Kernel Thinning

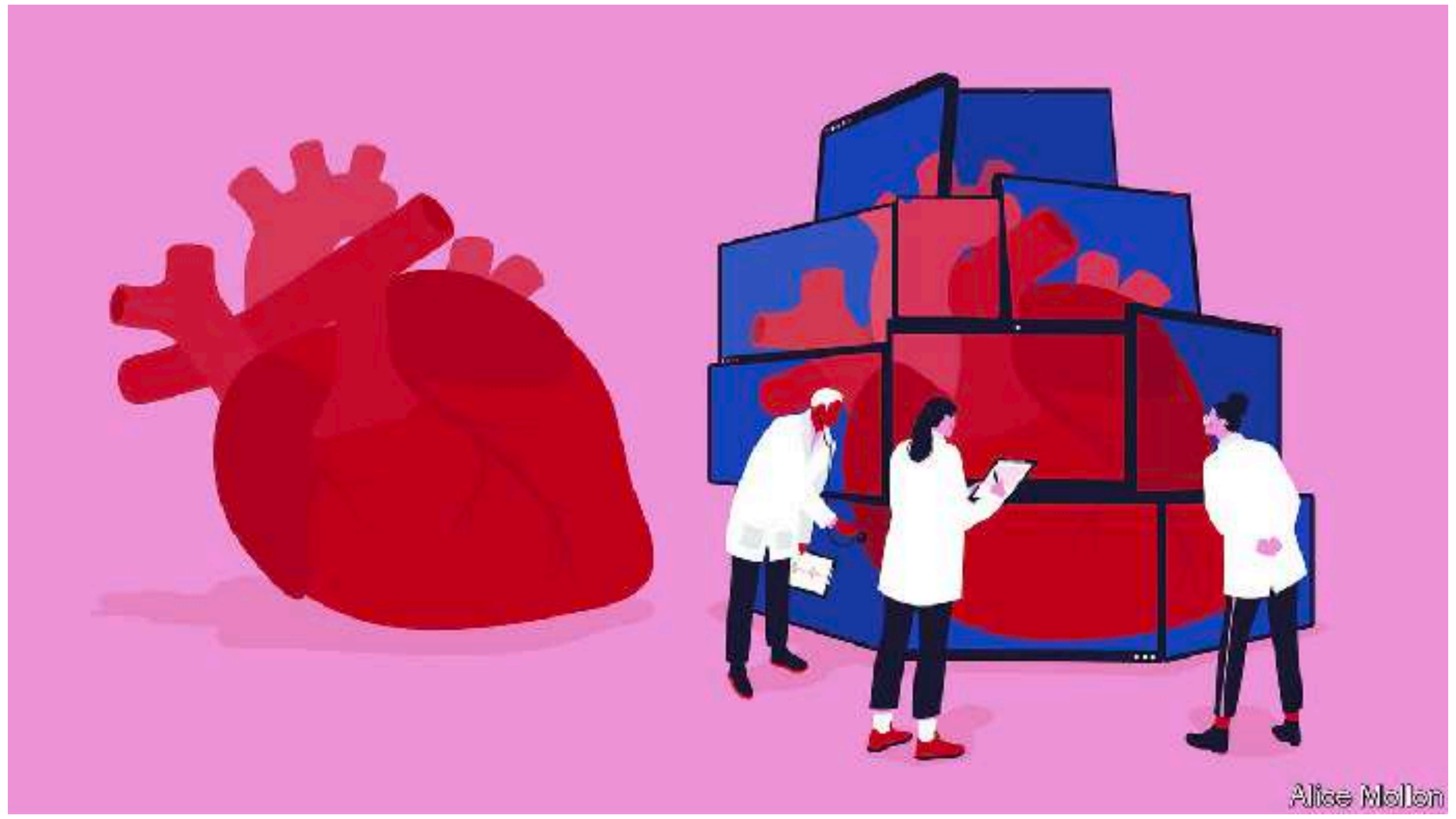
Raaz Dwivedi, Lester Mackey
raaz.rsk@mit.edu

MIT EECS
Harvard John A. Paulson School of Engineering and Applied Sciences
Microsoft Research

Talk at Data-Centric Engineering Reading Group
The Alan Turing Institute
Sep 29, 2021

Motivation: Computational Cardiology

- **Digital twin heart**: Develop a model for human heart to predict disease progression, and therapy response in a non-invasive way

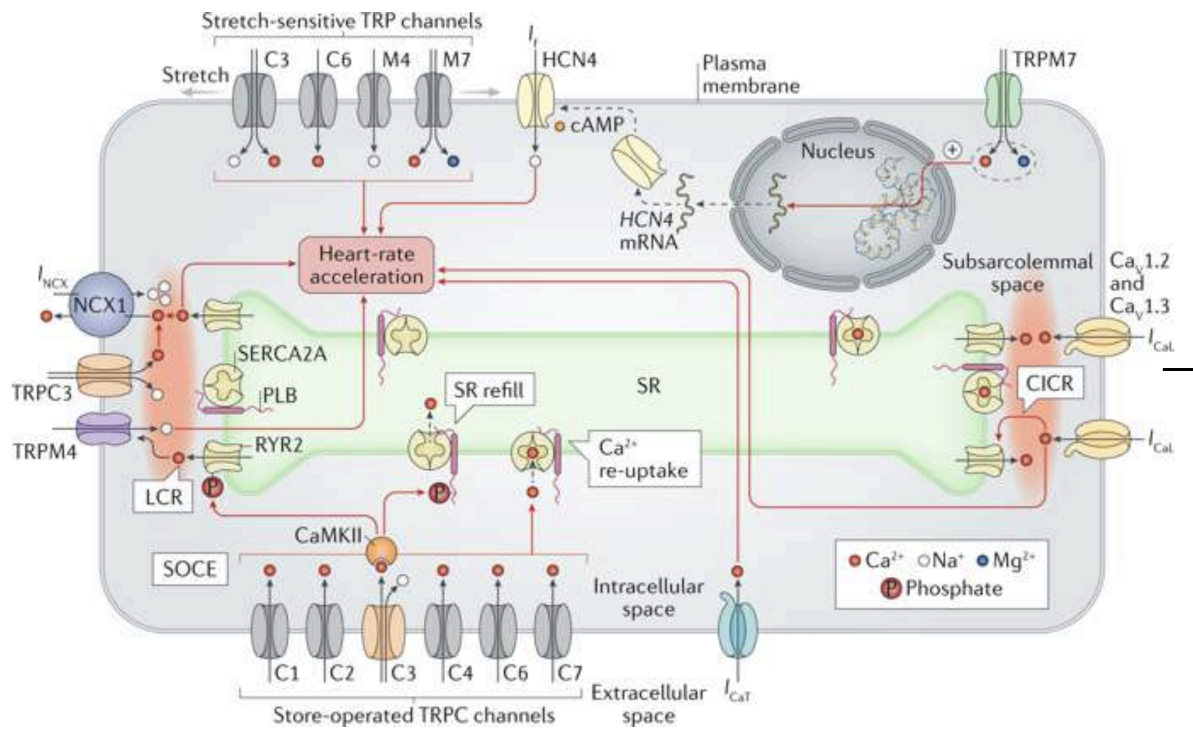


Heart beat and arrhythmias

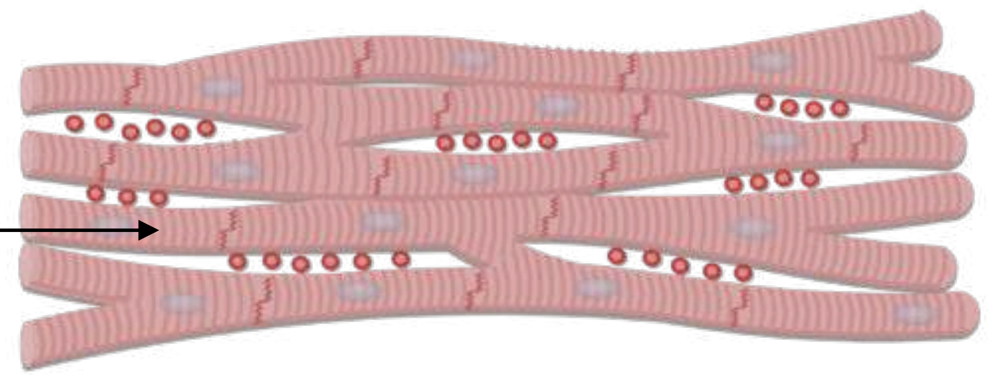
- Calcium signaling in heart cells affects whole-organ heartbeats; its dysregulation can lead to life-threatening heart arrhythmias

