Algorithms for unifying statistical inference

Ryan Chan

9 Feb 2021

Outline

Fusion Problem

- Target:

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

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

- Assume we can sample $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$
- Applications:
  - Big Data (by construction)
  - Tempering (by construction)
  - Expert elicitation: combining views of multiple experts
  - Privacy setting

Fusion Problem

- Target:

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

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

- Assume we can sample $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$
- Applications:
  - Big Data (by construction)
  - Tempering (by construction)
  - Expert elicitation: combining views of multiple experts
  - Privacy setting

Fusion Problem

- Target:

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

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

- Assume we can sample $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$
- 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 $\boldsymbol{x}$ with a large number of observations n
- The likelihood $\ell(\boldsymbol{x} \mid \theta)$ becomes expensive to calculate
  - This makes MCMC prohibitively slow for big data
- Potential solution:

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

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

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

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

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

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

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

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

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

  where $\boldsymbol{x}_c$ 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* $\pi_\beta(\boldsymbol{x}) = [\pi(\boldsymbol{x})]^\beta$ for $\beta \in (0, 1]$
- MCMC can become computationally expensive to sample from multi-modal densities and can get stuck in modes
- Potential solution:

$$\pi(\boldsymbol{x}) \propto \pi(\boldsymbol{x})^{\frac{1}{\beta}\cdot\beta} \propto \prod_{c=1}^{\frac{1}{\beta}} \pi_\beta(\boldsymbol{x})$$

where $\frac{1}{\beta} \in \mathbb{N}$

Fusion for Tempering

- Consider the *power-tempered target distribution* $\pi_\beta(\boldsymbol{x}) = [\pi(\boldsymbol{x})]^\beta$ for $\beta \in (0, 1]$
- MCMC can become computationally expensive to sample from multi-modal densities and can get stuck in modes
- Potential solution:

$$\pi(\boldsymbol{x}) \propto \pi(\boldsymbol{x})^{\frac{1}{\beta}\cdot\beta} \propto \prod_{c=1}^{\frac{1}{\beta}} \pi_\beta(\boldsymbol{x})$$

where $\frac{1}{\beta} \in \mathbb{N}$

Fusion for Tempering

- Consider the *power-tempered target distribution* $\pi_\beta(\boldsymbol{x}) = [\pi(\boldsymbol{x})]^\beta$ for $\beta \in (0, 1]$
- MCMC can become computationally expensive to sample from multi-modal densities and can get stuck in modes
- Potential solution:

$$\pi(\boldsymbol{x}) \propto \pi(\boldsymbol{x})^{\frac{1}{\beta} \cdot \beta} \propto \prod_{c=1}^{\frac{1}{\beta}} \pi_\beta(\boldsymbol{x})$$

where $\frac{1}{\beta} \in \mathbb{N}$

Fusion in a privacy setting

- Suppose have C parties that wish to combine their inferences but either:
  - underlying model $f_c(\boldsymbol{x})$ cannot be shared, or
  - underlying data $\boldsymbol{x}_c$ 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 $f_c(\boldsymbol{x})$ cannot be shared, or
  - underlying data $\boldsymbol{x}_c$ 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 $f_c(\boldsymbol{x})$ cannot be shared, or
  - underlying data $\boldsymbol{x}_c$ 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

