Distributed MCMC

Mattias Villani

Division of Statistics and Machine Learning
Department of Computer and Information Science
Linköping University

Lecture Overview

- Parallelism - potential and limitations
- Distributed MCMC

Parallelism - potential

- **Why**? Cores don't get much faster, but we get more of them.
- **MCMC is serial** - hard to parallelize.
- **Independence MH** can be parallelized by:
  1. In parallel: generate **all** proposal draws and evaluate posterior densities.
  2. Run Markov Chain on single core using pre-computed quantities from 1.
  3. Inject a random walk step every r:th step to avoid getting stuck.
- Graphic cards (**GPU**) have thousands of cores.
  Hours $\rightarrow$ Seconds in favorable cases. [1, 2, 3]
- **Conditional independence** can open up parallelism. [4]
- Special solutions: **Pre-fetching**. Predict the future state of the MCMC chain and pre-compute ahead of time. [5, 6]

Parallelism - limitations

- Need special code. **Harder to debug**.
- Graphic cards (**GPU**) requires very careful data management to be really effective. RAM/Shared/Constant/Registers. See the MCMC in [3]
- **Avoiding marginalization** $\Rightarrow$ Conditional independences $\Rightarrow$ Parallelism, but can slow down convergence.
- **Amdahl's law**. 'The theoretical speedup is always limited by the part of the task that cannot benefit from the improvement'.
- **Communication overhead** limits gains from parallelism. Asynchronous MCMC may help. [7]
- The data set may be **larger than RAM** memory.

Distributed MCMC

- **Map-Reduce** philosophy: bring the computations to the data.
- General idea:
  - **Split** the data across many machines.
  - **Run separate** MCMC chains on each machine. **Subposteriors**.
  - **Combine** the MCMC **draws** after the MCMC.
- Posterior and subposteriors from S machines/data subsets

$$p(\theta|\mathbf{y}) \propto \prod_{s=1}^{S} p(\mathbf{y}_s|\theta)p(\theta)^{1/S}$$

- The **subsets** $\mathbf{y}_1, ..., \mathbf{y}_S$ are assumed **conditionally independent**.
- $p(\theta) = \prod_{s=1}^{S} p(\theta)^{1/S}$ to preserve total prior information. Make sure that $p(\theta)^{1/S}$ is proper!
- **How to combine draws** from different subposteriors?

Consensus Monte Carlo

- **Assume** that each subposterior $p(\theta|\mathbf{y}_s)$ is $N(\mu_s, \Omega_s)$. Then

$$p(\theta|\mathbf{y}) \propto \prod_{s=1}^{S} p(\theta|\mathbf{y}_s) = N(\mu, \Omega)$$

where

$$\Omega^{-1} = \sum_{s=1}^{S} \Omega_s^{-1} \qquad \mu = \Omega \left(\sum_{s=1}^{S} \Omega_s^{-1} \mu_s \right)$$

- Scott et al. (2013) **[8]** therefore propose to take (matrix) **weighted averages of the subposterior draws**:

$$\theta^{(i)} = \left(\sum_{s=1}^{S} \Omega_s^{-1} \right)^{-1} \sum_{s=1}^{S} \Omega_s^{-1} \theta_s^{(i)}, \; i = 1, ..., N$$

- Check: $\mathbb{E}(\theta^{(i)}) = \mu$ and $\mathbb{C}(\theta^{(i)}) = \Omega$.

Consensus Monte Carlo, cont.

