Principled Statistical Approaches For Sampling and Inference in High Dimensions

Raaz Dwivedi
PhD Advisors: Martin Wainwright and Bin Yu
EECS Department

Dissertation Talk
August 11, 2021
Advances of machine learning

The explosion in data abundance and the compute power is fueling the increase in number of complex ML models and algorithms and the advent of ML in high-stakes domains.
Which model/algorithm to prefer?

Judgment calls become critical and we need principled approaches to gather empirical and theoretical evidence to inform decision making.

- Supervised learning
- Unsupervised learning
- Sampling methods
- Causal inference

Blessed with hold-out accuracy

This dissertation provides principled approaches for choice making where hold-out accuracy is not readily available.
Which model/algorithm to prefer?

Judgment calls become critical and we need principled approaches to gather empirical and theoretical evidence to inform decision making.

- Supervised learning
- Unsupervised learning
- Sampling methods
- Causal inference

This dissertation provides principled approaches for choice making where hold-out accuracy is not readily available.

Blessed with hold-out accuracy

This talk
Drawing samples from probability distributions: A fundamental task!

**Monte Carlo simulation**
(Digital Heart Experiments)

**Bayesian inference**
(Uncertainty quantification)

**Generative modeling**
(generating natural scenarios)

**Numerical integration**
(solving complex PDEs)
Digital heart experiments in computational cardiology

• Digital twin heart experiments try to simulate patient’s response to various therapies in a non-invasive way

• Single-cell calcium signaling model a building block for the tissue- and heart-level models

*Picture credits: Google
Single-cell model

- Single-cell calcium signaling modeled via ODEs, and unknown parameters inferred from observed data using a Bayesian set-up

*Picture credits: Google, Hinch et al. 2004*
Organ-level modeling

- Single-cell model then passed to various tissue and organ-level simulators that take 1000s of CPU hours for single computation

*Picture credits: Google, Hinch et al. 2004
Estimate single-cell model parameters

How to draw random samples from the posterior?

$$\theta_1, \theta_2, \ldots, \theta_n \sim p^*$$

($\theta$ = single cell parameters)
Estimate single-cell model parameters

How to draw random samples from the posterior?

$$\theta_1, \theta_2, \ldots, \theta_n \sim p^*$$
($$\theta$$ = single cell parameters)

Estimate effect of therapy at organ-level

$$\mathbb{E}[g(\theta)] = \int g(\theta)p^*(\theta)d\theta \approx \frac{1}{n} \sum_{i=1}^{n} g(\theta_i)$$
Estimate single-cell model parameters

How to draw random samples from the posterior?

\[ \theta_1, \theta_2, \ldots, \theta_n \sim p^* \]

\( \theta = \text{single cell parameters} \)

Estimate effect of therapy at organ-level

\[ \mathbb{E}[g(\theta)] = \int g(\theta) p^*(\theta) d\theta \approx \frac{1}{n} \sum_{i=1}^{n} g(\theta_i) \]

\( g = \text{Heart simulator takes 1000s of hours} \)
Estimate single-cell model parameters

How to draw random samples from the posterior?

\[ \theta_1, \theta_2, \ldots, \theta_n \sim p^* \]
(\( \theta = \) single cell parameters)

Estimate effect of therapy at organ-level

\[ \mathbb{E}[g(\theta)] = \int g(\theta)p^*(\theta)d\theta \approx \frac{1}{n} \sum_{i=1}^{n} g(\theta_i) \]

\( g = \) Heart simulator takes 1000s of hours

How to compress the points to size \( t \ll n \) while ensuring

\[ \frac{1}{n} \sum_{i=1}^{n} g(\theta_i) \approx \frac{1}{t} \sum_{j=1}^{t} g(\tilde{\theta}_j) \]
Random sampling \[ \theta_1, \theta_2, \ldots, \theta_n \sim p^* \]

- Numerous sampling algorithms proposed, Markov chain Monte Carlo (MCMC) being the most popular