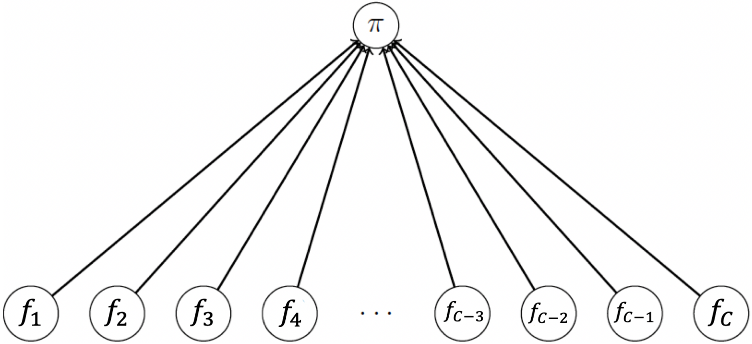
- Suppose have C parties that wish to combine their inferences but either:
  - underlying model $f_c(\boldsymbol{x})$ cannot be shared, or
  - underlying data $\boldsymbol{x}_c$ cannot be shared
- e.g. healthcare settings
- If we have a method that can combine inferences or samples, would need some mechanism 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 $f_c(\boldsymbol{x})$ cannot be shared, or
  - underlying data $\boldsymbol{x}_c$ 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, $\hat{f}_c(\boldsymbol{x})$ [Neiswanger et al., 2013]. Then approximate full posterior by

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

- If Gaussian kernels are used, $\hat{\pi}(\boldsymbol{x})$ 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, $\hat{f}_c(\boldsymbol{x})$ [Neiswanger et al., 2013]. Then approximate full posterior by

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

- If Gaussian kernels are used, $\hat{\pi}(\boldsymbol{x})$ 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, $\hat{f}_c(\boldsymbol{x})$ [Neiswanger et al., 2013]. Then approximate full posterior by

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

- If Gaussian kernels are used, $\hat{\pi}(\boldsymbol{x})$ 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, $\hat{f}_c(\boldsymbol{x})$ [Neiswanger et al., 2013]. Then approximate full posterior by

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

- If Gaussian kernels are used, $\hat{\pi}(\boldsymbol{x})$ 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, $\hat{f}_c(\boldsymbol{x})$ [Neiswanger et al., 2013]. Then approximate full posterior by

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

- If Gaussian kernels are used, $\hat{\pi}(\boldsymbol{x})$ 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

Consensus Monte Carlo

- Approximate the full posterior as a weighted average of the sub-posterior samples [Scott et al., 2016]
- Suppose have MCMC samples $\mathbf{x}_1^{(c)}, \ldots, \mathbf{x}_N^{(c)}$ from $f_c(\mathbf{x})$ for $c = 1, \ldots, C$. Then approximate full posterior

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

  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 $\boldsymbol{x}_1^{(c)}, \ldots, \boldsymbol{x}_N^{(c)}$ from $f_c(\boldsymbol{x})$ for $c = 1, \ldots, C$. Then approximate full posterior

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

  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 $\boldsymbol{x}_1^{(c)}, \ldots, \boldsymbol{x}_N^{(c)}$ from $f_c(\boldsymbol{x})$ for $c = 1, \ldots, C$. Then approximate full posterior

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

 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 $\boldsymbol{x}_1^{(c)}, \dots, \boldsymbol{x}_N^{(c)}$ from $f_c(\boldsymbol{x})$ for $c = 1, \dots, C$. Then approximate full posterior

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

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

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

admits the marginal density π for $\boldsymbol{y}$.

- Main idea: If we can sample from g, then we can can obtain a draw from the fusion density ($\boldsymbol{y} \sim \pi$)

Constructing a rejection sampler - An Extended Target

Proposition

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

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

admits the marginal density π for $\boldsymbol{y}$.

- Main idea: If we can sample from g, then we can can obtain a draw from the fusion density ($\boldsymbol{y} \sim \pi$)

Rejection Sampling (Double Langevin Approach)

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

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \Lambda_c \nabla \log f_c \left(\boldsymbol{x}_t^{(c)} \right) \mathrm{d}t + \Lambda_c^{1/2} \mathrm{d}\boldsymbol{W}_t^{(c)}$$

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

Rejection Sampling (Double Langevin Approach)

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

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \Lambda_c \nabla \log f_c \left(\boldsymbol{x}_t^{(c)} \right) \mathrm{d}t + \Lambda_c^{1/2} \mathrm{d}\boldsymbol{W}_t^{(c)}$$

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

Rejection Sampling (Double Langevin Approach)