- A key task: Model the effect of calcium-signaling dysregulation on heart function

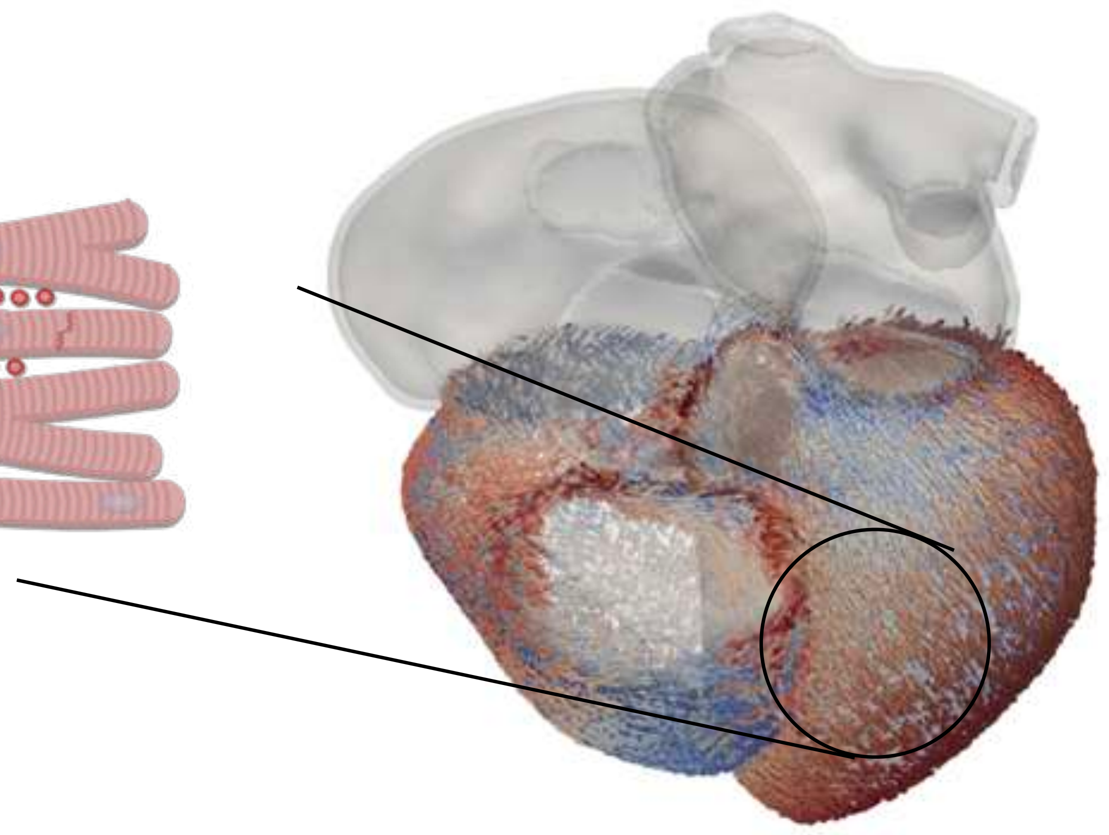
Single cell scale



Tissue scale



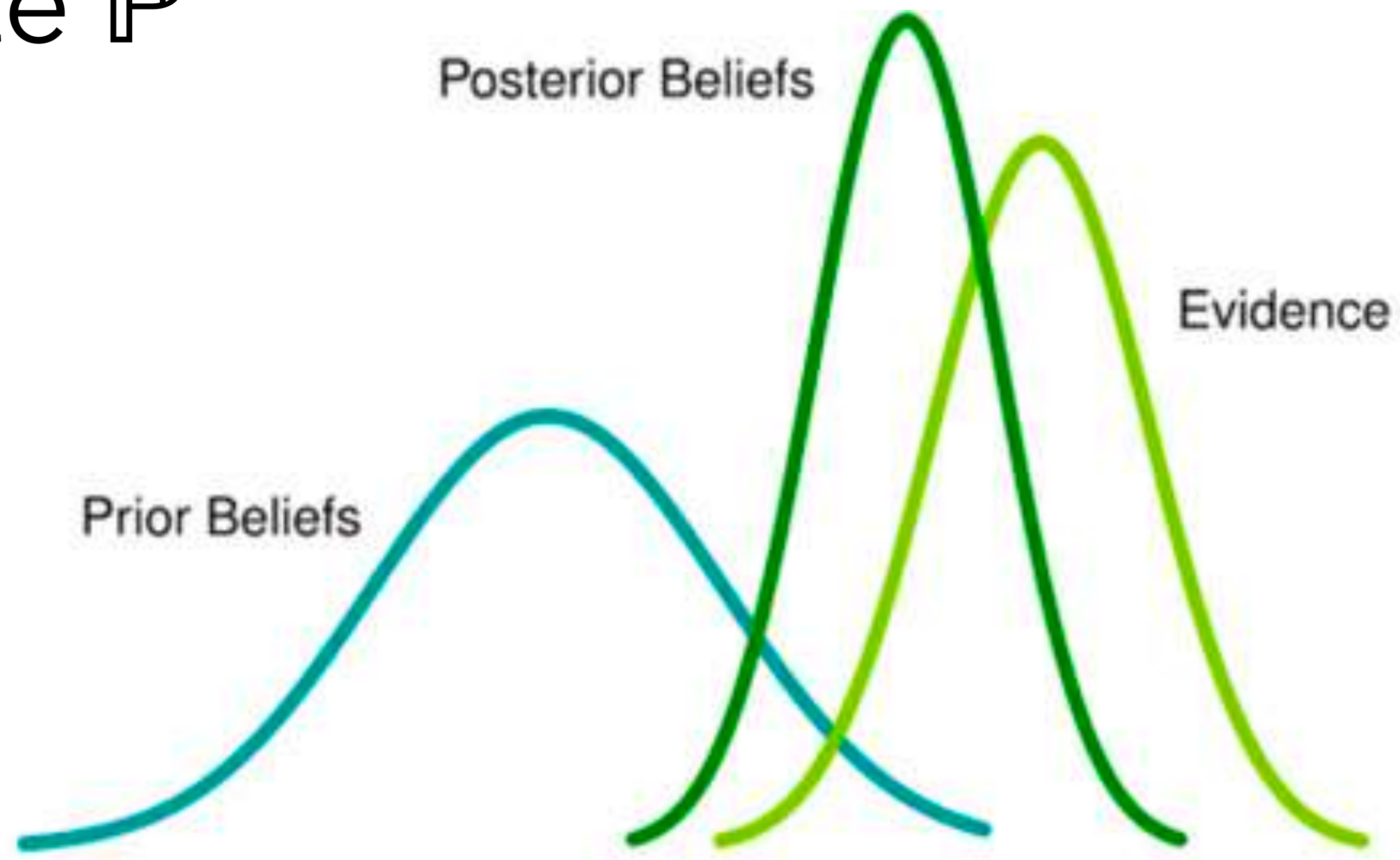
Organ scale



Pic credits: Marina Riabiz

An inferential pipeline

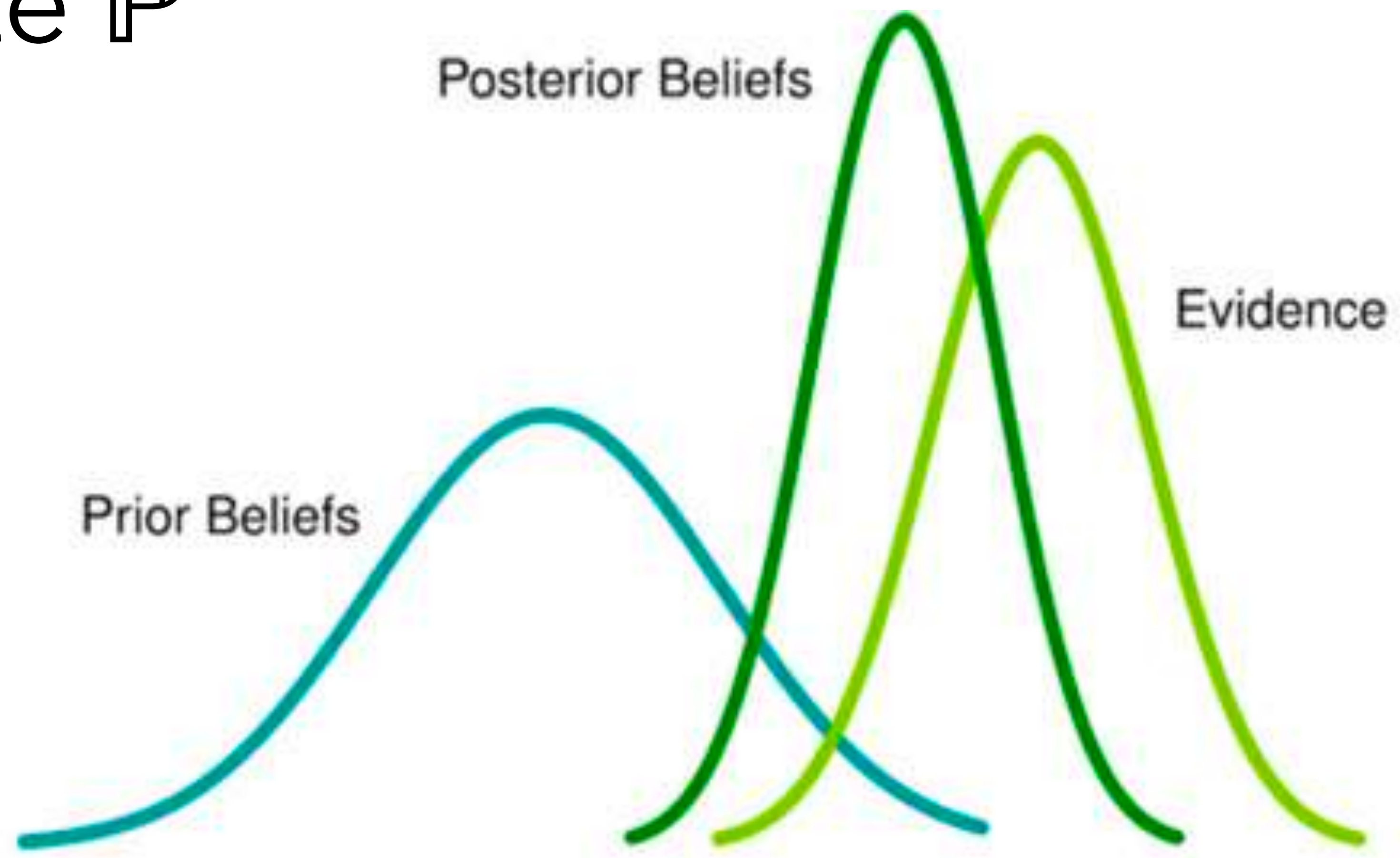
- Estimate calcium signaling model parameters from the patient data via Bayesian set-up (a 38-dimensional differential equation model, Hinch et al. 2004)
- Inference based on posterior $\mathbb{P}^\star$ approximated via drawing samples from Markov Chain Monte Carlo (MCMC)
- Millions of samples often drawn to faithfully approximate $\mathbb{P}^\star$
- Uncertainty propagated to whole-heart model by simulating for each sample for the calcium model



*Picture credits: Google

An inferential pipeline

- Estimate calcium signaling model parameters from the patient data via Bayesian set-up (a 38-dimensional differential equation model, Hinch et al. 2004)
- Inference based on posterior $\mathbb{P}^\star$ approximated via drawing samples from Markov Chain Monte Carlo (MCMC)
- Millions of samples often drawn to faithfully approximate $\mathbb{P}^\star$
- Uncertainty propagated to whole-heart model by simulating for each sample for the calcium model



Challenge: Each organ-level simulator requires 1000s of CPU hours

*Picture credits: Google

Goal:

Represent $\mathbb{P}^\star$ using a few high quality points $(x_i)_{i=1}^n$

$$\mathbb{P}^\star f := \int f(x) d\mathbb{P}^\star(x) \approx \frac{1}{n}\sum_{i=1}^n f(x_i) =: \mathbb{P}_n f$$

Finding representative points for $\mathbb{P}^\star$

- I.I.D. sampling from $\mathbb{P}^\star$ (whenever feasible)

- MCMC sampling from a chain converging to $\mathbb{P}^\star$, common for intractable distributions including posteriors

- **Issue:** Require too many points for a suitable accuracy

  $\left|\mathbb{P}^\star f - \mathbb{P}_n f\right| = \Theta(n^{-1/2})$ – the root-n Monte Carlo rate

  e.g., around a million points for 0.1% error

- Prohibitive for expensive downstream tasks, when f evaluation is expensive

A possibility: Data compression

- What if we approximate $\mathbb{P}_n$ itself using fewer points?
  Distribution approximation ≈ Data compression

- Standard solutions:
  - Uniform / i.i.d. thinning
  - Standard thinning: Choose every $t-$th point
  - **Issue**: Accuracy degrades with compression; $\Theta(\sqrt{t/n})$ worst-case error, same as the Monte Carlo rate with n/t points