- The averaging of draws

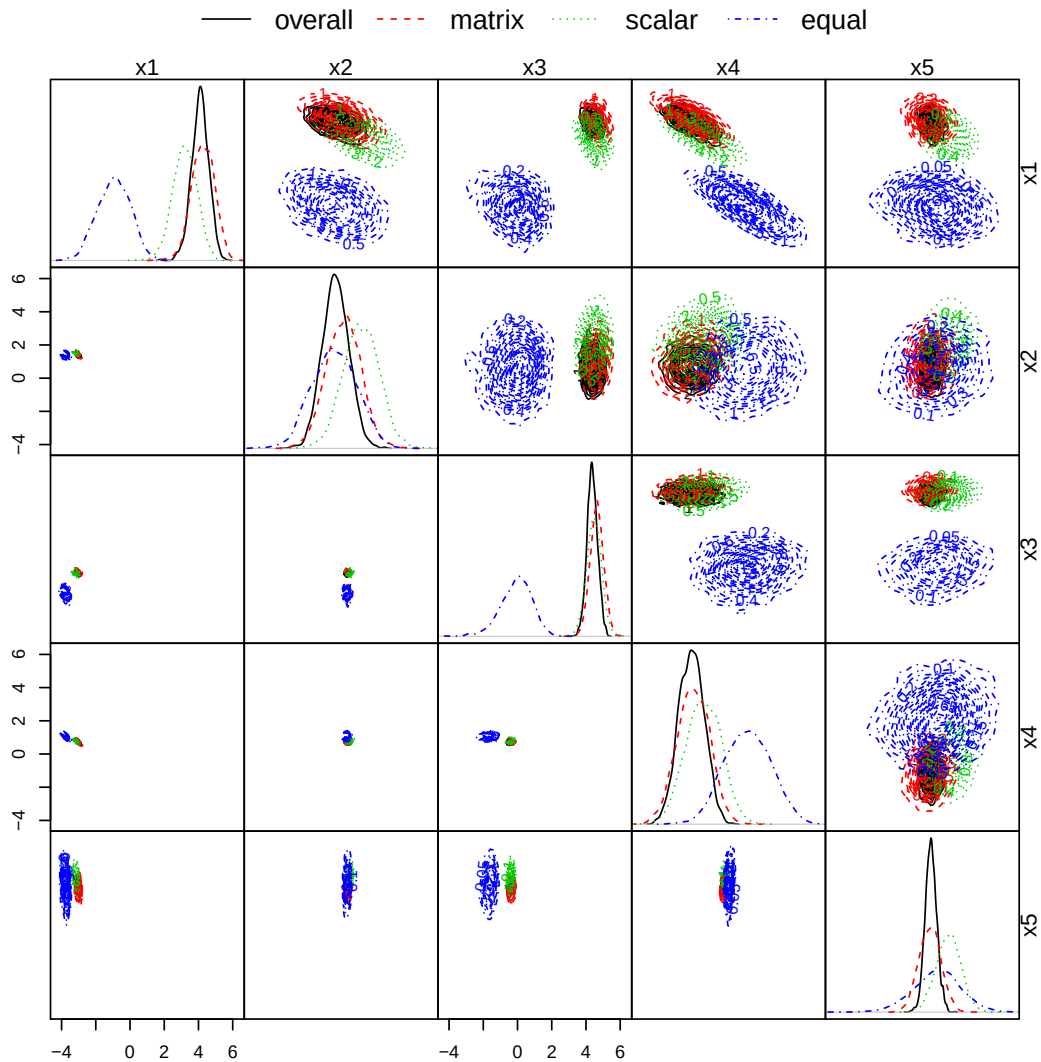
$$\theta^{(i)} = \left(\sum_{s=1}^{S} \Omega_s^{-1} \right)^{-1} \sum_{s=1}^{S} \Omega_s^{-1} \theta_s^{(i)}$$

  is **only formally correct when** each subposterior is normal.

- Posteriors are asymptotically normal (Bernstein-von Mises), but note that it need to hold for each subposterior.

