Algorithms for unifying statistical inference

Ryan Chan

9 Feb 2021

Warwick Statistics

The Alan Turing Institute

Outline

Fusion Problem

Target:

$$\pi(x) \propto \prod_{c=1}^{C} f_c(x)$$

where each *sub-posterior*, $f_c(x)$, is a density representing one of the C distributed inferences we wish to unify

Assume we can sample $x^{(c)} \sim f_c(x)$

Applications:

- Big Data (by construction)
- Tempering (by construction)
- Expert elicitation: combining views of multiple experts
- Privacy setting

Fusion Problem

Target:

$$\pi(x) \propto \prod_{c=1}^{C} f_c(x)$$

where each *sub-posterior*, $f_c(x)$, is a density representing one of the C distributed inferences we wish to unify

Assume we can sample $x^{(c)} \sim f_c(x)$

Applications:

- Big Data (by construction)
- Tempering (by construction)
- Expert elicitation: combining views of multiple experts
- Privacy setting

Fusion Problem

Target:

$$\pi(x) \propto \prod_{c=1}^{C} f_c(x)$$

where each *sub-posterior*, $f_c(x)$, is a density representing one of the C distributed inferences we wish to unify

Assume we can sample $x^{(c)} \sim f_c(x)$

Applications:

- Big Data (by construction)
- Tempering (by construction)
- Expert elicitation: combining views of multiple experts
- Privacy setting

Fusion for Big Data

Consider we have data x with a large number of observations n

The likelihood $\ell(x|\ )$ becomes expensive to calculate

- This makes MCMC prohibitively slow for big data

Potential solution:

$$(\ |x) \propto \prod_{i=1}^{n} \ell(x_i|\ )\,(\ ) = \prod_{c=1}^{C} \ell(x_c|\ )\,(\ )^{\frac{1}{C}}$$

where x_c denotes the c-th subset for $c = 1, \ldots, C$ and $(\ ) = \prod_{c=1}^{C} (\ )^{\frac{1}{C}}$ is the prior

Advantage: inference on each smaller dataset can be conducted independently and in parallel

Fusion for Big Data

- Consider we have data x with a large number of observations n
- The likelihood $\ell(x \mid \theta)$ becomes expensive to calculate
  - This makes MCMC prohibitively slow for big data
- Potential solution:

$$\pi(\theta \mid x) \propto \prod_{i=1}^{n} \ell(x \mid \theta)\,\pi(\theta) = \prod_{c=1}^{C} \ell(x_c \mid \theta)\,\pi(\theta)^{\frac{1}{C}}$$

where x_c denotes the c-th subset for $c = 1, \ldots, C$ and $\pi(\theta) = \prod_{c=1}^{C} \pi(\theta)^{\frac{1}{C}}$ is the prior

- Advantage: inference on each smaller dataset can be conducted independently and in parallel

Fusion for Big Data

- Consider we have data x with a large number of observations n
- The likelihood $\ell(x \mid \theta)$ becomes expensive to calculate
  - This makes MCMC prohibitively slow for big data
- Potential solution:

$$\pi(\theta \mid x) \propto \prod_{i=1}^{n} \ell(x \mid \theta)\,\pi(\theta) = \prod_{c=1}^{C} \ell(x_c \mid \theta)\,\pi(\theta)^{\frac{1}{C}}$$

where x_c denotes the c-th subset for $c = 1, \ldots, C$ and $\pi(\theta) = \prod_{c=1}^{C} \pi(\theta)^{\frac{1}{C}}$ is the prior

- Advantage: inference on each smaller dataset can be conducted independently and in parallel

