Recent Developments of Iterative Monte Carlo Methods for Big Data Analysis

Faming Liang

Texas A&M University

February 10, 2014
Abstract

Iterative Monte Carlo methods, such as MCMC, stochastic approximation, and EM, have proven to be very powerful tools for statistical data analysis. 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 analysis. We will provide an overview of the recent developments of iterative Monte Carlo methods for big data analysis. The portion by Liang will focus on the developments of the MCMC and stochastic approximation methods, and that by Liu will focus on the developments of the EM method.
Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...

—Dan Ariely
Big Data

- **Big Data**: Data too large to handle easily on a single server or too time consuming to analyze using traditional statistical methods.

- **Examples of Big Data**:
  - Genome data: The University of North Carolina at Chapel Hill is using big data to find better treatments for patients through genomic sequencing technologies.
  - Atmospheric sciences data: rapidly ballooning observations (e.g., radar, satellites, sensor networks), climate data, ensemble data. (Dr. M. Katzfuss)
  - Social sciences data: social networks (Facebook, LinkedIn, email network), social media data (news, telephone calls; Sentiment analysis, Dr. X. Shen)
  - Finance data, image data (Drs P. Qiu, X. Wang, Y. Wu, M. Xiong, J. Zhang, H. Zhu), etc.
Big Data Benefits

Benefits of Big Data are often spoken of in the future tense. As in, Big Data will someday provide enterprises of all types critical insights that allow for increased profitability, improved efficiency and other untold riches.
Big Data Challenges

• Accessing, using and visualizing data
• Server-side processing and distributed storage
• Limited number of scalable statistical methods: From the view of statistical inference, it is unclear how the current statistical methodology can be transported to the paradigm of big data.
• Modeling: With growing size typically comes a growing complexity of data structures and of the models needed to account for the structures.
• Missing data: It will be addressed by Dr. Chuanhai Liu.
Big Data Analysis

- **Parallel programming**: Spark, Hadoop, snowfall (R package), Dr. Jun Yan will address this issue.

- **Exploratory data analysis**: Tableau Software can be used to visually quickly explore large datasets to find the hidden insights and turn the data into accessible and informative charts, stories and dashboards. (?)

- **Pattern Search and Learning**: Dr. Ping Li, . . . . .

- **Modeling**: Neural networks (*Zite* by a team led by Nando de Freitas at UBC); Bayesian neural network (Liang and Jin, 2013)

- **Statistical Inference**: next page
Strategies used in Big Data Analysis

• Divide and combine:
  • Lin and Xi (2011, SII): Aggregated estimating equation
  • Xie (2013): high dimensional variable selection
  • Song and Liang (2013, JRSSB): Bayesian high dimensional variable selection
  • Gautier et al. (2013): Likelihood modeling

• Online Learning: stream data (Schifano et al., 2013)

• Data/model reduction: M. Katzfuss (2013), using low-rank models for approximate inference of massive data

• Resampling-based iterative Monte Carlo: Liang et al. (2013, JASA), Liang and Jin (2013)

• Bag of Little Bootstraps (Kleiner et al., 2012): provides an efficient way of bootstrapping for big data estimators, which functions by combining the results of bootstrapping multiple small subsets of the big original dataset.
Iterative Monte Carlo algorithms, such as MCMC, stochastic approximation, and EM algorithms, have been widely used in modern statistical data analysis and scientific computing. However, their computer-intensive nature, typically requiring a large number of iterations and a complete scan of the full dataset for each iteration, precludes their use for big data analysis.

Motivated by the successes of iterative Monte Carlo algorithms in analyzing data of complex structures, we propose a general principle for developing iterative Monte Carlo algorithms that are feasible for big data and workable on parallel and distributed architectures: using Monte Carlo averages that are calculated from subsamples in parallel to approximate the quantity that originally needs to calculate from the full dataset.
Resampling-based Iterative Monte Carlo Methods

- Resampling-based stochastic approximation (RSA) Method (Liang et al., 2013, JASA)
- Bootstrap Metropolis-Hastings (BMH) algorithm (Liang and Jin, 2013)
- It can be applied to EM to get an Monte Carlo EM-like algorithm.
RSA Algorithm: Notation

- Let $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ denote a big dataset of $n$ observations.
- Let $f(y|\theta)$ denote the probability density function that is used for modeling the observations, where $\theta$ is the parameter vector of the model.
- Let $Z = \{Y_1^*, \ldots, Y_m^*\}$ denote a subsample, which are drawn randomly and without replacement from the full dataset $\mathcal{Y}$. 
RSA Algorithm: Idea

We model $Z$ using the same density $f(z|\theta)$ as for the entire data, and propose to estimate $\theta$ by minimizing the Kullback-Leibler divergence

$$
KL(f_\theta, g) = - \int \log \left( \frac{f(z|\theta)}{g(z)} \right) g(z) dz,
$$

where $g(\cdot)$ denotes the true density from which the original data are generated.