- MCMC method = Set up a Markov chain that converges to the target distribution \( p^* \) as number of steps go to \( \infty \)

1. How many steps do we need to simulate the chain for?

2. How do we tune the Markov chain for fast convergence?
Thinning/compression

- Commonly used: Standard thinning—choose $t$ points uniformly at random, but approximation error gets worse quickly as $t$ reduces

- Other fancier methods: Do not apply to general enough function class

\[ \frac{1}{n} \sum_{i=1}^{n} g(\theta_i) \approx \frac{1}{t} \sum_{j=1}^{t} g(\tilde{\theta}_j) \text{ for } t \ll n \]

1. How to thin without losing information?

2. How to ensure validity for rich enough function class?
Explicit user-friendly guarantees for MCMC methods

Joint work with

Yuansi Chen  Martin Wainwright  Bin Yu
Sampling versus optimization

- Draw samples from the density
  \[ X \sim p^* \propto e^{-f} \]

- Unadjusted Langevin algorithm (ULA)
  \[ X_k = X_{k-1} - h \nabla f(X_{k-1}) + \sqrt{2h} \xi_k \]
  \[ \xi_k \sim \mathcal{N}(0, I_d) \]

- Find mode of the density (or MAP)
  \[ x^* \leftarrow \text{arg max } p^* = \text{arg min } f \]

- Gradient descent
  \[ x_k = x_{k-1} - h \nabla f(x_{k-1}) \]

‘81 Parisi ’94 Grenander-Miller, ’96 Roberts-Tweedie
Langevin algorithms: Origin

- Langevin diffusion

\[ dX_t = -\nabla f(X_t)dt + \sqrt{2}dB_t \]

- Under mild assumptions, diffusion converges to desired distribution

\[ \|P(X_t) - P^*\|_{tv} \to 0 \text{ as } t \to \infty \quad (p^* \propto e^{-f}) \]

- Unadjusted Langevin algorithm: Euler discretization of Langevin diffusion

\[ X_k = X_{k-1} - h \nabla f(X_{k-1}) + \sqrt{2h} \xi_k \]

\[ \xi_k \text{ i.i.d. standard normal} \]
Langevin algorithms: Origin

- Langevin diffusion

\[ dX_t = -\nabla f(X_t)dt + \sqrt{2}dB_t \]

- Under mild assumptions, diffusion converges to desired distribution

\[ \|P(X_t) - P^*\|_{\text{tv}} \to 0 \text{ as } t \to \infty \quad (p^* \propto e^{-f}) \]

- Unadjusted Langevin algorithm: Euler discretization of Langevin diffusion

\[ X_k = X_{k-1} - h \nabla f(X_{k-1}) + \sqrt{2h} \xi_k \]

How to choose \( h \)?

How many steps to take?

\( \xi_k \) i.i.d. standard normal
Langevin simulation: Trade-offs on convergence

Histogram of iterates upon convergence

Target density

Large $h$
ULA

density value

x-value

0.0 0.1 0.2 0.3

-2.5 0.0 2.5 5.0
Langevin simulation: Trade-offs on convergence

Histogram of iterates upon convergence
Langevin simulation: Trade-offs on convergence

Histogram of iterates upon convergence

- **Large $h$ ULA**
- **Small $h$ ULA**
- **Large $h$ ULA + accept-reject**
Langevin simulation: Trade-offs on convergence

Metropolis-adjusted Langevin algorithm (MALA)

1. Proposal step: \( z = x - h \nabla f(x) + \sqrt{2h}\xi \)
2. Accept-reject step: Accept \( z \) with probability

\[
\min \left\{ 1, \frac{e^{-f(x)} \cdot P_h(z \to x)}{e^{-f(z)} \cdot P_h(x \to z)} \right\}
\]
Langevin simulation: Trade-offs for mixing

- Small step ULA
- Large step ULA
- Large step MALA

Error vs. Iteration graph showing the performance of different methods.
Several asymptotic and non-explicit guarantees

- Existence, Harris recurrence
  ['95 Meyn-Tweedie, '96 Roberts-Rosenthal, '00 Diaconis-Holmes-Neal,...]