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

$$\mathrm{d}\boldsymbol{X}_t^{(c)} = \Lambda_c \nabla \log f_c \left(\boldsymbol{x}_t^{(c)} \right) \mathrm{d}t + \Lambda_c^{1/2} \mathrm{d}\boldsymbol{W}_t^{(c)}$$

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

Rejection Sampling (Double Langevin Approach)

- Extended Target Density:

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

- Consider the proposal density h for the extended target g:

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

  - $\tilde{\boldsymbol{x}} := \left(\sum_{c=1}^{C} \Lambda_c^{-1}\right)^{-1}\left(\sum_{c=1}^{C} \Lambda_c^{-1}\boldsymbol{x}^{(c)}\right)$
  - $\Lambda^{-1} := \frac{1}{T}\sum_{c=1}^{C} \Lambda_c^{-1}$
  - T is an arbitrary positive constant

Rejection Sampling (Double Langevin Approach)

- Extended Target Density:

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

- Consider the proposal density h for the extended target g:

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

  - $\tilde{\boldsymbol{x}} := \left(\sum_{c=1}^{C} \Lambda_c^{-1}\right)^{-1} \left(\sum_{c=1}^{C} \Lambda_c^{-1} \boldsymbol{x}^{(c)}\right)$
  - $\Lambda^{-1} := \frac{1}{T} \sum_{c=1}^{C} \Lambda_c^{-1}$
  - T is an arbitrary positive constant

Rejection Sampling (Double Langevin Approach)

- Simulation from h is easy:

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

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

Rejection Sampling (Double Langevin Approach)

- Simulation from h is easy:

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

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

Rejection Sampling (Double Langevin Approach)

- Simulation from h is easy:

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

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

Rejection Sampling - acceptance probability

- Acceptance probability:

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

where

$$\begin{cases} \rho(\boldsymbol{x}^{(1:C)}) = \exp\left\{-\sum_{c=1}^{C} \frac{(\tilde{\boldsymbol{x}}-\boldsymbol{x}^{(c)})^{\intercal}\Lambda_c^{-1}(\tilde{\boldsymbol{x}}-\boldsymbol{x}^{(c)})}{2T}\right\} \\ Q(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}) := \prod_{c=1}^{C} \mathbb{E}_{\mathbb{W}_{\Lambda_c}}\left[\exp\left\{-\int_0^T \left(\phi_c\left(\boldsymbol{x}_t^{(c)}\right) - \Phi_c\right) \mathrm{d}t\right\}\right] \end{cases}$$

where $\mathbb{W}_{\Lambda_c}$ denotes the law of a Brownian bridge $\{\boldsymbol{x}_t^{(c)}, t \in [0, t]\}$ with $\boldsymbol{x}_0^{(c)} := \boldsymbol{x}^{(c)}$ and $\boldsymbol{x}_T^{(c)} := \boldsymbol{y}$ and covariance matrix Λ_c

Rejection Sampling - acceptance probability

- Acceptance probability:

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

where

$$\begin{cases} \rho(\boldsymbol{x}^{(1:C)}) = \exp\left\{-\sum_{c=1}^{C} \frac{(\tilde{\boldsymbol{x}}-\boldsymbol{x}^{(c)})^\intercal \Lambda_c^{-1}(\tilde{\boldsymbol{x}}-\boldsymbol{x}^{(c)})}{2T}\right\} \\ Q(\boldsymbol{x}^{(1:C)}, \boldsymbol{y}) := \prod_{c=1}^{C} \mathbb{E}_{\mathbb{W}_{\Lambda_c}} \left[\exp\left\{-\int_0^T \left(\phi_c\left(\boldsymbol{x}_t^{(c)}\right) - \Phi_c\right) \mathrm{d}t\right\}\right] \end{cases}$$

where $\mathbb{W}_{\Lambda_c}$ denotes the law of a Brownian bridge $\{\boldsymbol{x}_t^{(c)}, t \in [0, t]\}$ with $\boldsymbol{x}_0^{(c)} := \boldsymbol{x}^{(c)}$ and $\boldsymbol{x}_T^{(c)} := \boldsymbol{y}$ and covariance matrix Λ_c