  - E.g., With $\sqrt{n}$ points $\Theta(n^{-1/4})$ error

Can we do better?

Minimax lower bounds

- There exists some $\mathbb{P}^\star$ such that the worst-case integration error
    - Is $\Omega(n^{-1/2})$ for any compression scheme returning $\sqrt{n}$ points
      [Philips and Tai, 2020]
    - Is $\Omega(n^{-1/2})$ for any approximation based on $\sqrt{n}$ i.i.d. points
      [Tolstikhin, Sriperumbudur, and Muandet, 2017]

This talk: Kernel thinning

A new algorithm that **provably and practically compresses** with rates matching the minimax rates up to log-factors

Problem set-up

- **Input**:

  Points $(x_i)_{i=1}^n \subset \mathbb{R}^d$ with empirical distribution $\mathbb{P}_n \triangleq \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$

  Target output size s ($s = \sqrt{n}$ for heavy compression)

- **Goal:**

  Return a subset of input points with size s, empirical distribution $\mathbb{P}_{out}$ with worst-case error $o(s^{-1/2})$, i.e., better than Monte Carlo rate

Error metric: Maximum mean discrepancy

- kernel maximum mean discrepancy (MMD) = worst-case integration discrepancy between two distributions over a class of real-valued test functions

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{out}) = \sup_{\|f\|_{\mathbf{k}} \leq 1} |\mathbb{P}_{in} f - \mathbb{P}_{out} f|$$

[Gretton-Borgwardt-Rasch-Schölkopf-Smola, 2012]

- This class = the unit ball of reproducing kernel Hilbert space of some positive definite reproducing kernel $\mathbf{k}$, e.g.,
  - Gaussian kernel: $\mathbf{k}(x, y) = \exp\left(-\frac{1}{2}\|x - y\|^2\right)$
  - Inverse multiquadric (IMQ) kernel: $\mathbf{k}(x, y) = \frac{1}{(1 + \|x - y\|_2^2)^{1/2}}$

Error metric: Maximum mean discrepancy

- kernel maximum mean discrepancy (MMD) = worst-case integration discrepancy between two distributions over a class of real-valued test functions

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{out}) = \sup_{\|f\|_{\mathbf{k}} \leq 1} |\mathbb{P}_{in} f - \mathbb{P}_{out} f|$$

[Gretton-Borgwardt-Rasch-Schölkopf-Smola, 2012]

- This class = the unit ball of reproducing kernel Hilbert space of some positive definite reproducing kernel $\mathbf{k}$, e.g.,
  - Gaussian kernel: $\mathbf{k}(x, y) = \exp\left(-\frac{1}{2}\|x - y\|^2\right)$
  - Inverse multiquadric (IMQ) kernel: $\mathbf{k}(x, y) = \frac{1}{(1 + \|x - y\|_2^2)^{1/2}}$

