if, as a random measure, it concentrates on arbitrarily small neighborhoods of $P_0$, with probability tending to 1 or almost surely, as $n \to \infty$. 
Theoretical Justification

A1 The log-likelihood function is Laplace-regular.
A2 $\pi(\theta)$ is four times continuous differentiable on $\Theta$.
A3 The number of subsets $k$ can increase slowly with $n$, but can not exceed $O(n^{1/2})$.
A4 $h(\theta)$ is a four times continuously differentiable function of $\theta$. 


Theoretical Justification

**Theorem 1.** If A1-A4 are satisfied, then we have

\[
E\left[ E_{\tilde{\pi}(h(\theta)|x)} h(\theta) - E_{\pi(\theta|x)} h(\theta) \right]^2 = O(m^{-2}),
\]

\[
E\left| \text{Var}_{\tilde{\pi}(h(\theta)|x)} h(\theta) - \text{Var}_{\pi(\theta|x)} h(\theta) \right| = o(n^{-1}),
\]

\[
E \left( d^2(\pi(h(\theta)|X), \delta_{h(\theta^*)}) \right) = 2 \frac{\text{tr} \left( H_{(1)}^* I^{-1} H_{(1)}^{*'} \right)}{n} + o(n^{-1}),
\]

\[
E \left( d^2(\tilde{\pi}(h(\theta)|X), \delta_{h(\theta^*)}) \right) = 2 \frac{\text{tr} \left( H_{(1)}^* I^{-1} H_{(1)}^{*'} \right)}{n} + o(n^{-1}),
\]

where \( H_{(1)}^* = \frac{\partial h(\theta)}{\partial \theta^T} \big|_{\theta=\theta^*}, \) and

\[
d^2(\tilde{\pi}(h(\theta)|X), \delta_{h(\theta^*)}) = \int_\Theta \left| h(\theta) - h(\theta^*) \right|^2 \tilde{\pi}(\theta|X) d\theta
\]

is the Wasserstein distance of order 2 between \( \tilde{\pi}(h(\theta)|X) \) and the Dirac measure at \( \theta^* \).
Parallel MCMC

- **Parallel MCMC**: Many short runs versus a single long run:

  Many short runs is only “better” than one long run when both give rotten answers. —Charles Geyer.

  Many short runs can only diagnose nonconvergence when you can quickly get from the starting distribution to every interesting feature of the equilibrium distribution. It only works in toy problems where you already know the answer. —Charles Geyer.

- **Parallel SAMC**: Population SAMC is more efficient than single chain SAMC due to the interactions between different chains (Song, Wu and Liang, 2014, Adv. Appl. Prob.)
Suppose that we are interested in sampling from a distribution,

\[ f(x) = c \psi(x), \quad x \in \mathcal{X}, \]

where \( \mathcal{X} \) is the sample space and \( c \) is an unknown constant.
Pop-SAMC Algorithm

Let $E_1, \ldots, E_m$ denote a partition of the sample space $\mathcal{X}$. For example, the sample space can be partitioned according to the energy function of $f(x)$, i.e., $U(x) = -\log \psi(x)$, into the following subregions: $E_1 = \{x : U(x) \leq u_1\}$, $E_2 = \{x : u_1 < U(x) \leq u_2\}$, $\ldots$, $E_{m-1} = \{x : u_{m-2} < U(x) \leq u_{m-1}\}$ and $E_m = \{x : U(x) \geq u_m\}$, where $u_1 < u_2 < \cdots < u_{m-1}$ are user-specified numbers.