Q Acceptance Probability

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

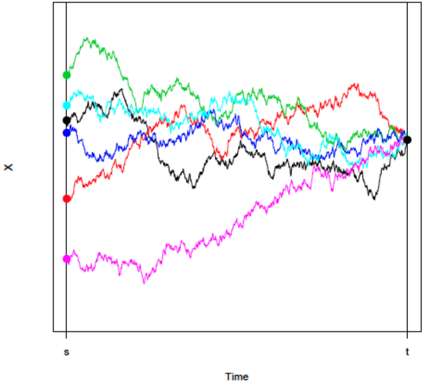
where

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

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



Double Langevin Approach - Summary

- Proposal:

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

- Accept $\boldsymbol{y}$ as a draw from fusion density π with probability:

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

- Monte Carlo Fusion Algorithm:
  1. Simulate $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$ and $\boldsymbol{y} \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Lambda)$
  2. Accept $\boldsymbol{y}$ with probability $\rho \cdot Q$

Double Langevin Approach - Summary

- Proposal:

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

- Accept $\boldsymbol{y}$ as a draw from fusion density π with probability:

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

- Monte Carlo Fusion Algorithm:
  1. Simulate $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$ and $\boldsymbol{y} \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Lambda)$
  2. Accept $\boldsymbol{y}$ with probability $\rho \cdot Q$

Double Langevin Approach - Summary

- Proposal:

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

- Accept $\boldsymbol{y}$ as a draw from fusion density π with probability:

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

- Monte Carlo Fusion Algorithm:
  1. Simulate $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$ and $\boldsymbol{y} \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Lambda)$
  2. Accept $\boldsymbol{y}$ with probability $\rho \cdot Q$

Double Langevin Approach - Summary

- Proposal:

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

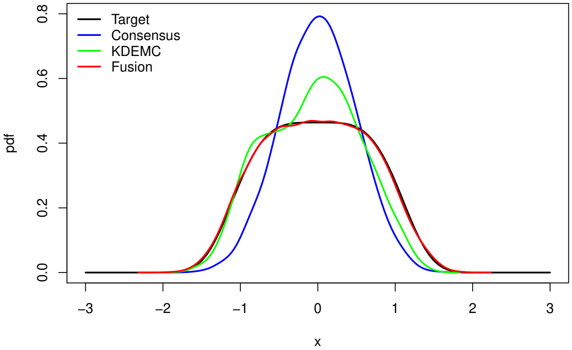
- Accept $\boldsymbol{y}$ as a draw from fusion density π with probability:

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

- Monte Carlo Fusion Algorithm:
  1. Simulate $\boldsymbol{x}^{(c)} \sim f_c(\boldsymbol{x})$ and $\boldsymbol{y} \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Lambda)$
  2. Accept $\boldsymbol{y}$ with probability $\rho \cdot Q$

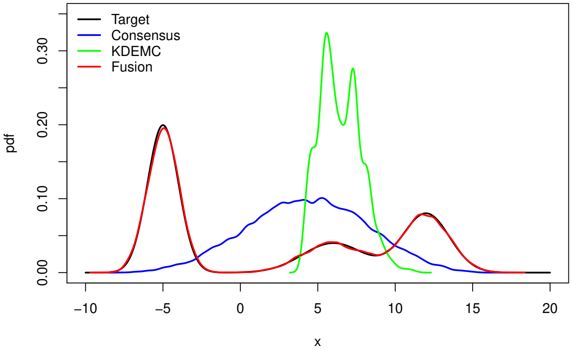
Density with light tails

- Target: $\pi(x) \propto e^{-\frac{x^4}{2}}$
- Sub-posteriors: $f_c(x) \propto e^{-\frac{x^4}{8}}$ for $c = 1, \ldots, 4$
- $N = 20,000$