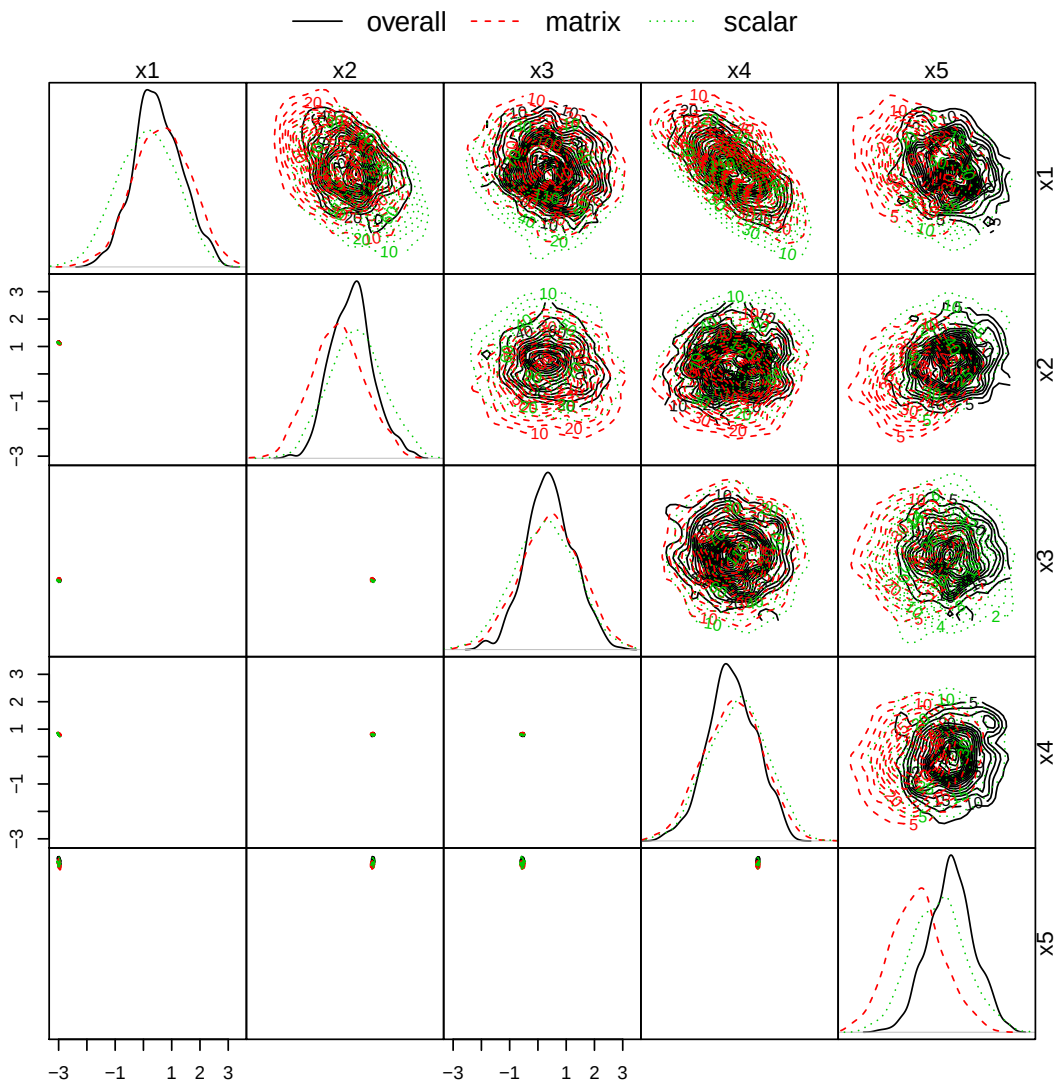
- Ω_s can be estimated by the sample covariance of $\theta_s^{(1)}, ..., \theta_s^{(N)}$. Simplification: assume Ω_s to be diagonal.