MMD metrizes convergence in distribution for popular infinite-dimensional kernels like Gaussian, Matern, IMQ, B-spline

[Simon-Gabriel-Barp-Schölkopf-Mackey, 2020]

Error metric: Maximum mean discrepancy

- kernel maximum mean discrepancy (MMD) = worst-case integration discrepancy between two distributions over a class of real-valued test functions

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{out}) = \sup_{\|f\|_{\mathbf{k}} \leq 1} |\mathbb{P}_{in} f - \mathbb{P}_{out} f|$$

[Gretton-Borgwardt-Rasch-Schölkopf-Smola, 2012]

$$|\mathbb{P}_{in} g - \mathbb{P}_{out} g| \leq \|g\|_{\mathbf{k}} \cdot \mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{out})$$

Kernel Thinning

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

smooth decaying $\mathbf{k}$

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

→ **Kernel Thinning (KT)** →

Non-uniform sub-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: $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ error

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

smooth decaying $\mathbf{k}$

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

→ **Kernel Thinning (KT)** →

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

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

$$\mathbb{P}_{KT} := \frac{1}{\sqrt{n}}\sum_{i=1}^{\sqrt{n}} \delta_{y_i}$$

With high probability over the randomness in KT, we have

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{KT}) \lesssim_d \begin{cases} n^{-1/2}\sqrt{\log n} & \text{(Compactly supported; e.g., B-spline } \mathbf{k}\text{)} \\ n^{-1/2}\sqrt{\log^{d/2+1} n \log\log n} & \text{(Sub-Gaussian tails; e.g., Gaussian } \mathbf{k}\text{)} \\ n^{-1/2}\sqrt{\log^{d+1} n \log\log n} & \text{(Sub-exponential tails; e.g., Matern } \mathbf{k}\text{)} \end{cases}$$

Kernel Thinning: $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ error

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

smooth decaying $\mathbf{k}$

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

→ **Kernel Thinning (KT)** →

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

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

$\mathbb{P}_{KT} := \frac{1}{\sqrt{n}}\sum_{i=1}^{\sqrt{n}}\delta_{y_i}$

With high probability over the randomness in KT, we have

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{KT}) \lesssim_d \begin{cases} n^{-1/2}\sqrt{\log n} & \text{(Compactly supported; e.g., B-spline } \mathbf{k}\text{)} \\ n^{-1/2}\sqrt{\log^{d/2+1} n \log\log n} & \text{(Sub-Gaussian tails; e.g., Gaussian } \mathbf{k}\text{)} \\ n^{-1/2}\sqrt{\log^{d+1} n \log\log n} & \text{(Sub-exponential tails; e.g., Matern } \mathbf{k}\text{)} \end{cases}$$

Significantly **faster than the Monte Carlo rate of $\Omega(n^{-1/4})$ with $\sqrt{n}$ points** from iid or standard thinning

Kernel Thinning: $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ error

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

smooth decaying $\mathbf{k}$

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

→ **Kernel Thinning (KT)** →

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

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

$$\mathbb{P}_{KT} := \frac{1}{\sqrt{n}}\sum_{i=1}^{\sqrt{n}} \delta_{y_i}$$

With high probability over the randomness in KT, we have

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}^\star, \mathbb{P}_{KT}) \precsim_d \begin{cases} n^{-1/2}\sqrt{\log n} & \text{(Compactly supported; e.g., B-spline } \mathbf{k}\text{)} \\ n^{-1/2}\sqrt{\log^{d/2+1} n \log\log n} & \text{(Sub-Gaussian tails; e.g., Gaussian } \mathbf{k}\text{)} \\ n^{-1/2}\sqrt{\log^{d+1} n \log\log n} & \text{(Sub-exponential tails; e.g., Matern } \mathbf{k}\text{)} \end{cases}$$

Whenever $\mathbf{MMD}_{\mathbf{k}}(\mathbb{P}^\star, \mathbb{P}_{in}) \precsim n^{-1/2}$ (holds for $\mathbb{P}_{in}$ with iid or fast mixing MCMC points)

Kernel Thinning: $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ error

- Rates are like Quasi-Monte Carlo, but apply to more general distributions and on $\mathbb{R}^d$
- See the paper for explicit constants with dependence on kernel hyper-parameters
- More generally, with input size n and output size s, the MMD error is
$$\widetilde{O}\left(\frac{s}{n}\right) \text{ for } s \geq \sqrt{n}$$

Related work: $\sqrt{n}$ points with $n^{-1/4}$ MMD

- **Known guarantees *no better* than Monte Carlo rate**:
  Standard thinning iid points [Tolstikhin-Sriperumbudur-Muandet, 2017]
  Standard thinning geometrically ergodic MCMC [Dwivedi-Mackey 2021]
  Kernel herding for infinite-dimensional kernels [Chen-Welling-Smola 2010, Lacoste-Julien-Lindsten-Bach 2015]
  Stein Points MCMC [Chen-Barp-Briol-Gorham-Girolami-Mackey-Oates, 2019]
  Greedy sign selection [Karnin-Liberty 2019]

- ***Unknown* guarantees**:
  Support points [Mak-Joseph 2018]
  Supersampling from a reservoir [Paige-Sejdinovic-Wood, 2016]:

Related work: Better than Monte Carlo MMD

- **Finite-dimensional linear kernels:** Discrepancy construction [Harvey and Samadi, 2014]

- **Uniform $\mathbb{P}^\star$ on $[0,1]^d$ (*bounded* support):**
  Quasi Monte Carlo [Hickernell 1998, Novak-Wozniakowski 2010] ,
  Haar thinning [Dwivedi-Feldheim-Gurel-Gurevich-Ramdas 2019]

- **$\mathbb{P}^\star$ with *bounded* support with known $\mathbb{P}^\star\mathbf{k}$:**
  Bayesian quadrature [O'Hagan 1991]
  Bayes' Sard cubtature [Karvonen et al. 2018]
  Determinantal point processes [Belhadji et al. 2020]

- **($\mathbf{k}$, $\mathbb{P}^\star$) with *known/bounded* eigenfunctions:**
  Determinantal point process kernel quadrature [Belhadji et al. 2019]
  Black-box importance sampling [Liu et al. 2018]

Kernel thinning advantages

1. $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$-MMD error (iid sampling gives $\Omega(n^{-1/4})$ error)
2. Valid for **non-uniform** target distributions with **unbounded support**
3. Valid for **infinite-dimensional** smooth/decaying kernels
4. Valid for **generic input points** including iid/MCMC/quadrature/herding with mild conditions
5. Requires **only kernel evaluations** to implement
6. **Matches MMD lower bounds** up to log factors
7. **Matches L^∞-error lower bounds** up to log factors