- Weak convergence and diffusion limits as $d \to \infty$
  ['98 Roberts-Rosenthal, '12 Pillai et al., '10 Beskos et al.,...]

- Geometric and uniform ergodicity, Lyapunov coupling
  ['96 Roberts-Tweedie, '04 Roberts-Rosenthal, '09 Bou-Rabee-Hairer, '16 Livingstone et al.,...]
Several asymptotic and non-explicit guarantees

• Existence, Harris recurrence
  ['95 Meyn-Tweedie, '96 Roberts-Rosenthal, ‘00 Diaconis-Holmes-Neal,…]

• Weak convergence and diffusion limits as $d \to \infty$
  ['98 Roberts-Rosenthal, ‘12 Pillai et al., ‘10 Beskos et al.,…]

• Geometric and uniform ergodicity, Lyapunov coupling

Asymptotic convergence and geometric ergodicity do not immediately reveal user-friendly mixing time bounds in high-dimensions.

Can we characterize dominance of MALA over ULA in a non-asymptotic sense?
Sampling analog of convex optimization

- **Assumption**: Log-concave target density $p^* \propto e^{-f}$ in $\mathbb{R}^d$ with $f$ strongly convex and smooth
  
  $$m_d \leq \nabla^2 f \leq L_d; \quad \kappa = L/m$$

- **Mixing-time guarantee**: Bound on iterations $T$
  with dimension $d$, conditioning $\kappa$, error $\delta$ such that
  
  $$\|P^* - P(X_T)\|_{tv} \leq \delta$$
Non-asymptotic mixing time for Langevin algorithms

$$\| \mathbf{P}^* - \mathbf{P}(X_T) \|_{tv} \leq \delta$$

$p^* \propto e^{-f}$ with $f : \mathbb{R}^d \rightarrow \mathbb{R}$ convex

$$m \|_d \leq \nabla^2 f \leq L \|_d; \quad \kappa = L/m$$

<table>
<thead>
<tr>
<th>ULA ['15 Dalalyan]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Mixing time</th>
</tr>
</thead>
</table>

$$dk^2 \log(1/\delta)$$
Non-asymptotic mixing time for Langevin algorithms

\[ \|P^* - P(X_T)\|_{tv} \leq \delta \]

\[ p^* \propto e^{-f} \text{ with } f: \mathbb{R}^d \to \mathbb{R} \text{ convex} \]

\[ m \|\nabla f\|_d \leq \nabla^2 f \leq L \|\nabla f\|_d; \quad \kappa = L/m \]

<table>
<thead>
<tr>
<th></th>
<th>ULA [’15 Dalalayan]</th>
<th>MALA [Our work]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixing time</td>
<td>(d \kappa^2 \log(1/\delta)/\delta^2)</td>
<td>(d \kappa \log(1/\delta))</td>
</tr>
</tbody>
</table>

**Accept-reject helps**
- Exponentially better dependence on \(\delta\)
- Better dependence on \(\kappa\)
Non-asymptotic mixing time for Langevin algorithms

\[ \|P^* - P(X_T)\|_{tv} \leq \delta \]

\[ p^* \propto e^{-f} \text{ with } f : \mathbb{R}^d \rightarrow \mathbb{R} \text{ convex} \]

\[ m \leq \nabla^2 f \leq L \; , \; \kappa = L/m \]

<table>
<thead>
<tr>
<th></th>
<th>ULA</th>
<th>MALA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mixing time</strong></td>
<td>( dk^2 \log(1/\delta) )  ( \delta^2 )</td>
<td>( dk \log(1/\delta) )</td>
</tr>
<tr>
<td><strong>Step size</strong></td>
<td>( \delta^2 ) ( d\kappa L )</td>
<td>( \frac{1}{dL} )</td>
</tr>
</tbody>
</table>