Fusion for Big Data

- Consider we have data x with a large number of observations n
- The likelihood $\ell(x\,|\ \ )$ becomes expensive to calculate
  - This makes MCMC prohibitively slow for big data
- Potential solution:

$$(\ \ |x) \propto \prod_{i=1}^{n} \ell(x\,|\ \ )\ (\ \ ) = \prod_{c=1}^{C} \ell(x_c\,|\ \ )\ (\ \ )^{\frac{1}{C}}$$

where x_c denotes the c-th subset for $c = 1, \ldots, C$ and $(\ \ ) = \prod_{c=1}^{C} (\ \ )^{\frac{1}{C}}$ is the prior

- Advantage: inference on each smaller dataset can be conducted independently and in parallel

Fusion for Big Data

- Consider we have data x with a large number of observations n
- The likelihood $\ell(x \mid \theta)$ becomes expensive to calculate
  - This makes MCMC prohibitively slow for big data
- Potential solution:

$$\pi(\theta \mid x) \propto \prod_{i=1}^{n} \ell(x \mid \theta)\, \pi(\theta) = \prod_{c=1}^{C} \ell(x_c \mid \theta)\, \pi(\theta)^{\frac{1}{C}}$$

where x_c denotes the c-th subset for $c = 1, \ldots, C$ and $\pi(\theta) = \prod_{c=1}^{C} \pi(\theta)^{\frac{1}{C}}$ is the prior

- Advantage: inference on each smaller dataset can be conducted independently and in parallel

Fusion for Tempering

Consider the *power-tempered target distribution*
$(x) = [\ (x)]$ for $2\ (0; 1]$

MCMC can become computationally expensive to sample from multi-modal densities and can get stuck in modes

Potential solution:

$$(x) \ / \ (x)^{\frac{1}{}} \ / \ \prod_{c=1}^{\frac{1}{}} \ (x)$$

where $\frac{1}{} \ 2 \ \mathrm{N}$

Fusion for Tempering

Consider the *power-tempered target distribution*

$$(x) = [\;(x)] \quad \text{for} \quad 2\;(0;1]$$

MCMC can become computationally expensive to sample from multi-modal densities and can get stuck in modes

Potential solution:

$$(x) \;/\; (x)^{1} \;/\; \prod_{c=1}^{1} (x)$$

where $\frac{1}{} \; 2 \; \mathbb{N}$

Fusion for Tempering

Consider the *power-tempered target distribution*
$(x) = [\ (x)]$ for $2\ (0;1]$

MCMC can become computationally expensive to sample from multi-modal densities and can get stuck in modes

Potential solution:

$$(x) \ / \ (x)^{\frac{1}{}} \ / \ \prod_{c=1}^{\frac{1}{}} \ (x)$$

where $\frac{1}{} \ 2 \ \mathbb{N}$

Fusion in a privacy setting

Suppose have C parties that wish to combine their inferences but either:

- underlying model $f_c(x)$ cannot be shared, or
- underlying data x_c cannot be shared

e.g. healthcare settings

If we have a method that can combine inferences or samples, would need some mechanism that ensures the privacy of the model or data

Fusion in a privacy setting

Suppose have C parties that wish to combine their inferences but either:

- underlying model $f_c(x)$ cannot be shared, or
- underlying data x_c cannot be shared

e.g. healthcare settings

If we have a method that can combine inferences or samples, would need some mechanism that ensures the privacy of the model or data

Fusion in a privacy setting

Suppose have C parties that wish to combine their inferences but either:

- underlying model $f_c(x)$ cannot be shared, or
- underlying data x_c cannot be shared

e.g. healthcare settings

If we have a method that can combine inferences or samples, would need some mechanism that ensures the privacy of the model or data

Fusion in a privacy setting

Suppose have C parties that wish to combine their inferences but either:

- underlying model $f_c(x)$ cannot be shared, or
- underlying data x_c cannot be shared

e.g. healthcare settings

If we have a method that can combine inferences or samples, would need some mechanism that ensures the privacy of the model or data

Fusion in a privacy setting

Suppose have C parties that wish to combine their inferences but either:

underlying model $f_c(x)$ cannot be shared, or
underlying data x_c cannot be shared

e.g. healthcare settings

If we have a method that can combine inferences or samples, would need some mechanism that ensures the privacy of the model or data