Kernel Thinning in Action

Kernel Thinning: IID experiments

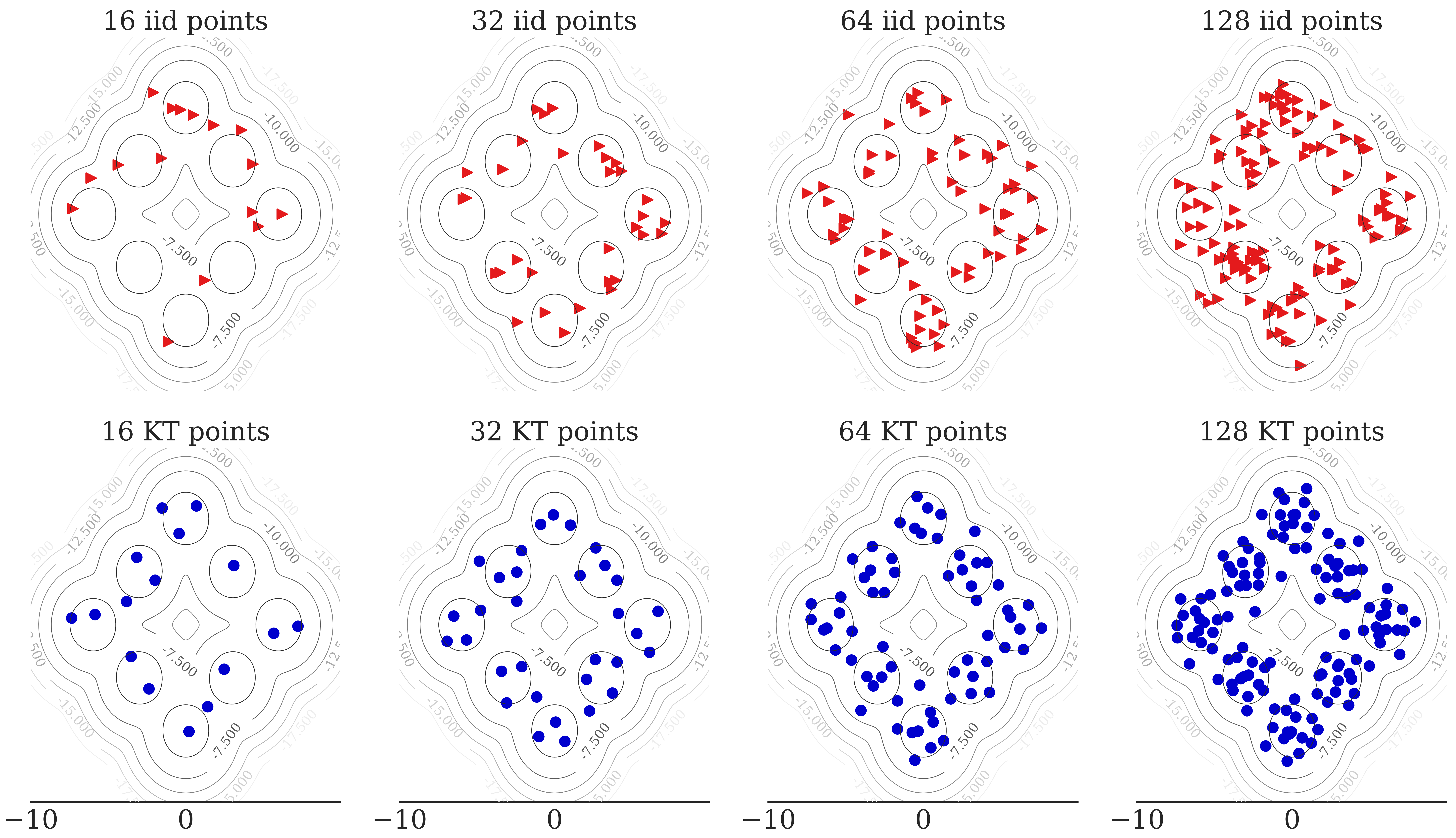
- n i.i.d. input points from $\mathbb{P}^\star$, $\sqrt{n}$ output points from standard & kernel thinning
- Kernel thinning run using Gaussian / squared-exponential kernel

$$\mathbf{k}(x, y) = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|_2^2\right)$$

- Error metric: Kernel maximum mean discrepancy (MMD), i.e., the worst-case integration error in the unit ball of RKHS

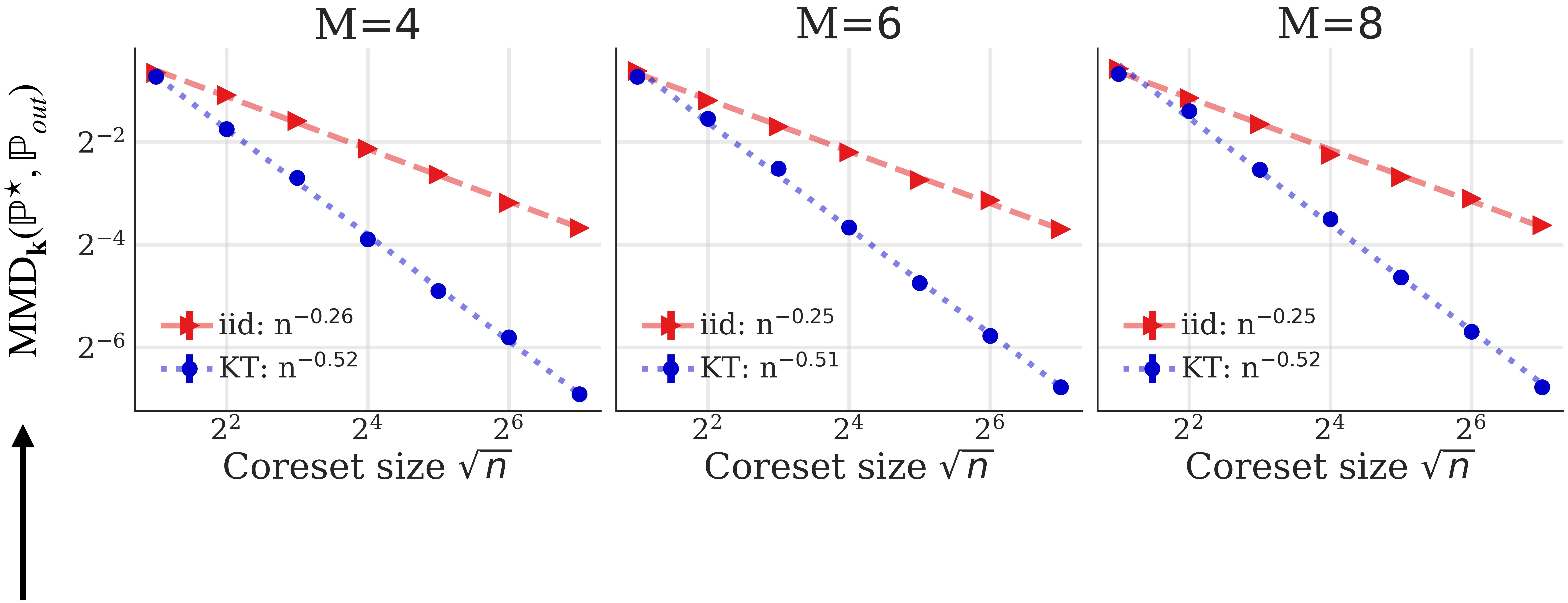
$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}^\star, \mathbb{P}_{out}) := \sup_{\|f\|_{\mathbf{k}}\leq 1} |\mathbb{P}^\star f - \mathbb{P}_{out} f|$$

Visualizing IID and KT points for mixture of Gaussians



Mixture of M Gaussians: $n^{-1/2}$ MMD-error for KT

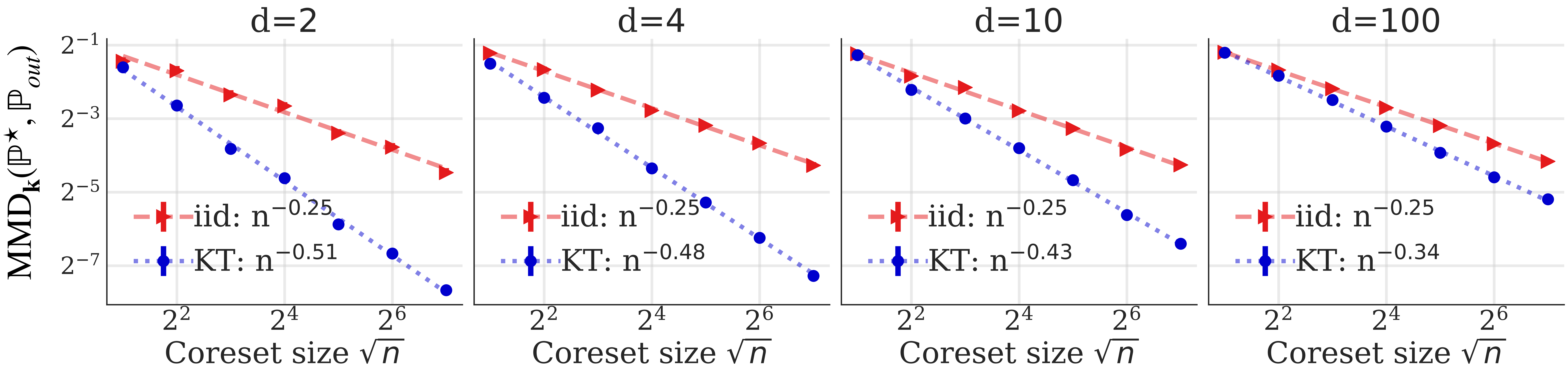
(with Gaussian kernel)



(Worst-case integration error in the unit ball of Gaussian RKHS)