Accept-reject helps
- Exponentially better dependence on \( \delta \)
- Better dependence on \( \kappa \)

Step size limited by bias in ULA, and by accept-reject step in MALA
Next: How does gradient information help?

\[ \|P^* - P(X_T)\|_{tv} \leq \delta \]

\[ p^* \propto e^{-f} \text{ with } f : \mathbb{R}^d \to \mathbb{R} \text{ convex} \]

\[ m \|\nabla \| \leq \nabla^2 f \leq L \|\nabla \|; \ k = L / m \]

<table>
<thead>
<tr>
<th>Proposal step</th>
<th>Metropolis-adjusted Langevin algorithm (MALA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ z = x - h \nabla f(x) + \sqrt{2h} \xi ] one gradient step</td>
<td></td>
</tr>
<tr>
<td>Mixing time</td>
<td>[ dk \log(1/\delta) ]</td>
</tr>
<tr>
<td>Step size</td>
<td>[ \frac{1}{dL} ]</td>
</tr>
</tbody>
</table>
**MRW: No gradient leads to slower mixing**

\[ \|P^* - P(X_T)\|_{tv} \leq \delta \]

\[ p^* \propto e^{-f} \text{ with } f: \mathbb{R}^d \rightarrow \mathbb{R} \text{ convex} \]

\[ m\|_d \leq \nabla^2 f \leq L\|_d; \ \kappa = L/m \]

<table>
<thead>
<tr>
<th></th>
<th>Metropolis random walk (MRW)</th>
<th>Metropolis-adjusted Langevin algorithm (MALA)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proposal step</strong></td>
<td>( z = x + \sqrt{2h}\xi ) no gradient</td>
<td>( z = x - h\nabla f(x) + \sqrt{2h}\xi ) one gradient step</td>
</tr>
<tr>
<td><strong>Mixing time</strong></td>
<td>( dk^2 \log(1/\delta) )</td>
<td>( dk \log(1/\delta) )</td>
</tr>
<tr>
<td><strong>Step size</strong></td>
<td>( \frac{1}{dkL} )</td>
<td>( \frac{1}{dL} )</td>
</tr>
</tbody>
</table>
HMC: Multiple gradient steps help mix faster

\[ \|P^* - P(X_T)\|_{tv} \leq \delta \]
\[ p^* \propto e^{-f} \text{ with } f : \mathbb{R}^d \to \mathbb{R} \text{ convex} \]
\[ m\|_d \leq \nabla^2 f \leq L\|_d, \quad \kappa = L/m \]

<table>
<thead>
<tr>
<th>Proposal step</th>
<th>Metropolis random walk (MRW)</th>
<th>Metropolis-adjusted Langevin algorithm (MALA)</th>
<th>Metropolis-adjusted Hamiltonian Monte Carlo (HMC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>z = x + \sqrt{2h}\xi \quad \text{no gradient}</td>
<td>z = x - h\nabla f(x) + \sqrt{2h}\xi \quad \text{one gradient step}</td>
<td>Hamiltonian dynamics with ( K ) gradients per step (non-Gaussian proposal)</td>
<td></td>
</tr>
<tr>
<td>Mixing time</td>
<td>( dk^2 \log(1/\delta) )</td>
<td>( dk \log(1/\delta) )</td>
<td>( d^{5/3}\kappa \log(1/\delta) )</td>
</tr>
<tr>
<td>Step size</td>
<td>( \frac{1}{dkL} )</td>
<td>( \frac{1}{dL} )</td>
<td>( \frac{1}{dL^{7/2} L^{1/2}} ) (( K = d^{1/4} ))</td>
</tr>
</tbody>
</table>

