Hierarchical Monte Carlo Fusion

Ryan Chan

Murray Pollock (Newcastle), Adam Johansen (Warwick), Gareth Roberts (Warwick)

10 July 2020

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:

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

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:

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

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:

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

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 parellel

Fusion for Big Data

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

$$(\quad j \, x) \ / \ \prod_{i=1}^{n} `(x \, j \quad ) \ ( \ ) = \prod_{c=1}^{C} `(x_c \, j \quad) \ (\ )^{\frac{1}{C}}$$

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

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

Fusion for Big Data

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

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

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 parellel

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:

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

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 parellel

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 $c = 1, \dots, C$ 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 parellel

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:

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

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

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:

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

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

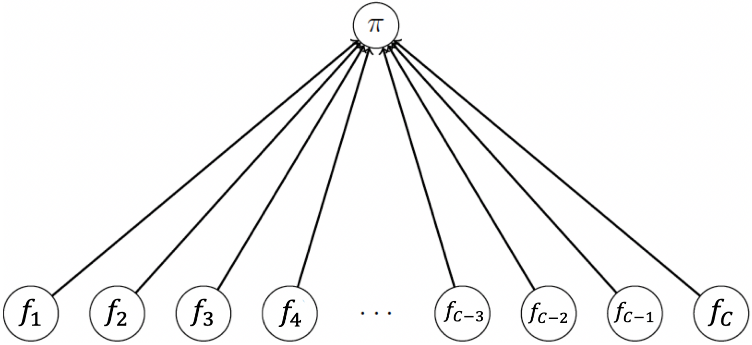
Potential solution:

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

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

Fork-and-join

The fork-and-join approach:



Current Fork-and-Join Methods

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

- Kernel density averaging [Neiswanger et al., 2013]
- Weierstrass sampler [Wang and Dunson, 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

- Kernel density averaging [Neiswanger et al., 2013]
- Weierstrass sampler [Wang and Dunson, 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

- Kernel density averaging [Neiswanger et al., 2013]
- Weierstrass sampler [Wang and Dunson, 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

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

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

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

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)

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

$$dX_t^{(c)} = \nabla \log f_c(X_t^{(c)})dt + dW_t^c,$$

- $W_t^{(c)}$ is d-dimensional Brownian motion
- ∇ 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

$$dX_t^{(c)} = \nabla \log f_c(X_t^{(c)})dt + dW_t^c,$$

$W_t^{(c)}$ is d-dimensional Brownian motion
∇ is the gradient operator over x
Has stationary distribution $f_c^2(x)$

Rejection Sampling (Double Langevin Approach)

Extended Target Density:

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

Consider the proposal density h for the extended target g:

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

- $\bar{x} = \frac{1}{C} \sum_{c=1}^{C} x^{(c)}$
- T is an arbitrary positive constant

Rejection Sampling (Double Langevin Approach)

Extended Target Density:

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

Consider the proposal density h for the extended target g:

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

$\bar{x} = \frac{1}{C} \sum_{c=1}^{C} x^{(c)}$
T is an arbitrary positive constant

Rejection Sampling (Double Langevin Approach)

Simulation from h is easy:

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

1. Simulate $x^{(c)} \sim f_c(x)$ independently
2. Simulate $y \sim N\left(\bar{x}, \frac{T\,\mathrm{I}_d}{C}\right)$

Rejection Sampling (Double Langevin Approach)

Simulation from h is easy:

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

1. Simulate $x^{(c)} \sim f_c(x)$ independently
2. Simulate $y \sim N\left(\bar{x}, \frac{T\,\mathrm{I}_d}{C}\right)$

Rejection Sampling (Double Langevin Approach)

Simulation from h is easy:

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

1. Simulate $x^{(c)} \sim f_c(x)$ independently
2. Simulate $y \sim \mathcal{N}\left(\bar{x}, \frac{T \mathbb{I}_d}{C}\right)$

Rejection Sampling - acceptance probability

Acceptance probability:

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

where

$$:= e^{-\frac{C\ \ ^2}{2T}}; \qquad {}^2 = \frac{1}{C}\sum_{c=1}^{C} \ x^{(c)} \quad \bar{x}\ ^2$$

$$Q := \mathbb{E}_{\bar{\mathbb{W}}}\left[\prod_{c=1}^{C} \exp\left\{ \int_0^T \ {}_c(x_t^{(c)}) \quad \Phi_c \ \mathrm{d}t \right\}\right]$$

where $\bar{\mathbb{W}}$ denotes the law of C independent Brownian bridges $x_t^{(1)},\dots,x_t^{(C)}$ with $x_0 = x^{(c)}$ and $x_T^{(c)} = y$

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

$$\begin{cases} \rho := e^{-\frac{C\sigma^2}{2T}}, & \sigma^2 = \frac{1}{C}\sum_{c=1}^{C} \left\| x^{(c)} - \bar{x} \right\|^2 \\ Q := \mathbb{E}_{\bar{\mathbb{W}}}\left[\prod_{c=1}^{C} \exp\left\{ -\int_0^T \left(\phi_c(x_t^{(c)}) - \Phi_c \right) dt \right\} \right] \end{cases}$$

where $\bar{\mathbb{W}}$ denotes the law of C independent Brownian bridges $x_t^{(1)},\ldots,x_t^{(C)}$ with $x_0 = x^{(c)}$ and $x_T^{(c)} = y$

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

$$\begin{cases} \rho := e^{-\frac{C\sigma^2}{2T}}, \qquad \sigma^2 = \frac{1}{C}\sum_{c=1}^{C}\left(x^{(c)}-\bar{x}\right)^2 \\ Q := \mathbb{E}_{\bar{\mathbb{W}}}\left[\prod_{c=1}^{C}\exp\left\{-\int_0^T \left(\phi_c(x_t^{(c)}) - \Phi_c\right)\mathrm{d}t\right\}\right] \end{cases}$$

where $\bar{\mathbb{W}}$ denotes the law of C independent Brownian bridges $x_t^{(1)},\ldots,x_t^{(C)}$ with $x_0 = x^{(c)}$ and $x_T^{(c)} = y$

Q Acceptance Probability

$$Q := \mathbb{E}_{\bar{\mathbb{W}}}\left[\prod_{c=1}^{C} \exp\left\{-\int_0^T \left(\phi_c(\mathbf{x}_t^{(c)}) - \Phi_c\right) \mathrm{d}t\right\}\right]$$

where

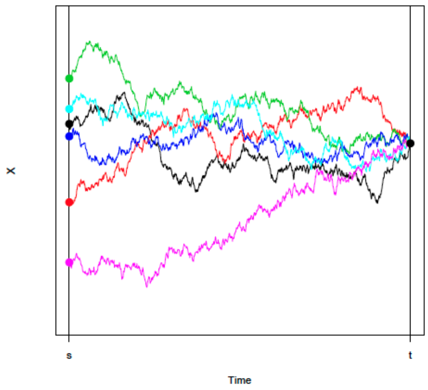
$$\phi_c(\mathbf{x}) = \frac{1}{2}\left(\|\nabla \log f_c(\mathbf{x})\|^2 + \Delta \log f_c(\mathbf{x})\right)$$

Φ_c are constants such that for all $\mathbf{x}$, $\phi_c(\mathbf{x}) \geq \Phi_c$ 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 average $\bar{x}$ of sub-posterior values to a Monte Carlo draw from (x) with acceptance probability Q



Double Langevin Approach - Summary

Proposal:

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

Accept y as a draw from fusion density with probability:

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

Monte Carlo Fusion Algorithm:

1. Simulate $x^{(c)} \sim f_c(x)$ and $y \sim N(\bar{x}, \frac{T\mathbb{I}_d}{C})$
2. Accept y with probability Q

Double Langevin Approach - Summary

Proposal:

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

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 $y \sim N(\bar{x}, \frac{T I_d}{C})$
2. Accept y with probability Q

Double Langevin Approach - Summary

Proposal:

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

Accept y as a draw from fusion density with probability:

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

Monte Carlo Fusion Algorithm:

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

Double Langevin Approach - Summary

Proposal:

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

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 $x^{(c)}$ $f_c(x)$ and y $N(\bar{x}, \frac{T \mathrm{I}_d}{C})$
2. Accept y with probability Q