Given an empirical estimator of $g(\cdot)$, the stochastic approximation algorithm can be used to estimate $\theta$ by solving the system of equations:

$$
\frac{\partial KL(f_\theta, \tilde{g})}{\partial \theta} = -(n)^{-1} \sum_{i=1}^{n} \frac{\partial \log f(z_i|\theta)}{\partial \theta} = -(n)^{-1} \sum_{i=1}^{n} H(\theta, z_i) = 0,
$$

where $\tilde{g}$ denotes an empirical approximation of $g(\cdot)$ based on the data $Y$, $H(\theta, z) = \frac{\partial \log f(z|\theta)}{\partial \theta}$ is the first-order derivative of $\log f(z|\theta)$ with respect to $\theta$, and $z_i$ denotes a subsample drawn from $Y$. Note that $\frac{\partial KL(f_\theta, \tilde{g})}{\partial \theta}$ forms a $U$-statistic with the kernel $H(\theta, z)$. 
To solve (2), the RSA algorithm starts with an initial guess of $\theta$, and then iterates as follows:

(i) Draw in parallel $r$ subsamples $Z_{t+1}^{(1)}, \ldots, Z_{t+1}^{(r)}$ from $\mathcal{Y}$.

(ii) Calculate $H_\theta(\theta_t, Z_{t+1}^{(i)})$ for $i = 1, \ldots, r$ in parallel, and update $\theta_t$ in the equation:

$$
\theta_{t+1} = \theta_t + \frac{a_{t+1}}{r} \sum_{i=1}^{r} H_\theta(\theta_t, Z_{t+1}^{(i)}),
$$

where $a_{t+1}$ is called a gain factor, and $\theta_t$ denote the working estimate of $\theta$ at iteration $t$. 
RSA Algorithm

The gain factor sequence \( \{a_t\} \) should be chosen to satisfy the conditions

\[
\lim_{t \to \infty} a_t = 0, \quad \sum_{t=0}^{\infty} a_t = \infty, \quad \sum_{t=1}^{\infty} a_t^2 < \infty. \tag{3}
\]

The varying truncation version of the stochastic approximation algorithm (Chen and Zhu, 1986; Chen, 2002; Andrieu et al., 2005) can also be applied here for estimation of \( \theta \). In this case, we can set \( \Theta = \mathbb{R}^{d_\theta} \).
Theoretical Justifications for the RSA Algorithm (Liang et al., 2013)

1. *(Convergence & Consistency)* As $t \to \infty$, $\theta_t$ converges almost surely to a solution of (2) conditioned on the set \{ $Y(s_1), \ldots, Y(s_n)$ \}; and as $n \to \infty$, the solution of (2) converges to a point in the set $\Theta_0 = \{ \theta^* : El_{\theta^*}(z) = \sup_{\theta \in \Theta} El_\theta(z) \}$, where $l_\theta(z)$ denotes the log-likelihood function of $z$. 
Theoretical Justifications for the RSA Algorithm (Liang et al., 2013)

2. (Asymptotic normality) As $t \to \infty$ and $n \to \infty$, 
\[
\frac{\theta_t - \theta^*}{\sqrt{a_t}} \text{ converges in distribution to a zero mean Gaussian random variable; that is,}
\]
\[
\theta_t - \theta^* \Rightarrow N(0, \Gamma),
\]
where $\Rightarrow$ denotes the weak convergence, $\theta^*$ denotes a point in $\Theta_0$, and
\[
\Gamma = a_t \Sigma_{sa}/r + H_*^{-1} \Sigma H_*^{-1},
\]
where $\Sigma_{sa}$ denotes the covariance matrix due to the random error of stochastic approximation, 
$H_* = E[\partial^2 l_\theta(z)/\partial \theta \partial \theta'|_{\theta=\theta_*}]$, and $\Sigma$ is the covariance matrix 
of the $U$-statistic defined by the kernel $\partial l_\theta(z)/\partial \theta|_{\theta=\theta_*}$. 
RSA Algorithm: Remarks

- RSA leads to a general parameter estimation method for big data problem: maximum mean log-likelihood estimation (MMLE); that is, finding $\theta$ such that $E_{\theta}(z)$ is maximized. By Jensen’s inequality, $E_{\theta^*} \log(f(z|\theta)/f(z|\theta^*)) \leq 0$, which implies

$$E_{\theta}(z) \leq E_{\theta^*}(z),$$

where $\theta^*$ denotes the true parameter value.

- RSA is scalable to big data, as for which each iteration involves only some subsamples that can be easily handled on a parallel machine.

- Since it does not involve steps such as data partition and results aggregation, RSA can be applied to the problem where the observations are generally dependent.
RSA Algorithm: Remarks

- The choice of $r$, the number of subsamples drawn at each iteration, can affect the convergence rate of $\theta_t$. The convergence theory can be studied under the framework of population stochastic approximation algorithms (Younes, 1999; Song, Wu and Liang, 2013).

- The subsample size, denoted by $m$, is not necessarily very large, as long as the dependent structure of the data can be captured by the subsamples. If $m = n$, then MMLE is reduced to the MLE.

- For some problems, subsampling is necessary even if the entire data can be stored in the memory of a single computer, as evaluation of the function $H(\cdot)$ for all data can be extremely time consuming or even impossible.
RSA Algorithm: Remarks

- The asymptotic normality of $\theta_t$ can hold in general, even under infill asymptotics (Stein, 1999) where data are collected by sampling more and more densely in a fixed domain. This result is important, as it provides a theoretical foundation for hypothesis testing based on the MMLE method.