Total #gradients = \( d^{11/2} \kappa \log(1/\delta) \)
HMC: Multiple gradient steps help mix faster

\[ ||P^* - P(X_T)||_{tv} \leq \delta \quad p^* \propto e^{-f} \text{ with } f: \mathbb{R}^d \rightarrow \mathbb{R} \text{ convex} \]

\[ m \|d\| \leq \nabla^2 f \leq L \|d\|; \quad \kappa = L/m \]

<table>
<thead>
<tr>
<th></th>
<th>Metropolis random walk (MRW)</th>
<th>Metropolis-adjusted Langevin algorithm (MALA)</th>
<th>Metropolis-adjusted Hamiltonian Monte Carlo (HMC)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proposal step</strong></td>
<td>[ z = x + \sqrt{2h}\xi ] no gradient</td>
<td>[ z = x - h \nabla f(x) + \sqrt{2h}\xi ] one gradient step</td>
<td>Hamiltonian dynamics with ( K ) gradients per step (non-Gaussian proposal)</td>
</tr>
<tr>
<td><strong>Mixing time</strong></td>
<td>[ dk^2 \log(1/\delta) ]</td>
<td>[ dk \log(1/\delta) ]</td>
<td>[ d^{\frac{2}{3}}\kappa \log(1/\delta) ]</td>
</tr>
<tr>
<td><strong>Step size</strong></td>
<td>[ \frac{1}{dkL} ]</td>
<td>[ \frac{1}{dL} ]</td>
<td>[ \frac{1}{d^{\frac{7}{12}}L^{\frac{1}{2}}} (K = d^{\frac{1}{4}}) ]</td>
</tr>
</tbody>
</table>

Total #gradients = \( d^{\frac{11}{12}}\kappa \log(1/\delta) \)

Previous HMC bounds either worse than MALA or had \( 1/\delta^2 \) dependence due to no accept-reject step.
How do the guarantees depend on where you start?

- Distance of initial distribution: $M = \sup_x \frac{p_0(x)}{p_\star(x)}$

- Previous mixing bounds scale $O(\log M)$

$M = O(e^d)$ quite common $\Rightarrow$ Extra $d$ factor in mixing time bounds
How do the guarantees depend on where you start?

- Distance of initial distribution: \( M = \sup_x \frac{p_0(x)}{p_*(x)} \)

- Previous mixing bounds scale \( \Theta(\log M) \)

\[
M = \Theta(e^d) \text{ quite common } \Rightarrow \text{ Extra } d \text{ factor in mixing time bounds}
\]

- We provide an exponential improvement

\[
\Theta(\log \log M) \text{ scaling } \Rightarrow \text{ Starting point doesn't affect much}
\]
Overview of MCMC guarantees

- Better use of gradients leads to faster mixing

ULAs + accept reject exponentially better mixing time

MALA + multiple gradients

HMC

MRW - gradient info

Refs:
1. Log-concave sampling: Metropolis-Hastings algorithms are fast
   [Dwivedi*•Chen*•Wainwright-Yu, ’19 JMLR]
2. Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients
   [Chen-Dwivedi-Wainwright-Yu, ’20 JMLR]
Thinning without losing

1. How to thin without losing information?

2. How to ensure validity for rich enough function class?
Recall: Motivation

Long runs of MCMC often simulated to ensure convergence and mixing

\[
\mathbb{P}^* g := \int g(\theta)p^*(\theta)\,d\theta \approx \frac{1}{n} \sum_{i=1}^{n} g(\theta_i) =: \mathbb{P}_ng \text{ for } x_i \text{'s from Markov Chain}
\]

When evaluating \( g \) expensive, samples often compressed/thinned to save computation
Standard Thinning

\[ x_1, x_2, \ldots, x_n \in \mathbb{R}^d \]

\[ P_{in} := \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \]

\[ \text{Standard-} m \text{ Thinning} \]

\[ y_1, \ldots, y_{n/m} \]

\[ P_{out} := \frac{1}{n/m} \sum_{i=1}^{n/m} \delta_{y_i} \]