KT vs iid: Gaussian $\mathbb{P}^\star$ in $\mathbb{R}^d$

(Gaussian kernel with $\sigma^2 = 2d$)



In practice, significant MMD gains even in dimension $d = 100$

MCMC experiments: Differential equation models

Dimension $d = 4$

1. Lotka-Volterra model
   oscillatory enzymatic control, [1925, 1926]
2. Goodwin model
   oscillatory predator-prey evolution, [1965]

MCMC experiments: Differential equation models

Dimension $d = 4$

1. Lotka-Volterra model
   oscillatory enzymatic control, [1925, 1926]
2. Goodwin model
   oscillatory predator-prey evolution, [1965]

x

1. Posterior

MCMC experiments: Differential equation models

Dimension $d = 4$

1. Lotka-Volterra model
   oscillatory enzymatic control, [1925, 1926]
2. Goodwin model
   oscillatory predator-prey evolution, [1965]

x

1. Posterior

x

1. Random walk (RW)
   [Metropolis et al. 1953, Hastings 1970]
2. Adaptive random walk (adaRW)
   [Haario et al. 1999]
3. Metropolis adjusted Langevin algorithm (MALA) [Roberts et al. 1996]
4. Preconditioned-MALA (pMALA)
   [Girolami et al. 2011]

MCMC samples taken from Riabiz-Chen-Cockayne-Swietach-Niederer-Mackey-Oates, 2021

MCMC experiments: Differential equation models

Dimension $d = 4$

1. Lotka-Volterra model
   oscillatory enzymatic control, [1925, 1926]
2. Goodwin model
   oscillatory predator-prey evolution, [1965]

Dimension $d = 38$

3. Hinch calcium signal model
   [Hinch-Greenstein-Tanskanen-Xu-Winslow, 2004]

x

1. Posterior

x

1. Random walk (RW)
   [Metropolis et al. 1953, Hastings 1970]
2. Adaptive random walk (adaRW)
   [Haario et al. 1999]
3. Metropolis adjusted Langevin algorithm (MALA) [Roberts et al. 1996]
4. Preconditioned-MALA (pMALA)
   [Girolami et al. 2011]

MCMC samples taken from Riabiz-Chen-Cockayne-Swietach-Niederer-Mackey-Oates, 2021

MCMC experiments: Differential equation models

Dimension $d = 4$

1. Lotka-Volterra model
   oscillatory enzymatic control, [1925, 1926]
2. Goodwin model
   oscillatory predator-prey evolution, [1965]

x

1. Posterior

x

1. Random walk (RW)
   [Metropolis et al. 1953, Hastings 1970]
2. Adaptive random walk (adaRW)
   [Haario et al. 1999]
3. Metropolis adjusted Langevin algorithm (MALA) [Roberts et al. 1996]
4. Preconditioned-MALA (pMALA)
   [Girolami et al. 2011]

Dimension $d = 38$

3. Hinch calcium signal model
   [Hinch-Greenstein-Tanskanen-Xu-Winslow, 2004]

x

1. Posterior
2. Tempered posterior

x

1. Random walk (RW) - run 1
2. Random walk (RW) - run 2

MCMC samples taken from Riabiz-Chen-Cockayne-Swietach-Niederer-Mackey-Oates, 2021

MCMC experiments: Differential equation models

Dimension $d = 4$

1. Lotka-Volterra model
   oscillatory enzymatic control, [1925, 1926]
2. Goodwin model
   oscillatory predator-prey evolution, [1965]

x

1. Posterior

x

1. Random walk (RW)
   [Metropolis et al. 1953, Hastings 1970]
2. Adaptive random walk (adaRW)
   [Haario et al. 1999]
3. Metropolis adjusted Langevin algorithm (MALA) [Roberts et al. 1996]
4. Preconditioned-MALA (pMALA)
   [Girolami et al. 2011]