Limitations of above rejection sampler

1. Scalability: the acceptance probability of Monte Carlo fusion can be small, especially when C is large or d is large
2. Use of simple average $\bar{x}$ of sub-posterior samples as the proposal

   but should we use a weighted average?
3. Use of same time T for each sub-posterior

   but different cores / sub-posteriors could contain different amounts of information
4. PSRS: methodology for PSRS can be computationally expensive

Aim: To construct a fusion algorithm / framework to alleviate some of these limitations

Limitations of above rejection sampler

1. Scalability: the acceptance probability of Monte Carlo fusion can be small, especially when C is large or d is large
2. Use of simple average $\bar{x}$ of sub-posterior samples as the proposal

   but should we use a weighted average?
3. Use of same time T for each sub-posterior

   but different cores / sub-posteriors could contain different amounts of information
4. PSRS: methodology for PSRS can be computationally expensive

Aim: To construct a fusion algorithm / framework to alleviate some of these limitations

Limitations of above rejection sampler

1. Scalability: the acceptance probability of Monte Carlo fusion can be small, especially when C is large or d is large
2. Use of simple average $\bar{x}$ of sub-posterior samples as the proposal

   but should we use a weighted average?
3. Use of same time T for each sub-posterior

   but different cores / sub-posteriors could contain different amounts of information
4. PSRS: methodology for PSRS can be computationally expensive

Aim: To construct a fusion algorithm / framework to alleviate some of these limitations

Limitations of above rejection sampler

1. Scalability: the acceptance probability of Monte Carlo fusion can be small, especially when C is large or d is large
2. Use of simple average $\bar{x}$ of sub-posterior samples as the proposal

   but should we use a weighted average?
3. Use of same time T for each sub-posterior

   but different cores / sub-posteriors could contain different amounts of information
4. PSRS: methodology for PSRS can be computationally expensive

Aim: To construct a fusion algorithm / framework to alleviate some of these limitations

Limitations of above rejection sampler

1. Scalability: the acceptance probability of Monte Carlo fusion can be small, especially when C is large or d is large
2. Use of simple average $\bar{x}$ of sub-posterior samples as the proposal

   but should we use a weighted average?
3. Use of same time T for each sub-posterior

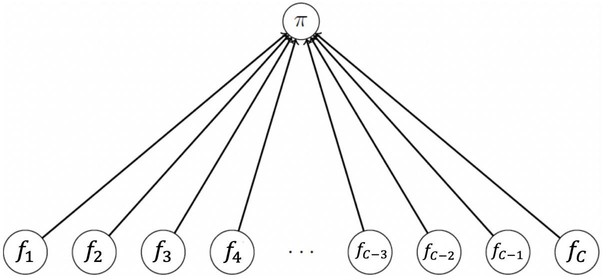
   but different cores / sub-posteriors could contain different amounts of information
4. PSRS: methodology for PSRS can be computationally expensive

Aim: To construct a fusion algorithm / framework to alleviate some of these limitations

Hierarchical Monte Carlo Fusion

1. Scalability with C

The Monte Carlo Fusion algorithm implies a fork-and-join approach:



Hierarchical Monte Carlo Fusion

Solution: Hierarchical Monte Carlo Fusion

- We could perform fusion in a proper divide-and-conquer framework
- Two possible choices are hierarchical (left) and progressive (right) trees

Note: Other trees are possible

Hierarchical Monte Carlo Fusion

Solution: Hierarchical Monte Carlo Fusion

- We could perform fusion in a proper divide-and-conquer framework
- Two possible choices are hierarchical (left) and progressive (right) trees

Note: Other trees are possible

Hierarchical Monte Carlo Fusion

Solution: Hierarchical Monte Carlo Fusion

- We could perform fusion in a proper divide-and-conquer framework
- Two possible choices are hierarchical (left) and progressive (right) trees

Note: Other trees are possible

Hierarchical Monte Carlo Fusion

Solution: Hierarchical Monte Carlo Fusion

- We could perform fusion in a proper divide-and-conquer framework
- Two possible choices are hierarchical (left) and progressive (right) trees

Note: Other trees are possible

Example

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

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

Time-adapting Monte Carlo Fusion

2. Use of simple average x of sub-posterior samples as the proposal
3. Use of same time T for each sub-posterior

Time-adapting Monte Carlo Fusion

Solution: Time-adapting Monte Carlo Fusion

We assign weights to each sub-posterior and use a weighted average: $\bar{x} = \sum_c w_c x^{(c)} / \sum_c w_c$