Fork-and-join fusion

The fork-and-join approach:

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

Kernel density averaging (KDEMC)

- Apply a kernel density estimation to each sub-posterior, $\hat{f}_c(x)$ [Neiswanger et al., 2013]. Then approximate full posterior by

$$\hat{}(x) = \prod_{c=1}^{C} \hat{f}_c(x)$$

- If Gaussian kernels are used, $\hat{}(x)$ is a product of Gaussian mixtures with $O(NC)$ components (N samples, C sub-posteriors)
- Neiswanger et al. [2013] suggest sampling from the Gaussian mixture using MCMC
- Can be computationally expensive and inefficient
- Does not scale well with dimension

Kernel density averaging (KDEMC)

Apply a kernel density estimation to each sub-posterior $\hat{f}_c(x)$ [Neiswanger et al., 2013]. Then approximate full posterior by

$$\hat{\pi}(x) = \prod_{c=1}^{C} \hat{f}_c(x)$$

If Gaussian kernels are used, $\hat{\pi}(x)$ is a product of Gaussian mixtures with $O(NC)$ components (N samples, C sub-posteriors)

Neiswanger et al. [2013] suggest sampling from the Gaussian mixture using MCMC

Can be computationally expensive and inefficient

Does not scale well with dimension

Kernel density averaging (KDEMC)

- Apply a kernel density estimation to each sub-posterior, $\hat{f}_c(x)$ [Neiswanger et al., 2013]. Then approximate full posterior by

$$\hat{\pi}(x) = \prod_{c=1}^{C} \hat{f}_c(x)$$

- If Gaussian kernels are used, $\hat{\pi}(x)$ is a product of Gaussian mixtures with $O(NC)$ components (N samples, C sub-posteriors)
- Neiswanger et al. [2013] suggest sampling from the Gaussian mixture using MCMC
- Can be computationally expensive and inefficient
- Does not scale well with dimension

Kernel density averaging (KDEMC)

- Apply a kernel density estimation to each sub-posterior, $\hat{f}_c(x)$ [Neiswanger et al., 2013]. Then approximate full posterior by

$$\hat{\pi}(x) = \prod_{c=1}^{C} \hat{f}_c(x)$$

- If Gaussian kernels are used, $\hat{\pi}(x)$ is a product of Gaussian mixtures with $O(NC)$ components (N samples, C sub-posteriors)
- Neiswanger et al. [2013] suggest sampling from the Gaussian mixture using MCMC
- Can be computationally expensive and inefficient
- Does not scale well with dimension

Kernel density averaging (KDEMC)

Apply a kernel density estimation to each sub-posterior, $\hat{f}_c(x)$ [Neiswanger et al., 2013]. Then approximate full posterior by

$$\hat{\pi}(x) = \prod_{c=1}^{C} \hat{f}_c(x)$$

- If Gaussian kernels are used, $\hat{\pi}(x)$ is a product of Gaussian mixtures with $O(N^C)$ components (N samples, C sub-posteriors)
- Neiswanger et al. [2013] suggest sampling from the Gaussian mixture using MCMC
- Can be computationally expensive and inefficient
- Does not scale well with dimension

Consensus Monte Carlo

- Approximate the full posterior as a weighted average of the sub-posterior samples [Scott et al., 2016]
- Suppose have MCMC samples $x_1^{(c)}, \ldots, x_N^{(c)}$ from $f_c(x)$ for $c = 1, \ldots, C$. Then approximate full posterior

$$x_i = \left(\sum_{c=1}^{C} W_c\right)^{-1} \left(\sum_{c=1}^{C} W_c x_i^{(c)}\right)$$

where $W_c \in \mathbb{R}^d$ is a weight matrix for sub-posterior c (typically take $W_c = \hat{\Sigma}_c$)
- Method is exact if sub-posteriors are Gaussian (motivated by Bernstein-von Mises Theorem)
- Very scalable