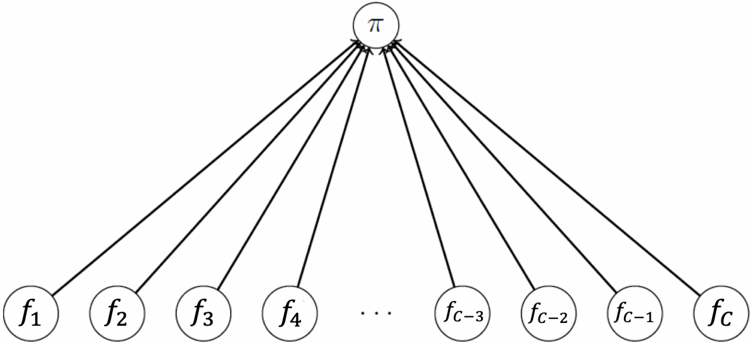
Mixture Gaussian

- Target: $\pi(x) \propto 0.5\mathcal{N}(-5,1) + 0.2\mathcal{N}(6,2) + 0.3\mathcal{N}(12,1.5)$
- Sub-posteriors: $f_c(x) \propto \pi(x)^{1/4}$ for $c = 1, \ldots, 4$
- $N = 20,000$



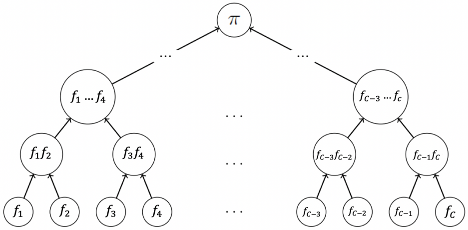
Recall: Fork-and-join

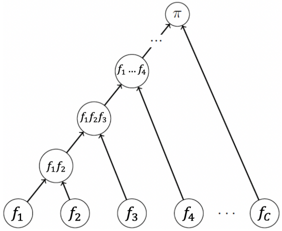
The fork-and-join approach:



Hierarchical Monte Carlo Fusion

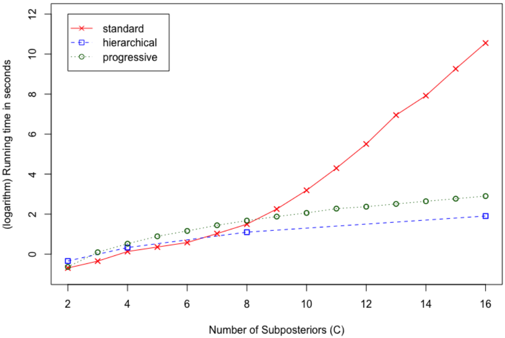
Solution: adopt a divide-and-conquer approach:





Example

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



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

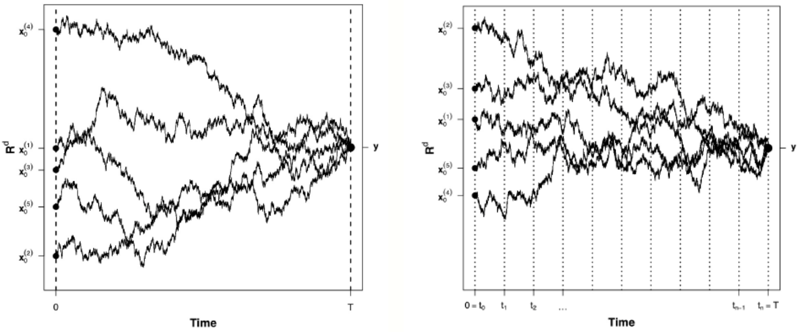
- Can apply Sequential Monte Carlo in the hierarchical fusion framework
- Rejection sampling can be wasteful: large number of proposed samples are rejected
- Motives the use of Sequential Importance Sampling / Resampling ideas
  - Replace the rejection sampling steps with importance sampling steps
  - 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 ε-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.