Given the partition, pop-SAMC seeks to draw samples from the distribution

$$f_w(x) \propto \sum_{i=1}^{m} \frac{\pi_i \psi(x)}{w_i} I(x \in E_i),$$

where $w_i = \int_{E_i} \psi(x) dx$, and $\pi_i$’s specify the desired sampling frequencies for each of the subregions and they satisfy the constraints: $\pi_i > 0$ for all $i$ and $\sum_{i=1}^{m} \pi_i = 1$. 
Pop-SAMC Algorithm

1. (Population sampling) For $i = 1, \ldots, \kappa$, simulate a sample $x_{t+1}^{(i)}$ by running, for one step, the Metropolis-Hastings algorithm which starts with $x_t^{(i)}$ and admit the stationary distribution:

$$f_{\theta_t}(x) \propto \sum_{i=1}^{m} \frac{\psi(x)}{e^{\theta_t,i}} I(x \in E_i),$$

where $\theta_t = (\theta_{t,1}, \ldots, \theta_{t,m})$ and $\theta_{t,i}$ denotes the working estimate of $\log(w_i/\pi_i)$ at iteration $t$. Denote the population of samples by $x_{t+1} = (x_{t+1}^{(1)}, \ldots, x_{t+1}^{(\kappa)}$).

2. (\theta-updating) Set $\theta_{t+1} = \theta_t + a_{t+1} H(\theta_t, x_{t+1})$, where

$$H(\theta_t, x_{t+1}) = \sum_{i=1}^{\kappa} (z_{t+1}^{(i)} - \pi)/\kappa,$$

$z_{t+1}^{(i)} = (I(x_{t+1}^{(i)} \in E_1), \ldots, I(x_{t+1}^{(i)} \in E_m))$, and $\pi = (\pi_1, \ldots, \pi_m)$.
Pop-SAMC Algorithm: Convergence & Asymptotic Normality

Song, Wu and Liang (2014) have proven the following results:

- \( \theta_t \to \theta^* \), a.s., where \( \theta^* = (\theta^*_1, \ldots, \theta^*_m) \) is given by

\[
\theta^*_i = C + \log \left( \int_{E_i} \psi(x) dx \right) - \log(\pi_i), \quad i = 1, \ldots, m,
\]

and \( C \) is a constant.

- Conditioned on the event \( \{ \theta_t \to \theta^* \} \),

\[
\frac{\theta_t - \theta^*}{\sqrt{a_t}} \Rightarrow N(0, \Sigma),
\]

where \( \Sigma \) denotes a positive definite matrix.
Let $\{\theta^p_t\}$ and $\{\theta^s_t\}$ denote the estimates resulted from the pop-SAMC and single-chain SAMC, respectively. Song, Wu and Liang (2014) showed that $(\theta^p_t - \theta_*)/\sqrt{a_t}$ and $(\theta^s_{\kappa t} - \theta_*)/\sqrt{\kappa a_{\kappa t}}$ have the same asymptotic distribution with the convergence rate ratio

\[
\frac{a_t}{\kappa a_{\kappa t}} = \kappa^{\zeta-1},
\]

where $\zeta \in (0.5, 1]$ is given by the gain factor sequence

\[
a_t = \frac{t_0}{t^{\zeta}}, \quad t \geq 1.
\]

Hence, when $\zeta < 1$, the pop-SAMC algorithm is asymptotically more efficient than the single-chain SAMC algorithm. When $\zeta = 1$, the two algorithms have the same asymptotic efficiency.
OpenMP is particularly suitable for a parallel implementation of the pop-SAMC algorithm.

- The **fork** step, which works on population sampling, costs the major portion of the CPU and the parallel execution provides a linear speedup for the simulation.

- The **join** step works on $\theta$-updating, where distribution of the updated $\theta_t$ to different threads is avoided due to its shared memory mode. As shown in our examples, the pop-SAMC algorithm can execute very quickly on OpenMP.
Example 1

A logistic regression with $n = 10^4$ and $\theta^* = (1, 1)^T$, where the covariates $X_1$ and $X_2$ are independently drawn from $N(0, I_n)$. The prior distribution for $\theta$ is $N(0, I_2)$. The samples are divided into 10 subsets.