Consensus Monte Carlo

- Approximate the full posterior as a weighted average of the sub-posterior samples [Scott et al., 2016]
- Suppose have MCMC samples $x_1^{(c)}, \ldots, x_N^{(c)}$ from $f_c(x)$ for $c = 1, \ldots, C$. Then approximate full posterior

$$x_i = \left(\sum_{c=1}^{C} W_c\right)^{-1} \left(\sum_{c=1}^{C} W_c x_i^{(c)}\right)$$

where $W_c \in \mathbb{R}^d$ is a weight matrix for sub-posterior c (typically take $W_c = \hat{\Sigma}_c$)

- Method is exact if sub-posteriors are Gaussian (motivated by Bernstein-von Mises Theorem)
- Very scalable

Consensus Monte Carlo

- Approximate the full posterior as a weighted average of the sub-posterior samples [Scott et al., 2016]
- Suppose have MCMC samples $x_1^{(c)}, \ldots, x_N^{(c)}$ from $f_c(x)$ for $c = 1, \ldots, C$. Then approximate full posterior

$$x_i = \left(\sum_{c=1}^{C} W_c \right)^{-1} \left(\sum_{c=1}^{C} W_c x_i^{(c)} \right)$$

where $W_c \in \mathbb{R}^d$ is a weight matrix for sub-posterior c (typically take $W_c = \hat{\Sigma}_c$)
- Method is exact if sub-posteriors are Gaussian (motivated by Bernstein-von Mises Theorem)
- Very scalable

Consensus Monte Carlo

- Approximate the full posterior as a weighted average of the sub-posterior samples [Scott et al., 2016]
- Suppose have MCMC samples $x_1^{(c)}, \ldots, x_N^{(c)}$ from $f_c(x)$ for $c = 1, \ldots, C$. Then approximate full posterior

$$x_i = \left(\sum_{c=1}^{C} W_c \right)^{-1} \left(\sum_{c=1}^{C} W_c x_i^{(c)} \right)$$

where $W_c \in \mathbb{R}^d$ is a weight matrix for sub-posterior c (typically take $W_c = \hat{\Sigma}_c$)
- Method is exact if sub-posteriors are Gaussian (motivated by Bernstein-von Mises Theorem)
- Very scalable

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

A primary weakness of these methods is that the recombination is inexact in general and involve approximations

However, Monte Carlo Fusion [Dai et al., 2019] is exact

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

A primary weakness of these methods is that the recombination is inexact in general and involve approximations

However, Monte Carlo Fusion [Dai et al., 2019] is exact

Current Fork-and-Join Methods

Several fork-and-join methods have been developed. For instance

- Gaussian approximations to sub-posteriors [Neiswanger et al., 2013]
- Kernel density averaging [Neiswanger et al., 2013]
- Consensus Monte Carlo [Scott et al., 2016]

A primary weakness of these methods is that the recombination is inexact in general and involve approximations

However, Monte Carlo Fusion [Dai et al., 2019] is exact

Constructing a rejection sampler - An Extended Target

Proposition

Suppose that $p_c(y \mid x^{(c)})$ is the transition density of a stochastic process with stationary distribution $f_c^2(x)$. The $(C+1)\,d$-dimensional (fusion) density proportional to the integrable function

$$g\left(x^{(1)},\dots,x^{(C)},y\right) \propto \prod_{c=1}^{C}\left[f_c^2\left(x^{(c)}\right) \, p_c\left(y \mid x^{(c)}\right) \, \frac{1}{f_c(y)} \right]$$

admits the marginal density for y.

