Parallelizing MCMC with Random Partition Trees

Xiangyu Wang*, Richard (Fangjian) Guo†, Katherine A. Heller†*, David B. Dunson*

*Department of Statistical Science, †Department of Computer Science, Duke University

http://bit.ly/1RdhuBW

Embarrassingly Parallel MCMC

1. Partition data into m subsets
2. Run MCMC **independently in parallel** on each subset with your favorite algorithm
3. Aggregate the sub-chains (sub-posteriors) by some algorithm

PART Algorithm

PART relies on the *product density equation (PDE)*. Assuming X is the observed data and θ is the parameter of interest, when the observations are iid conditioned on θ, for any partition of $X = X^{(1)} \cup X^{(2)} \cup \cdots \cup X^{(m)}$, then

$$p(\theta|X) \propto \pi(\theta)p(X|\theta) \propto p(\theta|X^{(1)})p(\theta|X^{(2)}) \cdots p(\theta|X^{(m)}), \quad (1)$$

if the prior on the full data and subsets satisfy $\pi(\theta) = \prod_{i=1}^m \pi_i(\theta)$.

Introduction

The modern scale of data has brought new challenges to Bayesian inference. In particular, conventional MCMC algorithms are computationally very expensive for large data sets. A promising approach is the embarrassingly parallel MCMC (EP-MCMC).

Existing EP-MCMC algorithms are limited by approximation accuracy and difficulty in resampling. Here we propose a new EP-MCMC algorithm **PART** that solves these problems. The new algorithm applies **random partition trees** to combine the subset posterior draws, which is distribution-free, easy to resample from and can adapt to multiple scales.

Features of PART

- Fast sub-posterior density estimation
- Efficient density aggregation
- Efficient resampling

Related Works

- **Simple averaging** and **Weighted averaging** (**Consensus Monte Carlo** [1]): weights are optimally chosen for a Gaussian posterior.
- **Weierstrass rejection sampler**: subset posterior samples are passed through a rejection sampler based on the Weierstrass transform [2].
- **Parametric density product**: a product of Laplacian approximations to sub-posteriors [3].
- **Nonparametric/Semiparametric density product**: a product of kernel density estimates (or its semiparametric variant) for subset posteriors [3], which is sampled with an independent Metropolis chain.

Step 1: Space Partitioning

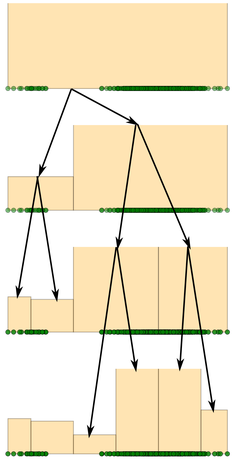
Since the full data posterior is $p(\theta|X) \propto \pi(\theta)\prod_{i=1}^m p(X^{(i)}|\theta) \propto \prod_{i=1}^m f^{(i)}(\theta)$, We use a **random partition tree** or **multi-scale histogram** to estimate each sub-posterior density $f^{(i)}(\theta)$.

Let $\mathcal{F}_\mathcal{K}$ be the collection of all $\mathbb{R}^p$-partitions formed by K disjoint rectangular blocks, where a rectangular block takes the form of $A_k{=}(l_{k,1}, r_{k,1}] \times (l_{k,2}, r_{k,2}] \times \cdots (l_{k,p}, r_{k,p}] \subseteq \mathbb{R}^p$ for some $l_{k,q} < r_{k,q}$. A K-block histogram is then defined as

$$\hat{f}^{(i)}(\theta) = \sum_{k=1}^K \frac{n_k^{(i)}}{N|A_k|}\mathbf{1}(\theta \in A_k), \quad (2)$$

where $\{A_k : k = 1, 2, \cdots, K\} \in \mathcal{F}_\mathcal{K}$ are the blocks and $N, n_k^{(i)}$ are the total number of posterior samples on the i^{th} subset and of those inside the block A_k respectively (assuming the same N across subsets).

A random partition tree is built like a decision tree by recursively (1) randomly selecting a dimension and (2) choosing an optimal cutting point by $\phi(\theta)$ to bisect that dimension, until the probability or area of a block drops below a threshold.



We consider two choices of the cutting point.

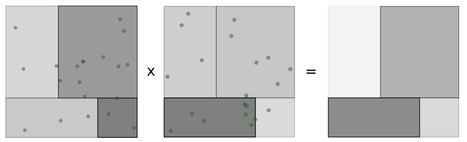
ML-tree Picking the point that maximizes the empirical likelihood.

- # of blocks adapted to complexity.
- Search in $O(n \log n)$.

KD-tree Picking the median.