- RSA is also closely related to bootstrap. For dependent data, blocking bootstrap methods have been extensively studied under expanding-domain asymptotics (e.g., Hall, 1985), which attempts to retain the dependence structure of the data under some mixing conditions for the process. However, we are not aware of any other results demonstrating the validity of bootstrapping under infill asymptotics. RSA provides a way for parameter estimation for big data under infill asymptotics.
Consider a Gaussian geostatistical model,

\[ Y(s_i) = \mu(s_i) + X(s_i) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} N(0, \tau^2), \quad (4) \]

where \( Y(s_i) \) denotes the observation at location \( s_i \), \( \mu(s_i) \) denotes the mean of \( Y(s_i) \), \( \{X(s_i)\} \) denotes a spatial Gaussian process with \( E(X(s_i)) = 0 \), \( \text{Var}(X(s_i)) = \sigma^2 \), and \( \text{Corr}(X(s_i), X(s_j)) = \rho(\|s_i - s_j\|) \) for an appropriate correlation function with Euclidean distance \( \| \cdot \| \), and \( \tau^2 \) is referred to as the nugget variance in this context. The correlation function is chosen from a certain parametric family, such as the Matérn, exponential or spherical covariance models (Cressie, 1993).
RSA Algorithm: Simulated Example

Under model (4), \( \{ Y(s) \} \) follows a multivariate Gaussian distribution as follows:

\[
\begin{bmatrix} Y(s_1), \ldots, Y(s_n) \end{bmatrix}^T \sim N(\mu, \Sigma),
\]

where \( \mu = (\mu(s_1), \ldots, \mu(s_n))^T \), \( \Sigma = \sigma^2 R + \tau^2 I \), \( I \) is the \( n \times n \) identity matrix, and \( R \) is an \( n \times n \) correlation matrix with the \((i,j)\)th element given by \( \rho(\|s_i - s_j\|) \).

Model (4) is perhaps the most popular model in geostatistics, and it can be easily extended to the regression setting with the mean \( \mu(s_i) \) being replaced by \( \mu(s_i) = \beta_0 + \sum_{j=1}^p \beta_j c_j(s_i) \), where \( c_j(\cdot) \) denotes the \( j \)th explanatory variable, and \( \beta_j \) denotes the corresponding regression coefficient.
RSA Algorithm: Simulated Example

Difficulties with the Gaussian geostatistical model:

- Evaluation of the likelihood function of model (4) involves inverting the matrix $\Sigma$. This is infeasible when $n$ is large, because the complexity of matrix inversion increases as $O(n^3)$.
- The estimator may be inconsistent due to the existence of equivalent probability measures for the Gaussian process (Stein, 2004; Zhang, 2004) under infill asymptotics.
### RSA Algorithm: Simulated Example

Table 1. A comparison of RSA with MLE and the Gaussian predictive process method (Banerjee et al., 2008) for 50 simulated datasets with nugget effect.

<table>
<thead>
<tr>
<th>Estimator</th>
<th>Setting</th>
<th>$\hat{\beta}_0$</th>
<th>$\hat{\beta}_1$</th>
<th>$\hat{\phi}/\hat{\sigma}^2$</th>
<th>$\hat{\tau}^2$</th>
<th>CPU(m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>—</td>
<td>1.000</td>
<td>1.000</td>
<td>25.000</td>
<td>1.0</td>
<td>—</td>
</tr>
<tr>
<td>$n = 2000$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>$m = 300$</td>
<td>1.016(0.065)</td>
<td>1.000(0.007)</td>
<td>22.046(0.684)</td>
<td>0.974(0.009)</td>
<td>6.4</td>
</tr>
<tr>
<td></td>
<td>$m = 500$</td>
<td>1.013(0.064)</td>
<td>1.001(0.007)</td>
<td>23.084(0.675)</td>
<td>0.977(0.008)</td>
<td>29.3</td>
</tr>
<tr>
<td></td>
<td>$m = 700$</td>
<td>0.997(0.063)</td>
<td>0.999(0.006)</td>
<td>24.023(0.659)</td>
<td>0.993(0.007)</td>
<td>81.5</td>
</tr>
<tr>
<td>GPP</td>
<td>$6 \times 6$</td>
<td>1.031(0.075)</td>
<td>0.998(0.008)</td>
<td>35.199(6.958)</td>
<td>1.809(0.045)</td>
<td>111.2</td>
</tr>
<tr>
<td></td>
<td>$10 \times 10$</td>
<td>1.031(0.075)</td>
<td>0.998(0.008)</td>
<td>21.212(4.683)</td>
<td>1.773(0.043)</td>
<td>344.0</td>
</tr>
<tr>
<td>MLE</td>
<td>—</td>
<td>1.000(0.061)</td>
<td>1.000(0.006)</td>
<td>25.269(0.72)</td>
<td>0.999(0.007)</td>
<td>19.4</td>
</tr>
<tr>
<td>$n = 50000$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>$m = 300$</td>
<td>1.036(0.047)</td>
<td>1.004(0.004)</td>
<td>21.327(0.633)</td>
<td>0.970(0.006)</td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td>$m = 500$</td>
<td>1.033(0.045)</td>
<td>1.002(0.004)</td>
<td>22.204(0.531)</td>
<td>0.976(0.006)</td>
<td>30.6</td>
</tr>
<tr>
<td></td>
<td>$m = 700$</td>
<td>1.006(0.045)</td>
<td>0.999(0.002)</td>
<td>23.494(0.677)</td>
<td>0.989(0.004)</td>
<td>82.4</td>
</tr>
</tbody>
</table>
RSA Algorithm: Simulated Example

Figure: Comparison of MLE and RSA estimates for the simulated example. The horizontal axis shows the MLE and the vertical axis shows the RSA estimates obtained with $m = 700$. 
RSA Algorithm: precipitation data

RSA has also been tested on the US precipitation data, the precipitation anomaly of April 1948, from the National Climatic Data Center.