Main idea: If we can sample from g, then we can can obtain a draw from the fusion density $y(\;)$

Constructing a rejection sampler - An Extended Target

Proposition

Suppose that $p_c(y \mid x^{(c)})$ is the transition density of a stochastic process with stationary distribution $f_c^2(x)$. The $(C+1)$ d-dimensional (fusion) density proportional to the integrable function

$$g\left(x^{(1)}, \ldots, x^{(C)}, y\right) \propto \prod_{c=1}^{C} f_c^2\left(x^{(c)}\right) p_c\left(y \mid x^{(c)}\right) \frac{1}{f_c(y)}$$

admits the marginal density for y.

Main idea: If we can sample from g, then we can can obtain a draw from the fusion density y(   )

Rejection Sampling (Double Langevin Approach)

- There are many possible choices for $p_c(y \mid x)$
- Let $p_c(y \mid x) := p_{T,c}(y \mid x)$, the transition density of the d-dimensional (double) Langevin (DL) diffusion processes $X_t^{(c)}$ for $c = 1, \ldots, C$, from x to y for a pre-defined time $T > 0$ given by

$$dX_t^{(c)} = \Lambda_c \nabla \log f_c\left(X_t^{(c)}\right) dt + \Lambda_c^{1/2} dW_t^{(c)}$$

  - $W_t^{(c)}$ is d-dimensional Brownian motion
  - Λ_c is the pre-conditioning matrix associated with sub-posterior f_c
  - ∇ is the gradient operator over x
  - Has stationary distribution $f_c^2(x)$

Rejection Sampling (Double Langevin Approach)

- There are many possible choices for $p_c(y \mid x)$
- Let $p_c(y \mid x) := p_{T,c}(y \mid x)$, the transition density of the d-dimensional (double) Langevin (DL) diffusion processes $X_t^{(c)}$ for $c = 1, \ldots, C$, from x to y for a pre-defined time $T > 0$ given by

$$dX_t^{(c)} = \Lambda_c \nabla \log f_c\left(X_t^{(c)}\right) dt + \sqrt{2}\,\Lambda_c^{1/2} dW_t^{(c)}$$

- $W_t^{(c)}$ is d-dimensional Brownian motion
- Λ_c is the pre-conditioning matrix associated with sub-posterior f_c
- ∇ is the gradient operator over x
- Has stationary distribution $f_c^2(x)$

Rejection Sampling (Double Langevin Approach)

- There are many possible choices for $p_c(y \mid x)$
- Let $p_c(y \mid x) := p_{T,c}(y \mid x)$, the transition density of the d-dimensional (double) Langevin (DL) diffusion processes $X_t^{(c)}$ for $c = 1, \ldots, C$, from x to y for a pre-defined time $T > 0$ given by

$$dX_t^{(c)} = \Lambda_c \nabla \log f_c\left(x_t^{(c)}\right) dt + \Lambda_c^{1/2} dW_t^{(c)}$$

- $W_t^{(c)}$ is d-dimensional Brownian motion
- Λ_c is the pre-conditioning matrix associated with sub-posterior f_c
- ∇ is the gradient operator over x
- Has stationary distribution $f_c^2(x)$

Rejection Sampling (Double Langevin Approach)

Extended Target Density:

$$g\left(x^{(1:C)}, y\right) \propto \prod_{c=1}^{C} \left[f_c^2\left(x^{(c)}\right) \cdot p_c\left(y \mid x^{(c)}\right) \cdot \frac{1}{f_c(y)} \right]$$

Consider the proposal density h for the extended target g:

$$h\left(x^{(1:C)}, y\right) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \cdot \exp\left(-\frac{1}{2}(y - \tilde{x})^{\intercal} \Lambda^{-1} (y - \tilde{x}) \right)$$

$\tilde{x} := \left(\sum_{c=1}^{C} \Lambda_c^{-1}\right)^{-1} \sum_{c=1}^{C} \Lambda_c^{-1} x^{(c)}$

$\Lambda^{-1} := \frac{1}{T} \sum_{c=1}^{C} \Lambda_c^{-1}$

T is an arbitrary positive constant

Rejection Sampling (Double Langevin Approach)