Time T is adapted for each posterior: $T_c = \frac{T}{w_c}$ for $c = 1, \ldots, C$

Time-adapting Monte Carlo Fusion

Solution: Time-adapting Monte Carlo Fusion

We assign weights to each sub-posterior and use a weighted average: $\tilde{x} = \frac{\sum_c w_c x^{(c)}}{\sum_c w_c}$

Time T is adapted for each posterior: $T_c = \frac{T}{w_c}$ for $c = 1, \ldots, C$

Time-adapting Monte Carlo Fusion

Solution: Time-adapting Monte Carlo Fusion

We assign weights to each sub-posterior and use a weighted average: $\tilde{x} = \sum_c w_c x^{(c)} / \sum_c w_c$

Time T is adapted for each posterior: $T_c = \frac{T}{w_c}$ for $c = 1, \ldots, C$

Time-adapting Monte Carlo Fusion

Time-adapting Monte Carlo Fusion Algorithm:

1. Choose time T and weights for sub-posteriors $w_c; c = 1, \dots, C$
2. Simulate $x^{(c)} \sim f_c(x)$ and $y \sim N(\bar{x}, \frac{T I_d}{\sum_c w_c})$
3. Accept y with probability $\rho^{ta} \cdot Q^{ta}$

Time-adapting Monte Carlo Fusion

Time-adapting Monte Carlo Fusion Algorithm:

1. Choose time T and weights for sub-posteriors $w_c; c = 1, \ldots, C$
2. Simulate $x^{(c)}$ $f_c(x)$ and y $N(\bar{x}; \frac{T I_d}{\sum_c w_c})$
3. Accept y with probability ^{ta} Q^{ta}

Time-adapting Monte Carlo Fusion

Time-adapting Monte Carlo Fusion Algorithm:

1. Choose time T and weights for sub-posteriors $w_c;\ c = 1, \ldots, C$
2. Simulate $x^{(c)} \sim f_c(x)$ and $y \sim \mathcal{N}\left(\bar{x}, \frac{T I_d}{\sum_c w_c}\right)$
3. Accept y with probability $\rho^{ta} \cdot Q^{ta}$

Time-adapting Monte Carlo Fusion

Time-adapting Monte Carlo Fusion Algorithm:

1. Choose time T and weights for sub-posteriors $w_c; c = 1; \ldots; C$
2. Simulate $x^{(c)} \quad f_c(x)$ and $y \quad N \quad (\bar{x}; \frac{T I_d}{\sum_c w_c})$
3. Accept y with probability $^{ta} \quad Q^{ta}$

Divide-and-Conquer SMC

4. PSRS: methodology for PSRS can be computationally expensive

Solution: 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
- Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC

4. PSRS: methodology for PSRS can be computationally expensive

Solution: 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
- Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC

4. PSRS: methodology for PSRS can be computationally expensive

   Solution: 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
   - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC

4. PSRS: methodology for PSRS can be computationally expensive

    Solution: 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
    - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC

4. PSRS: methodology for PSRS can be computationally expensive

Solution: 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
- Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Divide-and-Conquer SMC

4. PSRS: methodology for PSRS can be computationally expensive

    Solution: 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
    - Fits into the Divide-and-Conquer SMC (D&C-SMC) framework by Lindsten et al. [2017]

Logistic Regression Example - Taiwan default payments

Predicting if customers defaulted on their payments using gender and education levels ($n = 30{,}000$ and $d = 5$)

Split into $C = 32$ subsets and apply the following methods:

1. Consensus Monte Carlo [Scott et al., 2016]
2. Kernel density averaging [Neiswanger et al., 2013]
3. Weierstrass rejection sampler [Wang and Dunson, 2013]
4. Weierstrass importance sampler [Wang and Dunson, 2013]
5. Hierarchical Time-adapting SMC Fusion

Logistic Regression Example - Taiwan default payments

Predicting if customers defaulted on their payments using gender and education levels ($n = 30,000$ and $d = 5$)

Split into $C = 32$ subsets and apply the following methods:

1. Consensus Monte Carlo [Scott et al., 2016]
2. Kernel density averaging [Neiswanger et al., 2013]
3. Weierstrass rejection sampler [Wang and Dunson, 2013]
4. Weierstrass importance sampler [Wang and Dunson, 2013]
5. Hierarchical Time-adapting SMC Fusion

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \dots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \dots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \dots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \dots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \ldots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \ldots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \ldots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \ldots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \ldots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \ldots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \dots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \dots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

How to apply Hierarchical Time-adapting SMC Fusion

- Choose a time $T = 0.5$
- Use balanced hierarchical tree where we combine $m = 2$ sub-posteriors at each level, i.e. have $L = 6$ levels in the tree
- Weights are chosen according to how much data they have relative to the bottom level:
  - At $L = 6$: sub-posteriors are given weight $w_c = 1$ for $c = 1, \dots, 32$, i.e. $T_c = 0.5$
  - At $L = 5$: sub-posteriors are given weight $w_c = 2$ for $c = 1, \dots, 16$ (have twice more data than the start), i.e. $T_c = \frac{0.5}{2} = 0.25$
  - and so on up the levels...

Logistic Regression Example - Taiwan default payments

To compare methods we calculate the integrated absolute distance

$$\text{IAD} = \frac{1}{2d} \sum_{j=1}^{d} \int \left| \hat{f}(x_j) - f(x_j) \right| dx_j$$

where $\hat{f}(x_j)$ is the marginal density for x_j based on the method applied and $f(x_j)$ is the benchmark estimate (obtained using NUTS with Stan)

Fix computational cost by restricting run-time allowed for algorithm

Logistic Regression Example - Taiwan default payments

To compare methods we calculate the integrated absolute distance

$$\mathrm{IAD} = \frac{1}{2d} \sum_{j=1}^{d} \int \left| \hat{f}(x_j) - f(x_j) \right| dx_j$$

where $\hat{f}(x_j)$ is the marginal density for x_j based on the method applied and $f(x_j)$ is the benchmark estimate (obtained using NUTS with Stan)

Fix computational cost by restricting run-time allowed for algorithm

Logistic Regression Example - Taiwan default payments

Logistic Regression Example - Taiwan default payments

Ongoing directions

Combining con icting sub-posteriors

Con dential fusion (Con-fusion): where sharing information/data between cores is not permitted

Bayesian Fusion: tailored to big data Bayesian problems

Di erent proposals: e.g. Ornstein-Uhlenbeck bridges, more general Langevin di usion with pre-conditioning matrix for each sub-posterior

Theory for D&C-SMC with Monte Carlo Fusion

Ongoing directions

Combining con icting sub-posteriors

Con dential fusion (Con-fusion): where sharing information/data between cores is not permitted

Bayesian Fusion: tailored to big data Bayesian problems

Di erent proposals: e.g. Ornstein-Uhlenbeck bridges, more general Langevin di usion with pre-conditioning matrix for each sub-posterior

Theory for D&C-SMC with Monte Carlo Fusion

Ongoing directions

- Combining con icting sub-posteriors
- Con dential fusion (Con-fusion): where sharing information/data between cores is not permitted
- Bayesian Fusion: tailored to big data Bayesian problems
- Di erent proposals: e.g. Ornstein-Uhlenbeck bridges, more general Langevin di usion with pre-conditioning matrix for each sub-posterior
- Theory for D&C-SMC with Monte Carlo Fusion

Ongoing directions

Combining conflicting sub-posteriors

Confidential fusion (Con-fusion): where sharing information/data between cores is not permitted

Bayesian Fusion: tailored to big data Bayesian problems

Different proposals: e.g. Ornstein-Uhlenbeck bridges, more general Langevin diffusion with pre-conditioning matrix for each sub-posterior

Theory for D&C-SMC with Monte Carlo Fusion

Ongoing directions

- Combining conflicting sub-posteriors
- Confidential fusion (Con-fusion): where sharing information/data between cores is not permitted
- Bayesian Fusion: tailored to big data Bayesian problems
- Different proposals: e.g. Ornstein-Uhlenbeck bridges, more general Langevin diffusion with pre-conditioning matrix for each sub-posterior
- Theory for D&C-SMC with Monte Carlo Fusion

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.

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass Sampler. *arXiv e-prints*, page arXiv:1312.4605.