This dataset consisted of 11,918 observations, where a random subsample of 11,000 observations was used for model training and the remaining 918 observations were used for prediction.
RSA Algorithm: precipitation data

Figure: Figure 2: Observed and predicted precipitation anomaly for April 1948. (a) Observed image. (b) Predicted image based on an RSA estimate produced with $m = 500$. (c) Predicted image based on the MLE.
RSA Algorithm: CPU comparison for the Real Data example

- Figure 2 shows that the prediction produced by RSA is even more accurate than that by MLE.
- RSA cost about 29.6 minutes on a 3.0GHz personal computer.
- On the same computer, the MLE costs about 10,340 minutes, about 349 times longer than does RSA!
Bootstrap MH Algorithm: Motivation

- MCMC methods have been widely used in statistical data analysis, and they have proven to be a very powerful and typically unique computational tool for analyzing data of complex structures.
- MCMC methods are computer-intensive, which typically require a large number of iterations and a complete scan of the full dataset for each iteration. This feature precludes their use for big data analysis.
- We aim to develop a framework under which the powerful MCMC methods can be tamed to be used for big data analysis, such as parameter estimation, optimization, and model selection.
Bootstrap MH Algorithm: Basic Idea

- The bootstrap Metropolis-Hastings (BMH) algorithm works by replacing the full data log-likelihood by a Monte Carlo average of the log-likelihoods that are calculated in parallel from multiple bootstrap samples, where the bootstrap sample refers to a small set of observations drawn from the full dataset at random and with/without replacement.

- By this way, BMH avoids repeated scans of the full dataset in iterations, while it is still able to produce sensible solutions, such as parameter estimates or posterior samples, to the problem under consideration. BMH is feasible for big data and workable on parallel and distributed architectures.
BMH Algorithm: Notation

- Let \( D_i \) denote a bootstrap sample of \( \mathcal{D} \), which is resampled from the full dataset at random and with/without replacement.
- Let \( m \) denote the size of \( D_i = \{ x_{ij}^* : j = 1, 2, \ldots, m \} \). If resampling is done without replacement, \( D_i \) is called a subsample or \( \binom{n}{m} \)-bootstrap sample. Otherwise, \( D_i \) is called an \( m \)-out-of-\( n \) bootstrap sample or \( m/n \)-bootstrap sample.
BMH Algorithm: Notation

- Let \( \tilde{f}(D_i|\theta) \) denote a likelihood-like function of \( D_i \), and define

\[
l_{m,n,k}(D_s|\theta) = \frac{1}{k} \sum_{i=1}^{k} \log \tilde{f}(D_i|\theta), \tag{6}\]

where \( k \) denotes the number of bootstrap samples drawn from \( D \), and \( D_s = \{D_1, \ldots, D_k\} \) is the collection of the bootstrap samples.

- The definition of \( \tilde{f}(D_i|\theta) \) depends on the feature of \( D \). If the observations in \( D \) are independently and identically distributed (i.i.d.), then, regardless \( D_i \) is a \( \binom{n}{m} \)- or \( m/n \)-bootstrap sample, we define

\[
\tilde{f}(D_i|\theta) = \prod_{j=1}^{m} f(x_{ij}^*|\theta). \tag{7}\]
BMH Algorithm: Algorithm

1. Draw $\vartheta$ from a proposal distribution $Q(\theta_t, \vartheta)$.

2. Draw $k$ bootstrap samples $D_1, \ldots, D_k$ via $(n \choose m)$- or $m/n$-bootstrapping. Let $D_s = \{D_1, \ldots, D_k\}$.

3. Calculate the BMH ratio:

$$r(\theta_t, D_s, \vartheta) = \exp \{l_{m,n,k}(D_s|\vartheta) - l_{m,n,k}(D_s|\theta_t)\} \frac{\pi(\vartheta) Q(\vartheta, \theta_t)}{\pi(\theta_t) Q(\theta_t, \vartheta)}.$$

4. Set $\theta_{t+1} = \vartheta$ with probability

$$\alpha(\theta_t, D_s, \vartheta) = \min\{1, r(\theta_t, D_s, \vartheta)\},$$

and set $\theta_{t+1} = \theta_t$ with the remaining probability.
BMH Algorithm: Remarks

• In BMH, \( \{\theta_t\} \) form a Markov chain with the transition kernel given by

\[
P_{m,n,k}(\theta, d\vartheta) = \sum_{D_s\in\mathbb{D}} \alpha(\theta, D_s, \vartheta) Q(\theta, \vartheta) \psi(D_s) \\
+ \delta_\theta(d\vartheta) \left[ 1 - \sum_{D'_s\in\mathbb{D}} \int_{\Theta} \alpha(\theta, D'_s, \vartheta') Q(\theta, d\vartheta') \psi(D'_s) \right]
\]

(8)

where \( \mathbb{D} \) denote the space of \( D_s \), \( \psi(D_s) \) denotes the probability of drawing \( D_s \), and \( \delta_\theta(\cdot) \) is an indicator function.