Extended Target Density:

$$g\left(x^{(1:C)}; y\right) \propto \prod_{c=1}^{C} \left[f_c^2\left(x^{(c)}\right) \cdot p_c\left(y \mid x^{(c)}\right) \cdot \frac{1}{f_c(y)} \right]$$

Consider the proposal density h for the extended target g:

$$h\left(x^{(1:C)}; y\right) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \cdot \exp\left(-\frac{1}{2}(y - \bar{x})^{\intercal} \Lambda^{-1} (y - \bar{x}) \right)$$

$\bar{x} := \left(\sum_{c=1}^{C} \Lambda_c^{-1} \right)^{-1} \sum_{c=1}^{C} \Lambda_c^{-1} x^{(c)}$

$\Lambda^{-1} := \frac{1}{T} \sum_{c=1}^{C} \Lambda_c^{-1}$

T is an arbitrary positive constant

Rejection Sampling (Double Langevin Approach)

Simulation from h is easy:

$$h(x^{(1:C)}, y) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \exp\left(-\frac{1}{2}(y - \bar{x})^{\intercal} \Lambda^{-1} (y - \bar{x}) \right)$$

1. Simulate $x^{(c)} \sim f_c(x)$ independently
2. Simulate $y \sim N(\bar{x}, \Lambda)$

Rejection Sampling (Double Langevin Approach)

Simulation from h is easy:

$$h(x^{(1:C)}; y) \propto \prod_{c=1}^{C} \left[f_c \left(x^{(c)} \right) \right] \exp \left(\frac{1}{2}(y - \bar{x})^{\intercal} \Lambda^{-1} (y - \bar{x}) \right)$$

1. Simulate $x^{(c)} \sim f_c(x)$ independently
2. Simulate $y \sim N(\bar{x}; \Lambda)$

Rejection Sampling (Double Langevin Approach)

Simulation from h is easy:

$$h(x^{(1:C)}, y) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \cdot \exp\left(-\frac{1}{2}(y - \bar{x})^{\intercal} \Lambda^{-1} (y - \bar{x}) \right)$$

1. Simulate $x^{(c)} \sim f_c(x)$ independently
2. Simulate $y \sim \mathcal{N}(\bar{x}; \Lambda)$

Rejection Sampling - acceptance probability

Acceptance probability:

$$\frac{g(x^{(1)},\ldots,x^{(C)},y)}{h(x^{(1)},\ldots,x^{(C)},y)} \propto \rho \cdot Q$$

where

$$\rho(x^{(1:C)}) = \exp\left\{ -\sum_{c=1}^{C} \frac{(\bar{x} - x^{(c)})^{\intercal} \Lambda_c^{-1} (\bar{x} - x^{(c)})}{2T} \right\}$$

$$Q(x^{(1:C)}, y) := \prod_{c=1}^{C} \mathbb{E}_{W_c}\left[\exp\left\{ -\int_0^T \phi_c\left(x_t^{(c)}\right) dt \right\} \right]$$

where W_c denotes the law of a Brownian bridge $\{x_t^{(c)}, t \in [0, T]\}$ with $x_0^{(c)} := x^{(c)}$ and $x_T^{(c)} := y$ and covariance matrix Λ_c

Rejection Sampling - acceptance probability

Acceptance probability:

$$\frac{g(x^{(1)};:::;x^{(C)};y)}{h(x^{(1)};:::;x^{(C)};y)} / \quad Q$$

where

$$(x^{(1:C)}) = \exp\left\{ \sum_{c=1}^{C} \frac{(\bar{x} \; x^{(c)})^{\intercal} \; {}_{c}^{-1}(\bar{x} \; x^{(c)})}{2T} \right\}$$

$$Q(x^{(1:C)};y) := \prod_{c=1}^{C} \mathbb{E}_{W_c}\left[\exp\left\{ \int_0^T \; {}_c \; x_t^{(c)} \quad {}_c \; dt \right\} \right]$$

where W_c denotes the law of a Brownian bridge $\{x_t^{(c)}; t \in [0; t]\}$ with $x_0^{(c)} := x^{(c)}$ and $x_T^{(c)} := y$ and covariance matrix $_c$