LOGISTIC REGRESSION 100 OBS PER MACHINE



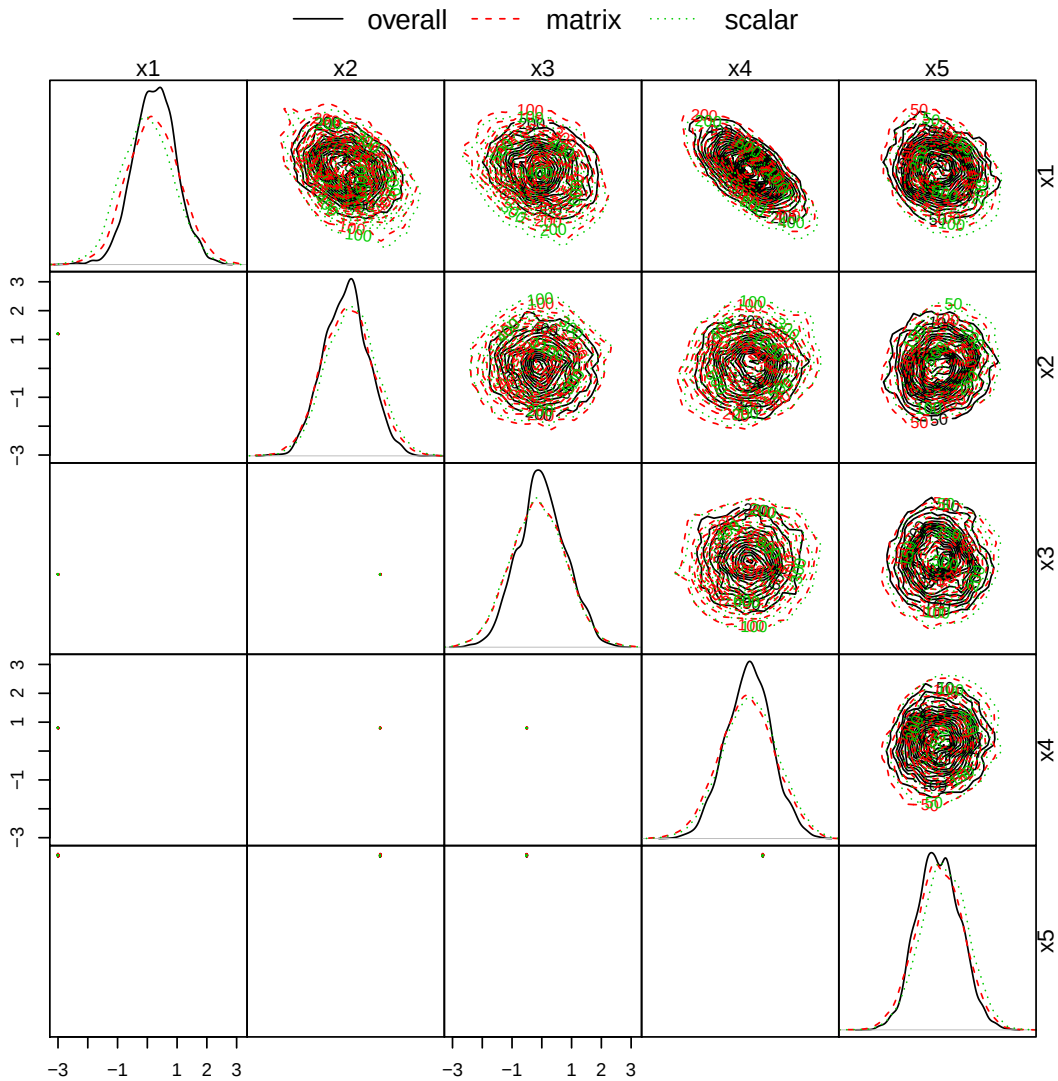
From Scott et al. (2013) [8]

LOGISTIC REGRESSION 1000 OBS PER MACHINE



From Scott et al. (2013) [8]