For \( \binom{n}{m} \)-bootstrapping, \( \psi(D_s) = \binom{n}{m}^{-k} \); and for \( m/n \)-bootstrapping, \( \psi(D_s) = 1/n^{mk} \).
BMH Algorithm: Remarks

- When the observations in $\mathcal{D}$ are i.i.d, both the resampling schemes, $\binom{n}{m}$- or $m/n$-bootstrapping lead to the same stationary distribution of BMH.

- Since BMH is proposed for simulations on parallel computers, the parameter $k$ specifies the number of processors/nodes used in computing the averaged log-likelihood function. Theoretically, a large value of $k$ is preferred. However, an extremely large value of $k$ may slow down the computation due to the increased inter-node communications. In our experience, to achieve a good performance for BMH, $k$ does not need to be very large.

- The choice of $m$ can depend on the complexity of the model under consideration, in particular, the dimension of $\theta$. In general, $m$ should increase with the complexity of the model.
BMH Algorithm: Convergence

• Let $g_m(\mathcal{D}|\theta) = \exp\{E[\log \tilde{f}(D_i|\theta)]\}$, where $E[\cdot]$ denotes the expectation.

• Define the transition kernel

$$P_m(\theta, \vartheta) = \alpha(\theta, \vartheta)Q(\theta, \vartheta) + \delta_\theta(d\vartheta) \left[1 - \int_{\Theta} \alpha(\theta, \vartheta')Q(\theta, \vartheta')d\vartheta'\right], \quad (9)$$

which is induced by the proposal $Q(\cdot, \cdot)$ for a MH move with the invariant distribution given by

$$\tilde{\pi}_m(\theta|\mathcal{D}) \propto g_m(\mathcal{D}|\theta)\pi(\theta). \quad (10)$$
BMH Algorithm: Convergence

Assume the following conditions hold:

(A) \( \sup_{\theta \in \Theta} E|\log \tilde{f}(D_i|\theta)| < \infty. \)

(B) Assume that \( P_m \) defines an irreducible and aperiodic Markov chain such that \( \tilde{\pi}_m(\cdot)P_m = \tilde{\pi}_m(\cdot). \) Therefore, for any starting point \( \theta_0 \in \Theta, \lim_{t \to \infty} \|P_m^t(\theta_0, \cdot) - \tilde{\pi}_m(\cdot)\| = 0, \) where \( \| \cdot \| \) denotes the total variation norm.

(C) For any \((\theta, \vartheta) \in \Theta \times \Theta, \)

\[
0 < \exp\{l_{m,n,k}(D_s|\vartheta) - l_{m,n,k}(D_s|\theta)\}/[g_m(D|\vartheta) - g_m(D|\theta)] < \infty, \quad \psi(D_s)
\]

where \( \psi(D_s) \) is the resampling probability of \( D_s \) from \( D. \)
BMH Algorithm: Convergence

**Theorem 1.** \((\binom{n}{m}\text{-bootstrapping})\) Assume the observations in \(D\) are iid and the conditions \((A), (B)\) and \((C)\) hold. Then for any \(\epsilon \in (0, 1]\) and any \(\theta_0 \in \Theta\), there exist \(N(\epsilon, \theta_0) \in \mathbb{N}\), \(K(\epsilon, \theta_0, n) \in \mathbb{N}\), and \(T(\epsilon, \theta_0, n, k) \in \mathbb{N}\) such that for any \(n > N(\epsilon, \theta_0), k > K(\epsilon, \theta_0, n),\) and \(t > T(\epsilon, \theta_0, n, k),\)

\[
\| P_{m,n,k}^t(\theta_0, \cdot) - \tilde{\pi}_m(\cdot) \| \leq \epsilon,
\]

where \(\tilde{\pi}_m(\cdot)\) is the stationary distribution of \(P_m\) as defined in (10).

**Theorem 2.** \((m/n\text{-bootstrapping})\) Under similar conditions to Theorem 1, BMH with \(m/n\)-bootstrapping has the same stationary distribution as with \(\binom{n}{m}\)-bootstrapping.
BMH Algorithm: Bayesian Inference

Some key points:

- Rewrite the full data posterior $\pi(\theta|D)$ by $\pi_n(\theta|D)$; i.e.,
  $$\pi_n(\theta|D) \propto \pi_n(\theta)L(D|\theta),$$
  where $L(D|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$ denotes the likelihood function of $D$, and $\pi_n(\theta)$ denotes the prior of $\theta$ which may depend on the value of $n$.

- For any fixed value of $m$ and for any $\theta \in \Theta$,
  $$[L(D|\theta)]^{m/n} \overset{a.s.}{\rightarrow} g_m(D|\theta), \quad \text{as } n \rightarrow \infty. \quad (11)$$

Choose prior such that
  $$\pi_m(\theta|D) \propto [\pi_n(\theta)]^{m/n} [L(D|\theta)]^{m/n}.$$  

Then the asymptotic normality of $\pi_m(\theta|D)$ implies the normality of $\tilde{\pi}_m(\theta|D)$. 
BMH Algorithm: Bayesian Inference

Some key points (continued):

- It follows from the asymptotic normality of posterior distributions, see e.g., Chen (1985), we have

\[
\pi_n(\theta|D) \xrightarrow{L} N(\mu_n, \Sigma_n),
\]

where \(\mu_n\) denotes the mode of \(\pi_n(\theta|D)\) and

\[
\Sigma_n = \left\{ -\partial^2 (\pi_n(\theta)L(D|\theta))/\partial\theta\partial\theta^T \right\}^{-1}.
\]

Therefore,

\[
\pi_m(\theta|D) \xrightarrow{L} N \left( \mu_n, \frac{n}{m}\Sigma_n \right),
\]

provided that \(m\) is reasonably large.