Dimension $d = 38$

3. Hinch calcium signal model
   [Hinch-Greenstein-Tanskanen-Xu-Winslow, 2004]

x

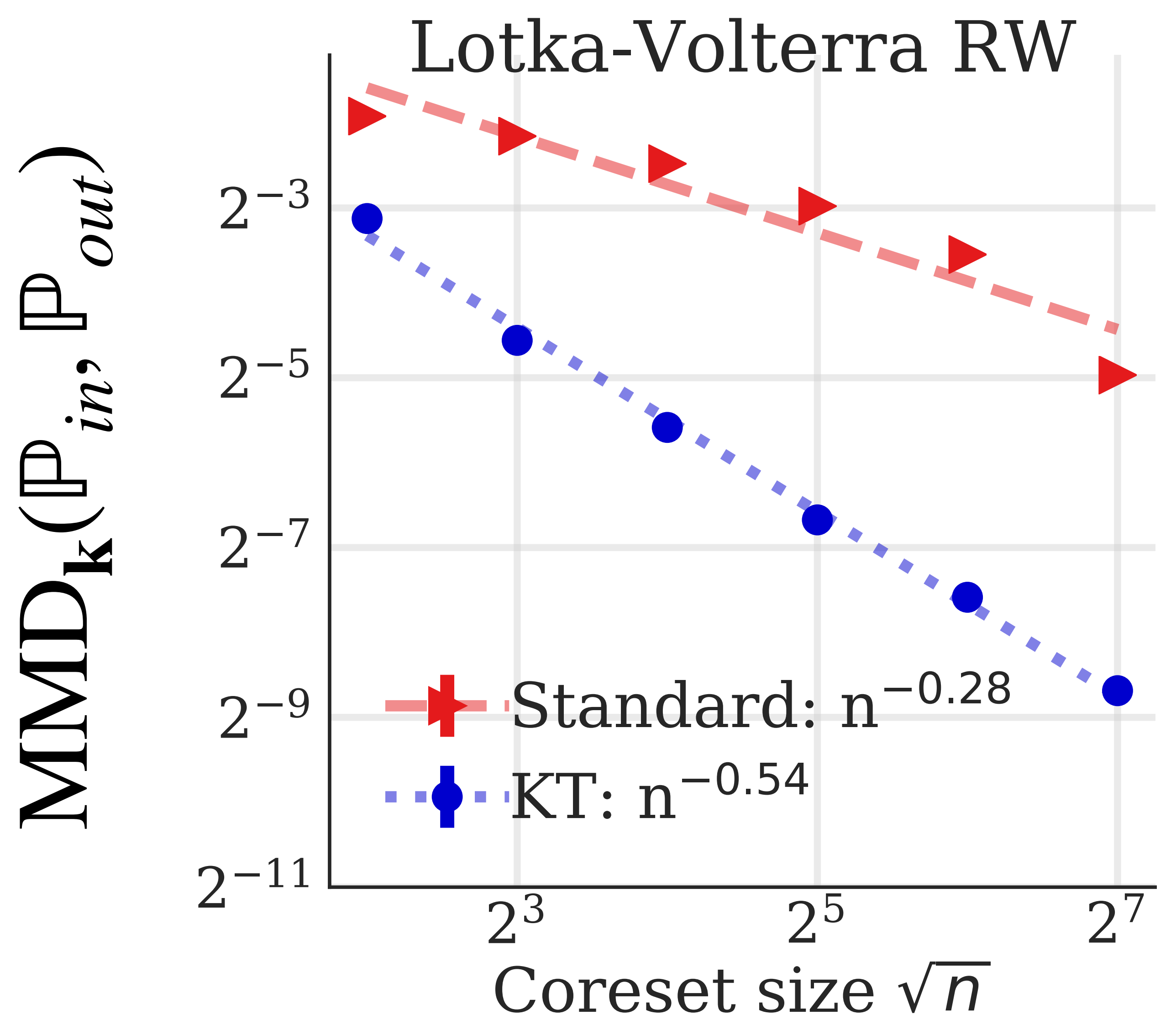
1. Posterior
2. Tempered posterior

x

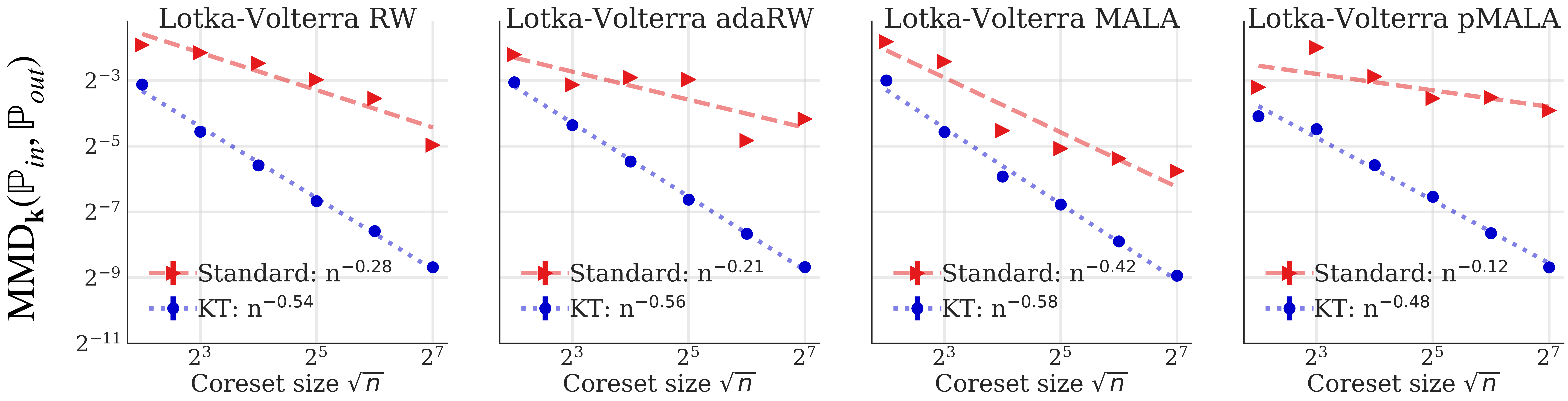
1. Random walk (RW) - run 1
2. Random walk (RW) - run 2

For KT, we use Gaussian kernel, and chose its bandwidth via median heuristic [Garreau et al. 2017]

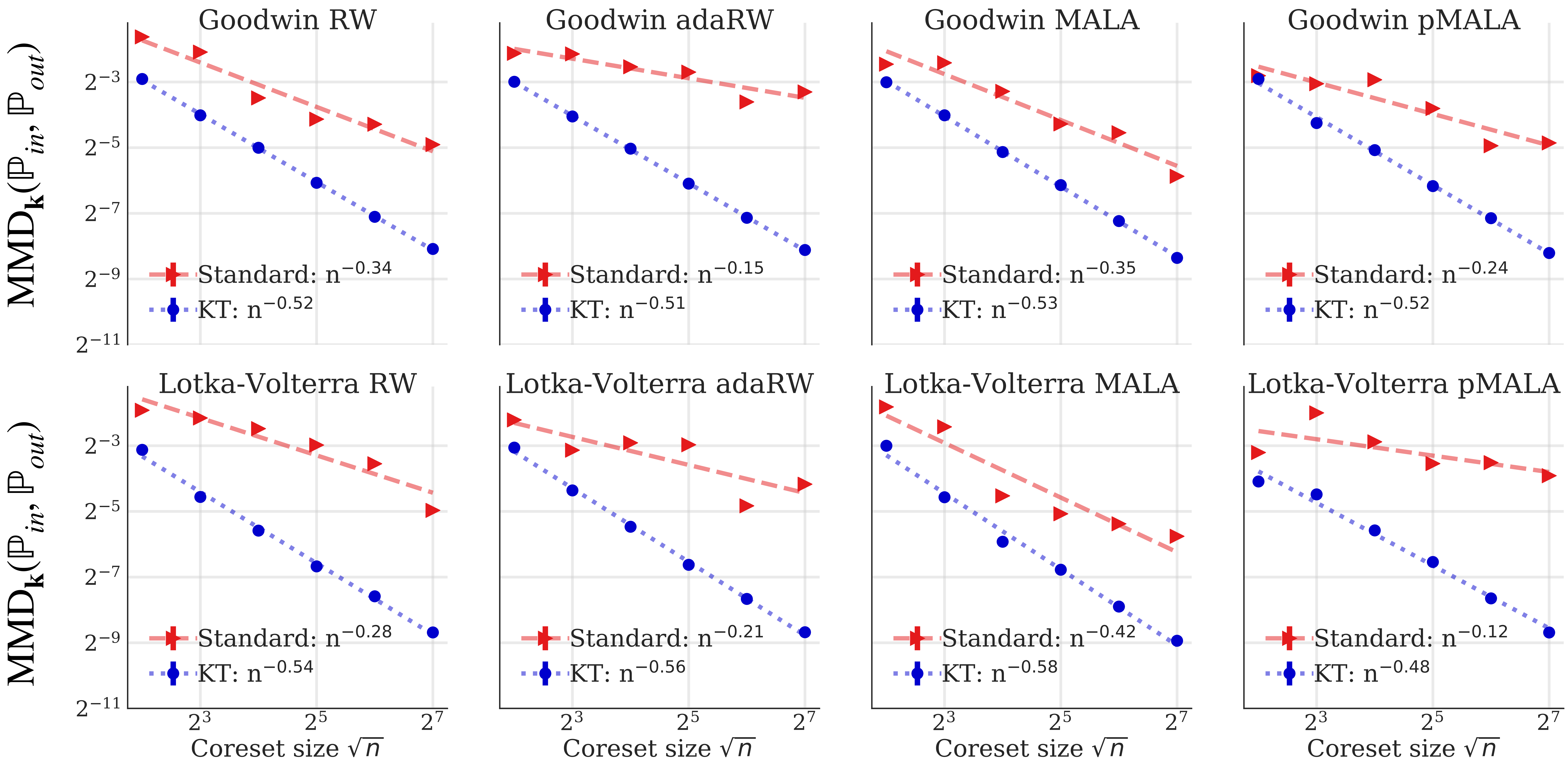
KT for Lotka-Volterra



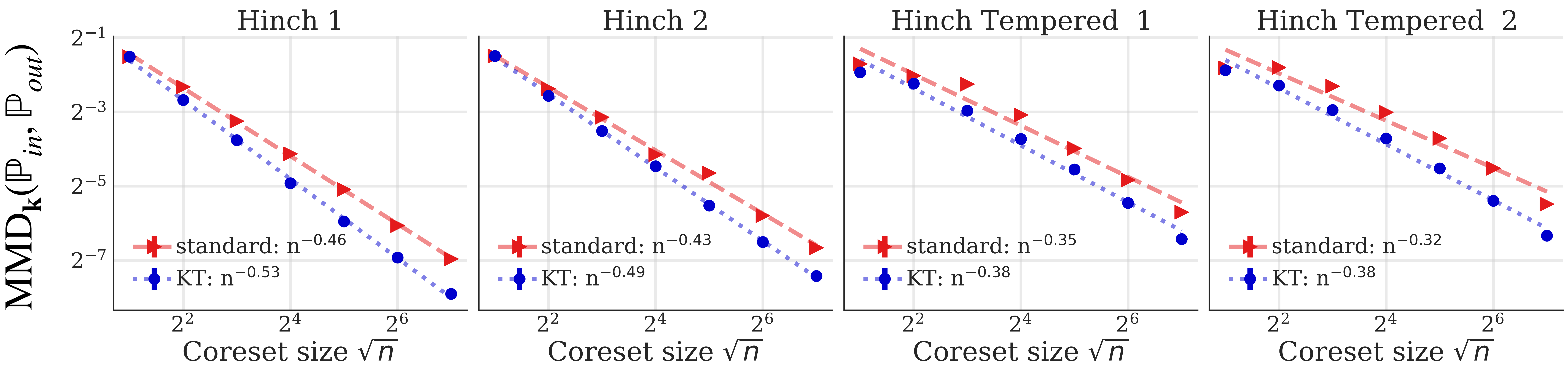
KT for Lotka-Volterra



KT for Lotka-Volterra & Goodwin: $n^{-1/2}$ error



KT for Hinch data



In this setting with $d = 38$, standard thinning is already good, but KT provides further improvement!