Q Acceptance Probability

$$Q := \prod_{c=1}^{C} \mathbb{E}_{\mathbb{W}_c}\left[\exp\left(-\int_0^T \left(\phi_c\left(x_t^{(c)}\right) - \Phi_c\right) dt\right)\right]$$

where

$$\phi_c(x) = \frac{1}{2}\left(\nabla \log f_c(x)^\intercal \Lambda_c \nabla \log f_c(x) + \sum_{k=1}^{d} \Lambda_{c;kk} \frac{\partial^2 \log f_c(x)}{\partial x_k^2}\right)$$

- Φ_c are constants such that for all x, $\phi_c(x) \geq \Phi_c$ for $c \in \{1, \dots, C\}$
- Events of probability Q can be simulated using Poisson thinning and methodology called Path-space Rejection Sampling (PSRS) or the Exact Algorithm (Beskos et al. [2005], Beskos et al. [2006], Pollock et al. [2016])

Interpretation

Correct a simple weighted average $\bar{x}$ of sub-posterior values to a Monte Carlo draw from (x) with acceptance probability Q

Double Langevin Approach - Summary

Proposal:

$$h\left(x^{(1:C)}, y\right) \propto \prod_{c=1}^{C}\left[f_c\left(x^{(c)}\right)\right] \exp\left(\frac{1}{2}(y \quad \bar{x})^{\intercal} \quad ^{-1}(y \quad \bar{x})\right)$$

Accept y as a draw from fusion density with probability:

$$\frac{g(x^{(1)},\dots,x^{(C)},y)}{h(x^{(1)},\dots,x^{(C)},y)} \propto \quad Q$$

Monte Carlo Fusion Algorithm:

1. Simulate $x^{(c)} \quad f_c(x)$ and $y \quad N(\bar{x}, \ )$
2. Accept y with probability Q

Double Langevin Approach - Summary

Proposal:

$$h\left(x^{(1:C)}, y\right) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \exp\left(-\frac{1}{2}(y-\bar{x})^{\intercal} \Lambda^{-1} (y-\bar{x})\right)$$

Accept y as a draw from fusion density with probability:

$$\frac{g(x^{(1)}, \ldots, x^{(C)}, y)}{h(x^{(1)}, \ldots, x^{(C)}, y)} \propto Q$$

Monte Carlo Fusion Algorithm:

1. Simulate $x^{(c)} \sim f_c(x)$ and $y \sim N(\bar{x}, \Lambda)$
2. Accept y with probability Q

Double Langevin Approach - Summary

Proposal:

$$h\left(x^{(1:C)}, y\right) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \exp\left(-\frac{1}{2}(y - \bar{x})^{\intercal} \Lambda^{-1} (y - \bar{x}) \right)$$

Accept y as a draw from fusion density with probability:

$$\frac{g(x^{(1)}, \ldots, x^{(C)}, y)}{h(x^{(1)}, \ldots, x^{(C)}, y)} \propto \quad Q$$

Monte Carlo Fusion Algorithm:

1. Simulate $x^{(c)} \sim f_c(x)$ and $y \sim N(\bar{x}, \Lambda)$
2. Accept y with probability Q

Double Langevin Approach - Summary

Proposal:

$$h\left(x^{(1:C)}, y\right) \propto \prod_{c=1}^{C} \left[f_c\left(x^{(c)}\right) \right] \exp\left(-\frac{1}{2}(y - \bar{x})^{\intercal} \Lambda^{-1}(y - \bar{x}) \right)$$