- The properties of \(\pi_n(\theta|D)\) can be conveniently inferred from BMH samples.
Simulated Example

Consider the problem of estimating the regression coefficients of a logistic regression with five predictors $X_1, \ldots, X_5$. Let $Y_i$ be the binary response and let $Z_i = (1, X_1^{(i)}, \ldots, X_5^{(i)})^T$, where $X_k^{(i)}$ denotes the $i$th realization of $X_k$. Then the logistic regression model can be written as

$$P(Y_i = 1) = \frac{\exp(Z_i^T \beta)}{1 + \exp(Z_i^T \beta)}, \quad i = 1, \ldots, n,$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_5)^T$ is the regression coefficient vector, and $n$ is the total number of observations. In simulations, we set $\beta = (1, 2, 3, 4, 5, 6)^T$ and $n = 10^6$. The predictors are drawn independently from the standard normal distribution.
### Simulated Example

Table 1. Parameter estimation results of BMH with $m/n$-bootstrapping for the simulated example. The “Resampling Rate” is calculated as $m/n \times 100\%$.

<table>
<thead>
<tr>
<th>$(k, m)$</th>
<th>Resampling Rate</th>
<th>$\hat{\beta}_0$</th>
<th>$\hat{\beta}_1$</th>
<th>$\hat{\beta}_2$</th>
<th>$\hat{\beta}_3$</th>
<th>$\hat{\beta}_4$</th>
<th>$\hat{\beta}_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20,200)</td>
<td>0.02%</td>
<td>1.0094</td>
<td>2.0042</td>
<td>3.0100</td>
<td>4.0096</td>
<td>5.0071</td>
<td>6.0134</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0026)</td>
<td>(0.0047)</td>
<td>(0.0067)</td>
<td>(0.0095)</td>
<td>(0.0111)</td>
<td>(0.0139)</td>
</tr>
<tr>
<td>(20,500)</td>
<td>0.05%</td>
<td>1.0057</td>
<td>1.9967</td>
<td>3.0001</td>
<td>3.9999</td>
<td>4.9940</td>
<td>5.9984</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0017)</td>
<td>(0.0030)</td>
<td>(0.0045)</td>
<td>(0.0059)</td>
<td>(0.0071)</td>
<td>(0.0084)</td>
</tr>
<tr>
<td>(20,1000)</td>
<td>0.1%</td>
<td>1.0078</td>
<td>2.0011</td>
<td>3.0056</td>
<td>4.0051</td>
<td>5.0031</td>
<td>6.0089</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0012)</td>
<td>(0.0021)</td>
<td>(0.0033)</td>
<td>(0.0048)</td>
<td>(0.0063)</td>
<td>(0.0069)</td>
</tr>
<tr>
<td>(50,200)</td>
<td>0.02%</td>
<td>1.0029</td>
<td>1.9923</td>
<td>2.9900</td>
<td>3.9872</td>
<td>4.9814</td>
<td>5.9803</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0016)</td>
<td>(0.0026)</td>
<td>(0.0042)</td>
<td>(0.0054)</td>
<td>(0.0065)</td>
<td>(0.0081)</td>
</tr>
<tr>
<td>(50,500)</td>
<td>0.05%</td>
<td>1.0083</td>
<td>2.0009</td>
<td>3.0030</td>
<td>4.0052</td>
<td>5.0015</td>
<td>6.0057</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0011)</td>
<td>(0.0023)</td>
<td>(0.0035)</td>
<td>(0.0043)</td>
<td>(0.0053)</td>
<td>(0.0062)</td>
</tr>
<tr>
<td>(50,1000)</td>
<td>0.1%</td>
<td>1.0055</td>
<td>1.9955</td>
<td>2.9957</td>
<td>3.9940</td>
<td>4.9885</td>
<td>5.9901</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0013)</td>
<td>(0.0017)</td>
<td>(0.0026)</td>
<td>(0.0035)</td>
<td>(0.0044)</td>
<td>(0.0051)</td>
</tr>
</tbody>
</table>
### Simulated Example

Table 2. Parameter estimation results of BMH with $\binom{n}{m}$-bootstrapping for the simulated example. The “Resampling Rate” is calculated as $m/n \times 100\%$.