LOGISTIC REGRESSION 10000 OBS PER MACHINE

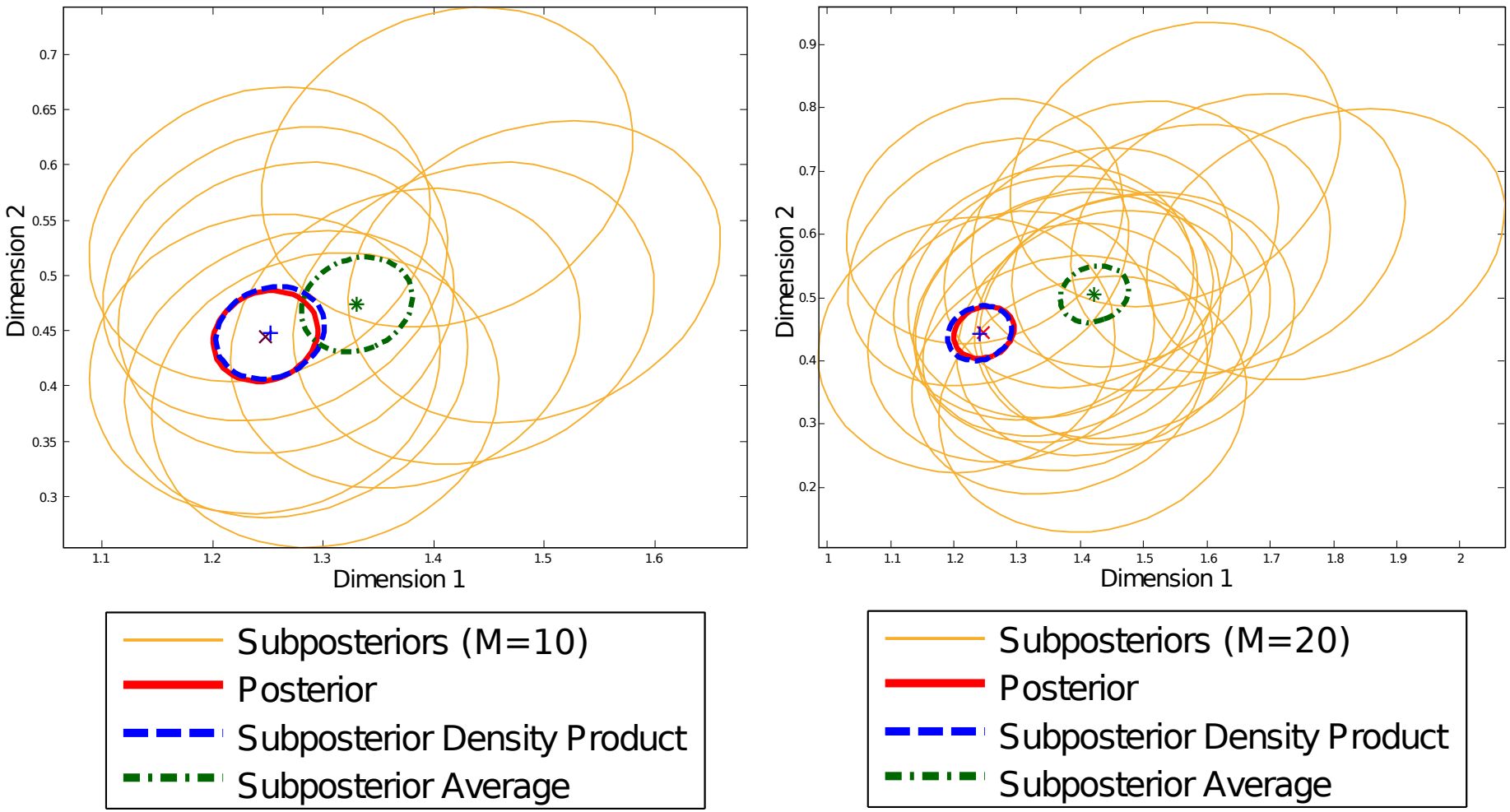


From Scott et al. (2013) [8]

Kernel-based approaches and cousins

- Neiswanger et al. (2013) [9]
  - **Nonparametric**. Estimates each subposterior by a kernel density estimator.
  - **Semiparametric**. Estimates each subposterior by the semiparametric density estimator in Hjort and Glad (1995) [10].
- **Yin Yang sampler** (Posekany and Fruhwirth-Schnatter, in progress).
- **Weierstrass sampler** (Wang and Dunson, 2013) [11]
- Hard to estimate KDEs in moderate to high dimensions.

CONSENSUS MC VS NONPARAMETRIC



From Neiswanger et al. (2013) [9]

Median subposterior (Minsker et al) [12]

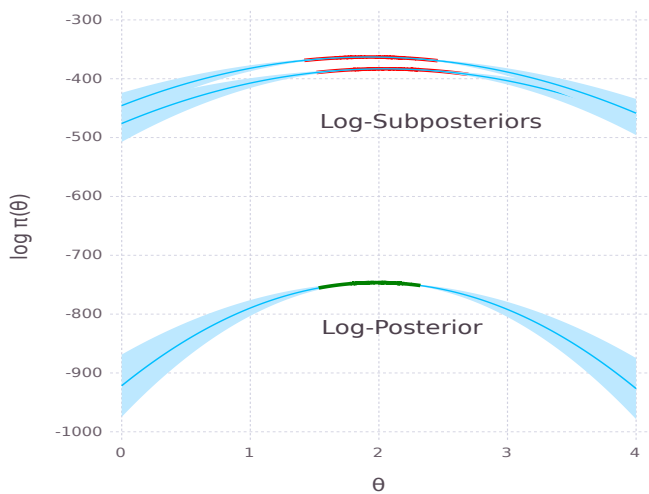
- **Algorithm**:
  - **Run MCMC on scaled subposteriors** (each of the data observations are replicated S times to mimic the posterior spread).
  - Return the **median subposterior** of the S subposteriors.
- Note: the **median is over a space of probability distribution** using some suitable metric. Uniform distribution over atoms $\mathbf{x}_1, ..., \mathbf{x}_m$:

$$\mathbf{x}_{median} \equiv \text{argmin}_{\mathbf{x} \in \mathcal{X}} \sum_{i=1}^{m} \|\mathbf{x} - \mathbf{x}_j\|$$

- The median is computed by an efficient algorithm. End result: weights on the subposteriors (draws).
- The median subposterior has two **advantages**:
  - it is a better approximation of the full posterior than the individual subposteriors.
  - it is more **resistant to outliers**.

Combining subposteriors using GPs

- Main idea in Nemeth and Sherlock (2016) [13]:
  - use the subposterior MCMC draws **and** the evaluated log-subposterior densities to **fit a GP to each log-subposterior**.
  - Approximate the full log-posterior by **summing the log-subposterior GPs** (sum of GPs is a GP, just like a sum of Gaussians is Gaussian).
- **Posterior distribution of the log posterior density function** (a GP is a distribution over functions), including uncertainty.
- **Prior mean: the subposterior is Gaussian**. Covariance of Gaussian is obtained from MCMC.



Combining subposteriors using GPs

- Algorithm:
  - **Run MCMC on each subposterior**. Save draws $\theta_s^{(i)}$ **and** $\log p(\theta_s^{(i)}|\mathbf{y}_s)$ evaluations, for $i = 1, ..., I$. Let $\mathcal{D}_s = \left\{\theta_s^{(1:I)}, \log p(\theta_s^{(1:I)}|\mathbf{y}_s)\right\}$.
  - **Fit a noise-free GP regression** to $\mathcal{D}_s$ with response $\log p(\theta_s^{(1:I)}|\mathbf{y}_s)$. Predictive distribution for the log subposterior at a new set of parameter values $\theta^{(1:J)}$

$$\log p_s(\theta^{(1:J)})|\mathcal{D}_s \sim GP\left(\mu_s(\theta^{(1:J)}), \Sigma_s(\theta^{(1:J)})\right)$$

  - **Sum subposterior GPs** to approximate the full data $\log p(\theta|\mathbf{y})$

$$\log p(\theta^{(1:J)}|\mathbf{y})|\mathcal{D} \sim GP\left(\sum_{s=1}^{S}\mu_s(\theta^{(1:J)}), \sum_{s=1}^{S}\Sigma_s(\theta^{(1:J)})\right)$$

GP-HMC SAMPLER

- **Posterior mean** of $p(\theta|\mathbf{y})|\mathcal{D}$ (using properties of log normal)

$$\hat{p}_E(\theta|\mathbf{y}) \equiv \mathbb{E}\left(p(\theta|\mathbf{y})|\mathcal{D}\right) = \exp\left(\sum_{s=1}^{S}\mu_s(\theta) + \frac{1}{2}\sum_{s=1}^{S}\Sigma_s(\theta)\right)$$