Accept y as a draw from fusion density with probability:

$$\frac{g(x^{(1)}, \ldots, x^{(C)}, y)}{h(x^{(1)}, \ldots, x^{(C)}, y)} \propto \rho \cdot Q$$

Monte Carlo Fusion Algorithm:

1. Simulate $x^{(c)} \sim f_c(x)$ and $y \sim N(\bar{x}, \Lambda)$
2. Accept y with probability $\rho \cdot Q$

Density with light tails

Target: $\pi(x) \propto e^{-\frac{x^4}{2}}$

Sub-posteriors: $f_c(x) \propto e^{-\frac{x^4}{8}}$ for $c = 1, \ldots, 4$

$N = 20{,}000$

Mixture Gaussian

Target: $(x) \propto 0.5N(\ \ 5; 1) + 0.2N(6; 2) + 0.3N(12; 1.5)$

Sub-posteriors: $f_c(x) \propto \ \ (x)^{1/4}$ for $c = 1, \dots, 4$

$N = 20{,}000$

Recall: Fork-and-join

The fork-and-join approach:

Hierarchical Monte Carlo Fusion

Solution: adopt a divide-and-conquer approach:

Example

Target: $(x) \; / \; e^{\frac{x^4}{2}}$

Sub-posteriors: $f_c(x) = e^{\frac{x^4}{2C}}$ for $c = 1; : : : ; C$

Divide-and-Conquer SMC with Fusion

- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - Introduce resampling at the nodes if the ESS falls below some threshold
  - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC with Fusion

- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - Introduce resampling at the nodes if the ESS falls below some threshold
  - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC with Fusion

- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - Introduce resampling at the nodes if the ESS falls below some threshold
  - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC with Fusion

- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - Introduce resampling at the nodes if the ESS falls below some threshold
  - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC with Fusion

- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - Introduce resampling at the nodes if the ESS falls below some threshold
  - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

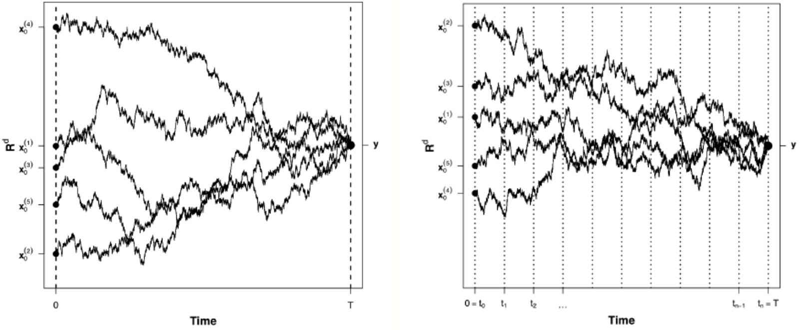
Divide-and-Conquer SMC with Fusion

- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - Introduce resampling at the nodes if the ESS falls below some threshold
  - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Bayesian Fusion

Ongoing work by Dai, H., Pollock, M. and Roberts, G.O.

Tailored to big data Bayesian problems



References

Beskos, A., Papaspiliopoulos, O., Roberts, G. O., and Fearnhead, P. (2006). Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):333–382.

Beskos, A., Roberts, G. O., et al. (2005). Exact simulation of diffusions. *The Annals of Applied Probability*, 15(4):2422–2444.

Dai, H., Pollock, M., and Roberts, G. (2019). Monte Carlo Fusion. *Journal of Applied Probability*, 56(1):174–191.

Lindsten, F., Johansen, A. M., Naesseth, C. A., Kirkpatrick, B., Schön, T. B., Aston, J., and Bouchard-Côté, A. (2017). Divide-and-conquer with Sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458.

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

Pollock, M., Johansen, A. M., Roberts, G. O., et al. (2016). On the exact and "-strong simulation of (jump) diffusions. *Bernoulli*, 22(2):794–856.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88.