<table>
<thead>
<tr>
<th>$(k, m)$</th>
<th>Resampling Rate</th>
<th>$\hat{\beta}_0$</th>
<th>$\hat{\beta}_1$</th>
<th>$\hat{\beta}_2$</th>
<th>$\hat{\beta}_3$</th>
<th>$\hat{\beta}_4$</th>
<th>$\hat{\beta}_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(20, 200)$</td>
<td>0.02%</td>
<td>0.9991</td>
<td>1.9866</td>
<td>2.9817</td>
<td>3.9763</td>
<td>4.9669</td>
<td>5.9658</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0025)</td>
<td>(0.0037)</td>
<td>(0.0051)</td>
<td>(0.0070)</td>
<td>(0.0093)</td>
<td>(0.0110)</td>
</tr>
<tr>
<td>$(20, 500)$</td>
<td>0.05%</td>
<td>1.0071</td>
<td>2.0002</td>
<td>3.0027</td>
<td>4.0033</td>
<td>4.9993</td>
<td>6.0028</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0021)</td>
<td>(0.0041)</td>
<td>(0.0063)</td>
<td>(0.0082)</td>
<td>(0.0102)</td>
<td>(0.0120)</td>
</tr>
<tr>
<td>$(20, 1000)$</td>
<td>0.1%</td>
<td>1.0090</td>
<td>2.0024</td>
<td>3.0063</td>
<td>4.0090</td>
<td>5.0065</td>
<td>6.0114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0015)</td>
<td>(0.0029)</td>
<td>(0.0044)</td>
<td>(0.0059)</td>
<td>(0.0073)</td>
<td>(0.0086)</td>
</tr>
<tr>
<td>$(50, 200)$</td>
<td>0.02%</td>
<td>0.9998</td>
<td>1.9849</td>
<td>2.9806</td>
<td>3.9746</td>
<td>4.9635</td>
<td>5.9620</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0019)</td>
<td>(0.0036)</td>
<td>(0.0054)</td>
<td>(0.0072)</td>
<td>(0.0088)</td>
<td>(0.0106)</td>
</tr>
<tr>
<td>$(50, 500)$</td>
<td>0.05%</td>
<td>1.0049</td>
<td>1.9941</td>
<td>2.9955</td>
<td>3.9932</td>
<td>4.9871</td>
<td>5.9896</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0012)</td>
<td>(0.0021)</td>
<td>(0.0034)</td>
<td>(0.0041)</td>
<td>(0.0051)</td>
<td>(0.0063)</td>
</tr>
<tr>
<td>$(50, 1000)$</td>
<td>0.1%</td>
<td>1.0058</td>
<td>1.9967</td>
<td>2.9988</td>
<td>3.9967</td>
<td>4.9915</td>
<td>5.9936</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.0013)</td>
<td>(0.0017)</td>
<td>(0.0029)</td>
<td>(0.0042)</td>
<td>(0.0052)</td>
<td>(0.0060)</td>
</tr>
</tbody>
</table>
Simulated Example

Table 3. Comparison of BMH \((k = 50, \; m = 500, \; 40 \; \text{runs})\) with the divide-and-combine (D&C) method \((40 \; \text{subsets}, \; 25,000 \; \text{observations per subset})\).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Estimate</th>
<th>SD</th>
<th>Estimate</th>
<th>SD</th>
<th>Estimate</th>
<th>SD</th>
<th>Estimate</th>
<th>SD</th>
<th>Estimate</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\beta_0)</td>
<td></td>
<td>(\beta_1)</td>
<td></td>
<td>(\beta_2)</td>
<td></td>
<td>(\beta_3)</td>
<td></td>
<td>(\beta_4)</td>
<td></td>
</tr>
<tr>
<td>D&amp;C</td>
<td>1.0064</td>
<td>0.0051</td>
<td>1.9973</td>
<td>0.0072</td>
<td>2.9991</td>
<td>0.0094</td>
<td>3.9985</td>
<td>0.0113</td>
<td>4.9936</td>
<td>0.0127</td>
</tr>
<tr>
<td>MSE((\times 10^{-4}))</td>
<td>10.71</td>
<td>20.27</td>
<td>34.75</td>
<td>49.62</td>
<td>63.10</td>
<td>92.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD((\times 10^{-4}))</td>
<td>2.68</td>
<td>3.95</td>
<td>7.52</td>
<td>11.49</td>
<td>14.71</td>
<td>22.17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMH</td>
<td>1.0056</td>
<td>0.0007</td>
<td>1.9960</td>
<td>0.0013</td>
<td>2.9973</td>
<td>0.0019</td>
<td>3.9959</td>
<td>0.0026</td>
<td>4.9909</td>
<td>0.0032</td>
</tr>
<tr>
<td>MSE((\times 10^{-4}))</td>
<td>0.60</td>
<td>1.11</td>
<td>2.18</td>
<td>3.88</td>
<td>6.47</td>
<td>8.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD((\times 10^{-4}))</td>
<td>0.09</td>
<td>0.15</td>
<td>0.28</td>
<td>0.55</td>
<td>0.96</td>
<td>1.17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simulated Example

Figure: Histograms of the posterior samples collected in 40 runs of D&C [plot (a)], 40 runs of BMH [plot (b)], a single run of D&C [plot (c)], and a single run of BMH [plot (d)] for the parameter $\beta_5$, whose true value is 6.
Comments on BMH

- It can asymptotically integrate the whole data information into a single simulation run.
- Like the MH algorithm, it can serve as a basic building block for developing advanced MCMC algorithms.
Universal approximation ability: A feed-forward network with a single hidden layer containing a finite number of hidden units is a universal approximator among continuous functions on compact subsets, under mild assumptions on the activation function.