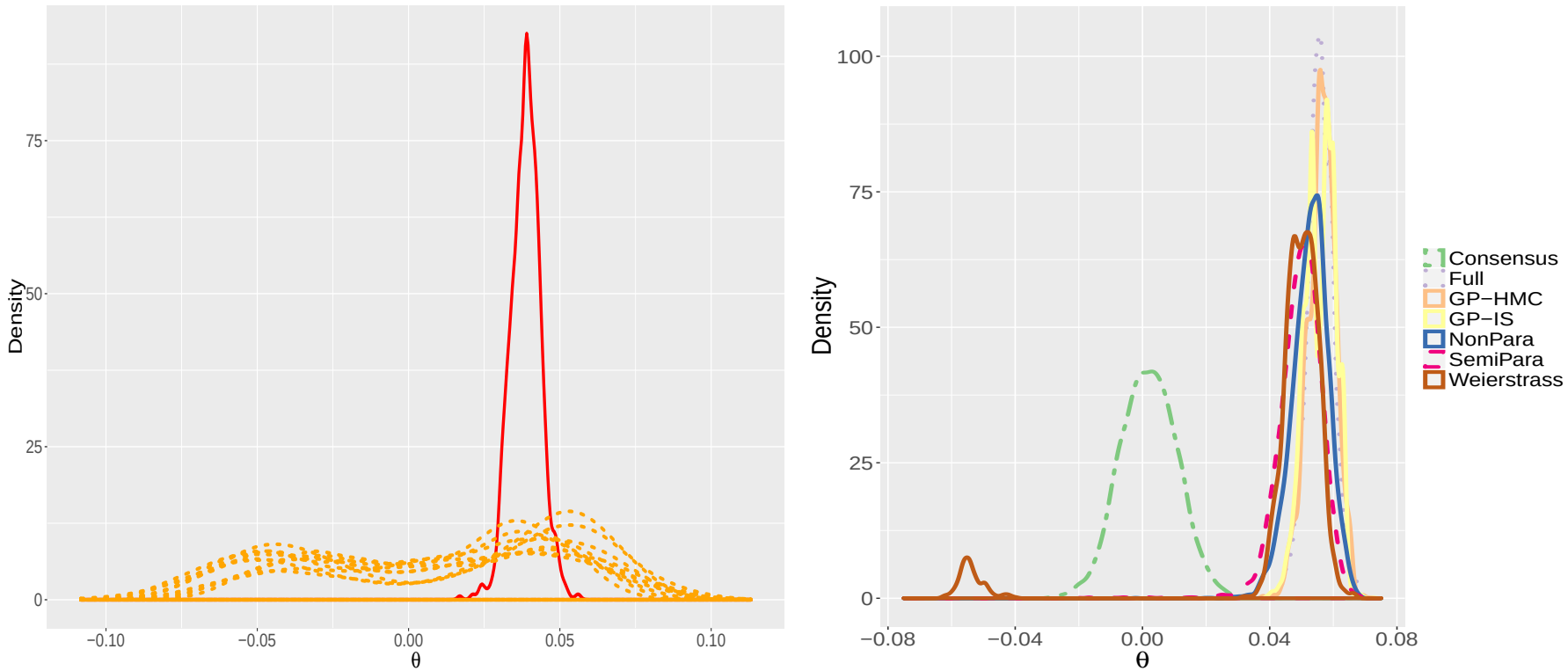
- **Sample** $\hat{p}_E(\theta|\mathbf{y})$ using **Hamiltonian Monte Carlo** (**HMC**) on a single machine.
  - $\hat{p}_E(\theta|\mathbf{y})$ does not depend on the original data. No need to transfer data from the subposterior machines.
  - $\hat{p}_E(\theta|\mathbf{y})$ relatively cheap to evaluate.
  - The costly leap-frog steps in HMC are cheap since gradients of $\log \hat{p}_E(\theta|\mathbf{y})$ are available in convenient closed form.

Distributed Importance Sampling

- Nemeth and Sherlock (2016) [13]: **distributed importance sampler** to **correct any** combination approach.
- Algorithm:
  - On central machine: propose $\theta^{(1:N)}$ from joint proposal distribution $q(\theta^{(1:N)})$ with identical marginals $q_1(\theta^{(i)})$.
  - Transfer $\theta^{(1:N)}$ to all submachines.
  - Evaluate $p(\theta^{(1:N)}|\mathbf{y}_s)$ on each submachine and return to central machine.
  - Compute weights on each draw $w_i = \prod_{s=1}^{S} p(\theta^{(i)}|\mathbf{y}_s) / q_1(\theta^{(i)})$.
  - Output a weighted posterior sample $\left\{\theta^{(i)}, w_i\right\}_{i=1}^{N}$.
- Unweighted posterior sample by iid sampling from $\left\{\theta^{(i)}, w_i\right\}_{i=1}^{N}$.
- $q(\theta^{(1:N)})$ is iid Gaussian in the consensus approach.
- $q(\theta^{(1:N)})$ sampled by HMC in the GP-HMC sampler.