Uniform subsample of size \( n/m \)
Standard Thinning

\[ x_1, x_2, \ldots, x_n \in \mathbb{R}^d \]

\[ \mathbb{P}_{in} := \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \]

Standard-\(m\) Thinning

\[ \mathbb{P}_{out} := \frac{1}{n/m} \sum_{i=1}^{n/m} \delta_{y_i} \]

Uniform subsample of size \(n/m\)

Standard thinning guarantee

\[ \sup_{\|g\| \leq 1} |\mathbb{P}_{in}g - \mathbb{P}_{out}g| \lesssim \sqrt{\frac{m}{n}} \]
Standard Thinning

\( x_1, x_2, \ldots, x_n \in \mathbb{R}^d \)

\[ P_{in} := \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \]

Standard-m Thinning

Uniform subsample of size \( n/m \)

\[ y_1, \ldots, y_{n/m} \]

\[ P_{out} := \frac{1}{n/m} \sum_{i=1}^{n/m} \delta_{y_i} \]

Standard thinning guarantee

\[ \sup_{\|g\|\leq 1} |P_{in}g - P_{out}g| \lesssim \sqrt{\frac{m}{n}} \]

Monte Carlo guarantee:

(Input = \( n \) iid or fast mixing MCMC points)

\[ \sup_{\|g\|\leq 1} |P_{in}g - \mathbb{P}^*g| \lesssim \sqrt{\frac{1}{n}} \]
Standard Thinning: **Can not compress too much**

\[ x_1, x_2, \ldots, x_n \in \mathbb{R}^d \]

\[ P_{in} := \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \]

\[ P_{out} := \frac{1}{n/m} \sum_{i=1}^{n/m} \delta_{y_i} \]

**Uniform subsample of size** \(n/m\)

\[ y_1, \ldots, y_{n/m} \]

**Standard thinning guarantee**

\[ \sup_{\|g\| \leq 1} |P_{in}g - P_{out}g| \lesssim \sqrt{\frac{m}{n}} \]

**Monte Carlo guarantee:**

(Input = \(n\) iid or fast mixing MCMC points)

\[ \sup_{\|g\| \leq 1} |P_{in}g - P_{out}^*g| \lesssim \sqrt{\frac{1}{n}} \]

\(m\) has to be a constant to have \(n^{-1/2}\) accuracy after thinning
How can we provably and practically compress much more while keeping $n^{-1/2}$ accuracy?

Via Kernel Thinning!

Joint work with

Lester Mackey
Kernel Thinning: Compress to $\sqrt{n}$ points

$x_1, x_2, \ldots, x_n \in \mathbb{R}^d$

suitable kernels

$\mathbb{P}_{in} := \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}$

Kernel Thinning (KT)

Non-uniform sample of size $\sqrt{n}$

$y_1, \ldots, y_{\sqrt{n}}$

$\mathbb{P}_{KT} := \frac{1}{\sqrt{n}} \sum_{i=1}^{\sqrt{n}} \delta_{y_i}$
Kernel Thinning: Compress to $\sqrt{n}$ points with $n^{-1/2}$ error

$x_1, x_2, \ldots, x_n \in \mathbb{R}^d$

suitable kernels

Kernel Thinning (KT)

Non-uniform sample of size $\sqrt{n}$

Non-uniform sample of size $\sqrt{n}$

$\sup_{\|g\|_{k} \leq 1} \|P_{\text{in}} g - P_{\text{KT}} g \| \lesssim_d \begin{cases} n^{-1/2} \sqrt{\log n} & \text{(Compactly supported)} \\ n^{-1/2} \sqrt{\log^{d+1} n \log \log n} & \text{(Sub-exponential tails)} \end{cases}$
Highlights of kernel thinning

• KT guarantees $n^{-1/2}$ error with $\sqrt{n}$ points, which
  • is significantly superior to $n^{-1/4}$ rates from Standard-$\sqrt{n}$ Thinning
  • applies to arbitrary functions in infinite-dimensional reproducing kernel Hilbert spaces (RKHS), and fairly generic input points (including MCMC points)
  • The algorithm requires only kernel evaluations for implementation
Effect of high dimensions on KT

IID input, Gaussian target distribution with Gaussian kernel
Summary: Thinning a lot without losing!