It is potentially a good tool for big data modeling!
Neural Network

**Figure:** A fully connected one hidden layer MLP network.
Tempering BMH for Learning BNN with Big Data

1. Draw $k$ bootstrap samples, $D_1, \ldots, D_k$, with/without replacement from the entire training dataset $\mathcal{D}$.

2. Try to update each sample of the current population $(\theta^1_t, \ldots, \theta^\Pi_t)$ by the local updating operators, where $t$ indexes iterations, and the energy function is calculated by averaging the energy values calculated from the bootstrap samples $D_1, \ldots, D_k$.

3. Try to exchange $\theta^i_t$ with $\theta^j_t$ for $n - 1$ pairs $(i, j)$ with $i$ being sampled uniformly on $\{1, \ldots, n\}$ and $j = i \pm 1$ with probability $\omega_{i,j}$, where $\omega_{i,i+1} = \omega_{i,i-1} = 0.5$ for $1 < i < \Pi$ and $\omega_{1,2} = \omega_{\Pi,\Pi-1} = 1$. 
Tempering BMH for Learning BNN with Big Data

**Figure:** Parallel implementation of tempering BMH: The flowchart of the tempering BMH algorithm with 3 processors.
Forest Cover Type Data

The goal of this study is to predict forest cover types from cartographic variables in the forested areas with minimal human-caused disturbances. The data were taken from four wilderness areas located in the Roosevelt National Forest of northern Colorado. It consisted of 581,012 observations. Each observation was obtained from the US Geological Survey (USGS) digital elevation model data based on $30 \times 30$-m raster cells, and it consisted of 54 cartographic explanatory variables including 10 quantitative variables, 4 binary wilderness area variables, and 40 binary soil type variables. These observations have been classified into seven classes according to their cover types. The respective class sizes are 211840, 283301, 35754, 2747, 9493, 17367, and 20510.
Table 4. BMH results for forest cover type data. The resampling rates for the 7 types of observations are 0.5%, 0.5%, 1%, 2.5%, 1.5%, 1% and 1%, respectively. The aggregated resampling rate for the training data is 0.59%.

<table>
<thead>
<tr>
<th>Bootstrapping</th>
<th>$k$</th>
<th>Average network Size</th>
<th>Training rate(%)</th>
<th>Prediction rate(%)</th>
<th>CPU(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m/n$</td>
<td>20</td>
<td>259.0 (1.18)</td>
<td>72.2 (0.17)</td>
<td>72.4 (0.17)</td>
<td>32.0 (2.9)</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>259.0 (0.95)</td>
<td>72.2 (0.15)</td>
<td>72.4 (0.12)</td>
<td>33.7 (3.2)</td>
</tr>
<tr>
<td>$(n \choose m)$</td>
<td>20</td>
<td>260.0 (1.52)</td>
<td>72.3 (0.07)</td>
<td>72.4 (0.07)</td>
<td>28.5 (2.5)</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>263.5 (0.83)</td>
<td>72.3 (0.15)</td>
<td>72.4 (0.16)</td>
<td>31.9 (3.0)</td>
</tr>
</tbody>
</table>
For comparison, we have applied parallel tempering to train the BNN with a single-threaded simulation on an Intel Nehalem server. At each local updating step of parallel tempering, the whole training dataset is scanned once. Hence, this algorithm runs extremely slow.

The first 5536 iterations of the simulation have taken 688 CPU hours, although the Intel Nehalem processor is much faster (approximately 2.5 times) than the processor used in the cluster machine. To finish 25000 iterations, it will take about 3000 CPU hours (125 days) on the Intel Nehalem server. Compared 3000 hours to 30 hours, it shows a great advantage of the parallelized BMH algorithm for big data problems.
Resampling on Distributed Architectures

Let $S_i$ denote the $i$th subset of data stored in node $i$, $i = 1, \ldots, k$. For each $i$,

1. Set $j = i - 1$ or $i + 1$ with equal probability. If $j = 0$, reset $j = k$; and if $j = k + 1$, reset $j = 1$.

2. Exchange $M$ randomly selected observations between $S_i$ and $S_j$, where $M$ can be a pre-specified or random number.

It follows from the standard theory of MCMC (see e.g. Geyer, 1991) that the above procedure will ensure that each subset stored in a single node is a random subset of the whole dataset.
Discussion

• Both the RSA and BMH algorithms are workable on parallel machines and avoid repeated scans of the full dataset in iterations, and are thus feasible for big data problems.

• RSA provides a general framework for parameter estimation for big data problems. In addition to the simple random sampling scheme used for generating subsamples, other sampling schemes, such as stratified sampling, also work.
Discussion (continued)

• BMH tames the powerful MCMC methods to be used for big data analysis, and can serve as the basic building block for developing advanced MCMC algorithms that are feasible for big data problems.

  • Sampling: Tempering BMH, which combines BMH with parallel tempering.
  • Model selection: Reversible jump BMH, which combines BMH with reversible jump MCMC.
  • Optimization: Simulated annealing BMH, which combines BMH with simulated annealing.

• Compared to the popular divide-and-combine method, BMH is generally more efficient as it can asymptotically integrate the whole data information into a single simulation run.

• RSA can also asymptotically integrate the whole data information into the parameter estimator.
Acknowledgments

- NSF grants
- KAUST grant
- Students:
  - Yichen Cheng
  - Jinsu Kim
  - Jincheol Park
  - Qifan Song