Mixture of Laplace example [13]

- Simulated data set from mixture of two Laplace distributions.
- 1M observations. 1 parameter. $S = 20$ machines.



From Nemeth and Sherlock (2016) [13]

Mixture of Laplace example

Algorithm	$D_{Mah.}$	$D_{KL}(\pi\|\|\hat{\pi})$
Consensus	13.46 (5.62)	5.73 (5.27)
Nonparametric	1.69 (1.66)	0.26 (0.18)
Semiparametric	2.39 (1.79)	0.77 (0.46)
Weierstrass	6.52 (1.76)	0.99 (0.75)
GP-HMC sampler	**1.10 (0.96)**	**0.15 (0.13)**
GP-IS sampler	1.62	0.63

From Nemeth and Sherlock (2016) [13]

Logistic regression example

- HEPMASS data set.
- 1M observations.
- 27 parameters.
- $S = 20$ machines.

Algorithm	$D_{Mah.}$	$D_{KL}(\pi \| \hat{\pi})$
Consensus	6.13 (5.80)	13.65 (**13.22**)
Nonparametric	10.73 (5.59)	17.23 (15.76)
Semiparametric	**5.36** (5.14)	15.93 (15.01)
Weierstrass	6.48 (**4.62**)	14.75 (13.98)
GP-HMC sampler	5.90 (5.27)	**13.28** (13.57)
GP-IS sampler	6.07	13.56

From Nemeth and Sherlock (2016) [13]


A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes, "On the utility of graphics cards to perform massively parallel simulation of advanced monte carlo methods," *Journal of computational and graphical statistics*, vol. 19, no. 4, pp. 769–789, 2010.


M. A. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West, "Understanding gpu programming for statistical computation: Studies in massively parallel massive mixtures," *Journal of Computational and Graphical Statistics*, vol. 19, no. 2, pp. 419–438, 2010.


A. Eklund, P. Dufort, M. Villani, and S. LaConte, "Broccoli: Software for fast fmri analysis on many-core cpus and gpus," *Recent Advances and the Future Generation of Neuroinformatics Infrastructure*, p. 208, 2015.


M. Magnusson, L. Jonsson, M. Villani, and D. Broman, "Parallelizing lda using partially collapsed gibbs sampling," *arXiv preprint arXiv:1506.03784*, 2015.


I. Strid, "Efficient parallelisation of metropolis–hastings algorithms using a prefetching approach," *Computational Statistics & Data Analysis*, vol. 54, no. 11, pp. 2814–2835, 2010.


M. Banterle, C. Grazian, and C. P. Robert, "Accelerating metropolis-hastings algorithms: Delayed acceptance with prefetching," *arXiv preprint arXiv:1406.2660*, 2014.


A. Terenin, D. Simpson, and D. Draper, "Asynchronous distributed gibbs sampling," *arXiv preprint arXiv:1509.08999*, 2015.


S. L. Scott, A. W. Blocker, F. V. Bonassi, H. Chipman, E. George, and R. McCulloch, "Bayes and big data: the consensus Monte Carlo algorithm," in *Bayes 250 conference*, vol. 16, 2013.


W. Neiswanger, C. Wang, and E. Xing, "Asymptotically exact, embarrassingly parallel MCMC," *arXiv preprint arXiv:1311.4780*, 2013.


N. L. Hjort and I. K. Glad, "Nonparametric density estimation with a parametric start," *The Annals of Statistics*, pp. 882–904, 1995.


X. Wang and D. B. Dunson, "Parallel MCMC via Weierstrass sampler," *arXiv preprint arXiv:1312.4605*, 2013.


S. Minsker, S. Srivastava, L. Lin, and D. Dunson, "Scalable and robust Bayesian inference via the median posterior," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1656–1664, 2014.


C. Nemeth and C. Sherlock, "Merging mcmc subposteriors through gaussian-process approximations," *arXiv preprint arXiv:1605.08576*, 2016.