- Split into two blocks with equal probability.
- $O(\log n)$

Step 2: Density Aggregation



By constraining all trees to take **the same** partitioning $\{A_k\}$, the aggregated density is still a tree with the same structure. It can be computed by block-wise multiplication and a renormalization in $O(MK)$.

$$\hat{p}(\theta|X) = \frac{1}{Z}\prod_{i=1}^m \hat{f}^{(i)}(\theta) = \frac{1}{Z}\sum_{k=1}^K \left(\prod_{i=1}^m \frac{n_k^{(i)}}{|A_k|}\right)\mathbf{1}(\theta \in A_k) = \sum_{k=1}^K w_k g_k(\theta),$$

where $Z = \sum_{k=1}^K \prod_{i=1}^m n_k^{(i)}/|A_k|^{m-1}$ is the normalizing constant, w_k's are the updated weights, and $g_k(\theta) = \text{unif}(\theta; A_k)$ is the block-wise distribution.

Why not Kernel Density Estimate?

$O(n^m)$ mixture components — very slow mixing.

Variance Reduction & Smoothing

Random Tree Ensemble Inspired by random forests, the full posterior is estimated by averaging T independent trees, which are built in parallel. Resampling from the aggregated posterior consists of

1. choosing a tree uniformly at random
2. sampling a block k by weight w_k
3. sampling $\theta \sim g_k(\theta)$.

Local Gaussian Smoothing The blockwise uniform distribution can be replaced by a Gaussian distribution $g_k = N(\theta; \mu_k, \Sigma_k)$, with mean and covariance estimated "locally" by samples within the block. A multiplied Gaussian approximation is used: $\Sigma_k = (\sum_{i=1}^m \hat{\Sigma}_k^{(i)-1})^{-1}, \mu_k = \Sigma_k(\sum_{i=1}^m \hat{\Sigma}_k^{(i)-1}\hat{\mu}_k^{(i)})$, where $\hat{\Sigma}_k^{(i)}$ and $\hat{\mu}_k^{(i)}$ are estimated with the i^{th} subset.

Toy Example

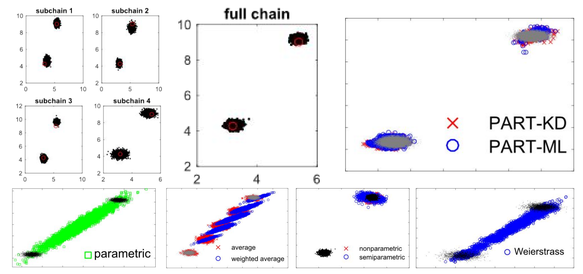
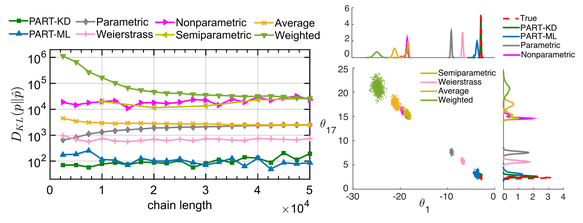


Figure 1: Inferring the mean of a mixture of two Gaussians.

Bayesian Logistic Regression

Synthetic Dataset $N = 50,000$ observations in $p = 50$ dimensions, split into $m = 40$ subsets. The posterior concentration ratio is defined as $r = \sqrt{\sum_j \|\hat{\boldsymbol{\theta}}_j - \boldsymbol{\theta}^*\|_2^2 / \sum_j \|\boldsymbol{\theta}_j - \boldsymbol{\theta}^*\|_2^2}$.

Method	RMSE	$D_{\text{KL}}(p\|\hat{p})$	$D_{\text{KL}}(\hat{p}\|p)$	r
PART (KD)	**0.587**	3.95×10^2	6.45×10^2	**3.94**
PART (ML)	1.399	$\mathbf{8.05 \times 10^1}$	$\mathbf{5.47 \times 10^2}$	9.17
average	29.93	2.53×10^3	5.41×10^4	184.62
weighted	38.28	2.60×10^4	2.53×10^5	236.15
Weierstrass	6.47	7.20×10^2	2.62×10^3	39.96
parametric	10.07	2.46×10^3	6.12×10^3	62.13
nonparametric	25.59	3.40×10^4	3.95×10^4	157.86
semiparametric	25.45	2.06×10^4	3.90×10^4	156.97



References

[1] Steven L Scott, Alexander W Blocker, Fernando V Bonassi, Hugh A Chipman, Edward I George, and Robert E McCulloch.
Bayes and big data: The consensus Monte Carlo algorithm.
In *EFaBBayes 250 conference*, volume 16, 2013.

[2] Xiangyu Wang and David B Dunson.
Parallel MCMC via Weierstrass sampler.
arXiv preprint arXiv:1312.4605, 2013.

[3] Willie Neiswanger, Chong Wang, and Eric Xing.
Asymptotically exact, embarrassingly parallel MCMC.
In *Proceedings of the Thirtieth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-14)*, 2014.