**Figure:** Binned kernel density estimates in logistic regression, where $\theta^* = (1, -1)^T$ (black dot).
Example 2

We consider a normal linear regression:

\[ y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i, \quad i = 1, 2, \ldots, n \]

where \((\beta_0, \beta_1, \beta_2, \beta_3) = (2, 0.25, 0.25, 0)\), and \(\epsilon_1, \cdots, \epsilon_n\) are i.i.d. normal random errors with mean 0 and variance \(\sigma^2 = 0.25\). The covariates \(x_1\) and \(x_2\) are drawn from \(N(0, I_n)\) independently, and \(x_3 = 0.7x_2 + 0.3e\), where \(e \sim N(0, I_n)\). Hence, \(x_2\) and \(x_3\) are highly correlated with a correlation coefficient of 0.919. For this model, \(\theta = (\beta_0, \beta_1, \beta_2, \beta_3, \sigma^2)\), \(n = 10^4\), and \(k = 10\) subsets. For \((\beta_0, \beta_1, \beta_2, \beta_3)\), we use non-informative prior; for \(\sigma^2\), we set \(\pi(\sigma^2) \propto (\frac{1}{\sigma})^{1/1000}\).
Example 2

Figure: QQ plot for $\theta$ in Example 2. Upper: Proposed method; middle: WASP Monte Carlo; lower: consensus Monte Carlo.
Example 2

- WASP: Due to the limitation of memory, 300 MCMC samples were used from each subset.
- Consensus Monte Carlo: produced $1.8 \times 10^5$ independent samples.
- Proposed method: produced $1.8 \times 10^6$ independent samples.
Real Data Analysis

We apply the proposed method to the MiniBooNE particle identification dataset, downloaded from UCI machine learning repository. This dataset records 130,065 events (observations), including 36,499 signal events and 93,565 background events. Each observation consists of the event type (background event or signal event) and 50 particle ID variables. Using event type as response variable, and 50 particle ID variables as predictors, the final goal is to explore the relationship between event type and particle variables.
Real Data Analysis

Table: The computational time and parameter estimates (of the first 10 variables) for the MiniBooNE particle identification dataset.

<table>
<thead>
<tr>
<th>Time (in hours)</th>
<th>MH Algorithm</th>
<th>Double Parallel (k=20, N=10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var1</td>
<td>-1.3537(0.0314)</td>
<td>-1.3945(0.0333)</td>
</tr>
<tr>
<td>Var2</td>
<td>-0.2113(0.0238)</td>
<td>-0.2138(0.0242)</td>
</tr>
<tr>
<td>Var3</td>
<td>0.0974(0.0168)</td>
<td>0.0996(0.0176)</td>
</tr>
<tr>
<td>Var4</td>
<td>-0.8798(0.0245)</td>
<td>-0.8998(0.0254)</td>
</tr>
<tr>
<td>Var5</td>
<td>0.3182(0.0186)</td>
<td>0.3391(0.0193)</td>
</tr>
<tr>
<td>Var6</td>
<td>0.3832(0.0138)</td>
<td>0.3887(0.0144)</td>
</tr>
<tr>
<td>Var7</td>
<td>0.2745(0.0155)</td>
<td>0.2885(0.0161)</td>
</tr>
<tr>
<td>Var8</td>
<td>-0.4049(0.0401)</td>
<td>-0.4175(0.0428)</td>
</tr>
<tr>
<td>Var9</td>
<td>-0.5273(0.0400)</td>
<td>-0.5591(0.0390)</td>
</tr>
<tr>
<td>Var10</td>
<td>0.2131(0.0158)</td>
<td>0.2128(0.0166)</td>
</tr>
</tbody>
</table>
Discussion

- We have proposed an extremely simple and efficient method for Bayesian analysis of big data.
- The proposed method can produce a consistent approximation to the exact full data posterior distribution.
- The proposed method can be easily implemented on the OpenMP platform.
Acknowledgments

- Student: Jingnan Xue
- NSF grants
- NIH R01GM117597