$n$ points $\rightarrow$ **Standard-$\sqrt{n}$ Thinning** $\rightarrow$ $\sqrt{n}$ points with $n^{-1/4}$ error

$n$ points $\rightarrow$ **Kernel Thinning** $\rightarrow$ $\sqrt{n}$ points with $n^{-1/2}$ error

Like Quasi Monte Carlo but applicable more widely

Nearly minimax integration error in many settings

Nearly optimal $L^\infty$-error

Refs: Kernel Thinning [Dwivedi-Mackey, ’21 COLT]
Going back to the cardiac experiments
Estimate single-cell model parameters

How to draw random samples from the posterior?

\[ \theta_1, \theta_2, \ldots, \theta_n \sim p^* \text{ (38-dimensional)} \]
\[ (\theta = \text{single cell parameters}) \]

Estimate effect of therapy at organ-level

\[
\mathbb{E}[g(\theta)] = \int g(\theta)p^*(\theta)d\theta \approx \frac{1}{n} \sum_{i=1}^{n} g(\theta_i)
\]
\[ g = \text{Heart simulator takes 1000s of hours} \]

How to compress the points to size \( t \ll n \) while ensuring

\[
\frac{1}{n} \sum_{i=1}^{n} g(\theta_i) \approx \frac{1}{t} \sum_{j=1}^{t} g(\tilde{\theta}_j)
\]
Estimate single-cell model parameters

How to draw random samples from the posterior?

$$\theta_1, \theta_2, \ldots, \theta_n \sim p^* \text{ (38-dimensional)}$$

$$\theta \text{ = single cell parameters}$$

Estimate effect of therapy at organ-level

$$\mathbb{E}[g(\theta)] = \int g(\theta)p^*(\theta)d\theta \approx \frac{1}{n} \sum_{i=1}^{n} g(\theta_i)$$

$$g = \text{Heart simulator takes 1000s of hours}$$

How to compress the points to size $$t \ll n$$ while ensuring

$$\frac{1}{n} \sum_{i=1}^{n} g(\theta_i) \approx \frac{1}{t} \sum_{j=1}^{t} g(\tilde{\theta}_j)$$
Estimate single-cell model parameters

How to draw random samples from the posterior?

\[ \theta_1, \theta_2, \ldots, \theta_n \sim p^* \quad \text{(38-dimensional)} \]

\( \theta = \text{single cell parameters} \)

MCMC samples fed to thinning methods

Error in Gaussian-RKHS

\( \text{standard: } n^{-0.27}, \quad \text{KT: } n^{-0.55} \)

(*MCMC samples taken from Riabiz et al. 2020)
Summary: Generating and thinning MCMC samples

- Guarantees for MCMC sampling:
  - finite time benefits of accept-reject step, and gradients
  - *Not covered*: Sampling under constraints

- New thinning methods that discard samples effectively:
  - without losing information for rich function classes
  - *Not covered*: Thin a little, and gain a lot

Ref: Fast MCMC sampling on polytopes
[Chen*-Dwivedi*-Wainwright-Yu, ’18 JMLR]

Ref: The power of online thinning in reducing discrepancy
[Dwivedi-Feldheim-Gurevich-Ramdas, ’19 PTRF]
Which model/algorithm to prefer?

Judgment calls become critical and we need principled approaches to gather empirical and theoretical evidence to inform decision making.

- Supervised learning
  - Blessed with hold-out accuracy
- Unsupervised learning
- Sampling methods
  - This talk
- Causal inference
  - This dissertation provides principled approaches for choice making where hold-out accuracy is not readily available
Acknowledgments