Overview of Kernel Thinning Algorithm

python™ pip install kernelthinning

```python
import kernelthinning as kt
```

```python
coreset = kt.thin(Sin, m, krt, k)
# Returns kernel thinning coreset
# of size floor(len(Sin)/2^m) as row indices into Sin
```

Input to algorithm

- Input: Kernels $\mathbf{k}$ and $\mathbf{k}_{rt}$, input points $\mathcal{S}_{in}$ of size n, thinning factor m

Input to algorithm

- Input: Kernels $\mathbf{k}$ and $\mathbf{k}_{rt}$, input points $\mathcal{S}_{in}$ of size n, thinning factor m
- What is $\mathbf{k}_{rt}$? It is the square-root kernel for $\mathbf{k}$

$$\mathbf{k}(x, y) = \int \mathbf{k}_{rt}(x, z)\mathbf{k}_{rt}(z, y)dz$$

Input to algorithm

- Input: Kernels $\mathbf{k}$ and $\mathbf{k}_{rt}$, input points $\mathcal{S}_{in}$ of size n, thinning factor m
- What is $\mathbf{k}_{rt}$? It is the square-root kernel for $\mathbf{k}$

$$\mathbf{k}(x, y) = \int \mathbf{k}_{rt}(x, z)\mathbf{k}_{rt}(z, y)dz$$

Name of kernel $\mathbf{k}(x, y) = \kappa(x-y)$	**Expression for** $\kappa(z)$	**Fourier transform** $\widehat{\kappa}(\omega)$	**Square-root kernel** $\mathbf{k}_{\mathrm{rt}}$
Gaussian(σ) : $\sigma > 0$	$\exp\left(-\frac{\|z\|_2^2}{2\sigma^2}\right)$	$\sigma^d \exp\left(-\frac{\sigma^2\|\omega\|_2^2}{2}\right)$	$\left(\frac{2}{\pi\sigma^2}\right)^{\frac{d}{4}}$**Gaussian**$\left(\frac{\sigma}{\sqrt{2}}\right)$
Matérn(ν, γ) : $\nu > d, \gamma > 0$	$c_{\nu-\frac{d}{2}}(\gamma\|z\|_2)^{\nu-\frac{d}{2}}K_{\nu-\frac{d}{2}}(\gamma\|z\|_2)$	$\phi_{d,\nu,\gamma}\,(\gamma^2 + \|\omega\|_2^2)^{-\nu}$	$A_{\nu,\gamma,d}$**Matérn**$(\frac{\nu}{2}, \gamma)$
B-spline$(2\beta + 1)$: $\beta \in 2\mathbb{N} + 1$	$S_{2\beta+2,d}\prod_{j=1}^{d} \circledast^{2\beta+2}\mathbf{1}_{[-\frac{1}{2},\frac{1}{2}]}(z_j)$	$S'_{2\beta+2,d}\prod_{j=1}^{d} \frac{\sin^{2\beta+2}(\frac{\omega_j}{2})}{\omega_j^{2\beta+2}}$	$\widetilde{S}_{\beta,d}$**B-spline**(β)

Input to algorithm

- Input: Kernels $\mathbf{k}$ and $\mathbf{k}_{rt}$, input points $\mathcal{S}_{in}$ of size n, thinning factor m
- What is $\mathbf{k}_{rt}$? It is the square-root kernel for $\mathbf{k}$

$$\mathbf{k}(x, y) = \int \mathbf{k}_{rt}(x, z)\mathbf{k}_{rt}(z, y)dz$$

- Exact square-root not necessary, see the paper for convenient choices for inverse multiquadrics, sech, Wendland, and any smooth and integrable kernel

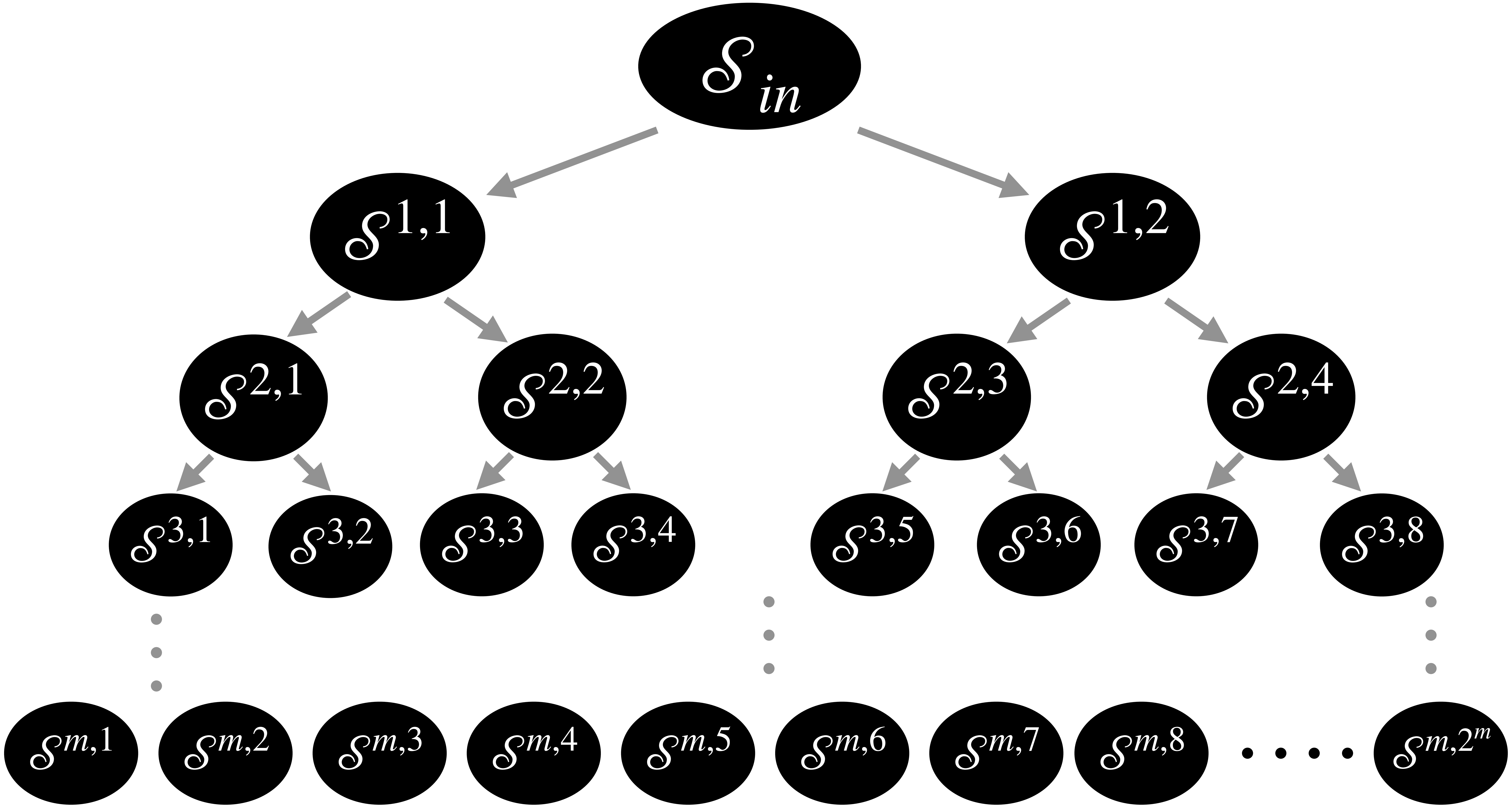
KT: A two-staged algorithm

- Input: Kernels $\mathbf{k}$ and $\mathbf{k}_{rt}$, input points $\mathcal{S}_{in}$ of size n, thinning factor m
- **KT-Split**:
  - Split $\mathcal{S}_{in}$ into 2^m balanced candidate coresets each of size $\frac{n}{2^m}$
  - When $m = \frac{1}{2}\log_2 n$, we have $\sqrt{n}$ coresets each of size $\sqrt{n}$
- **KT-Swap**:
  - Pick the best candidate coreset that minimized $\mathrm{MMD}_{\mathbf{k}}$ to input
  - Iteratively refine each point in the selected coreset by swapping with the best alternative $\mathcal{S}_{in}$ if it improves the MMD error

Computation: $\mathcal{O}(n^2)$ kernel evaluations
Storage: $n \min(n, d)$

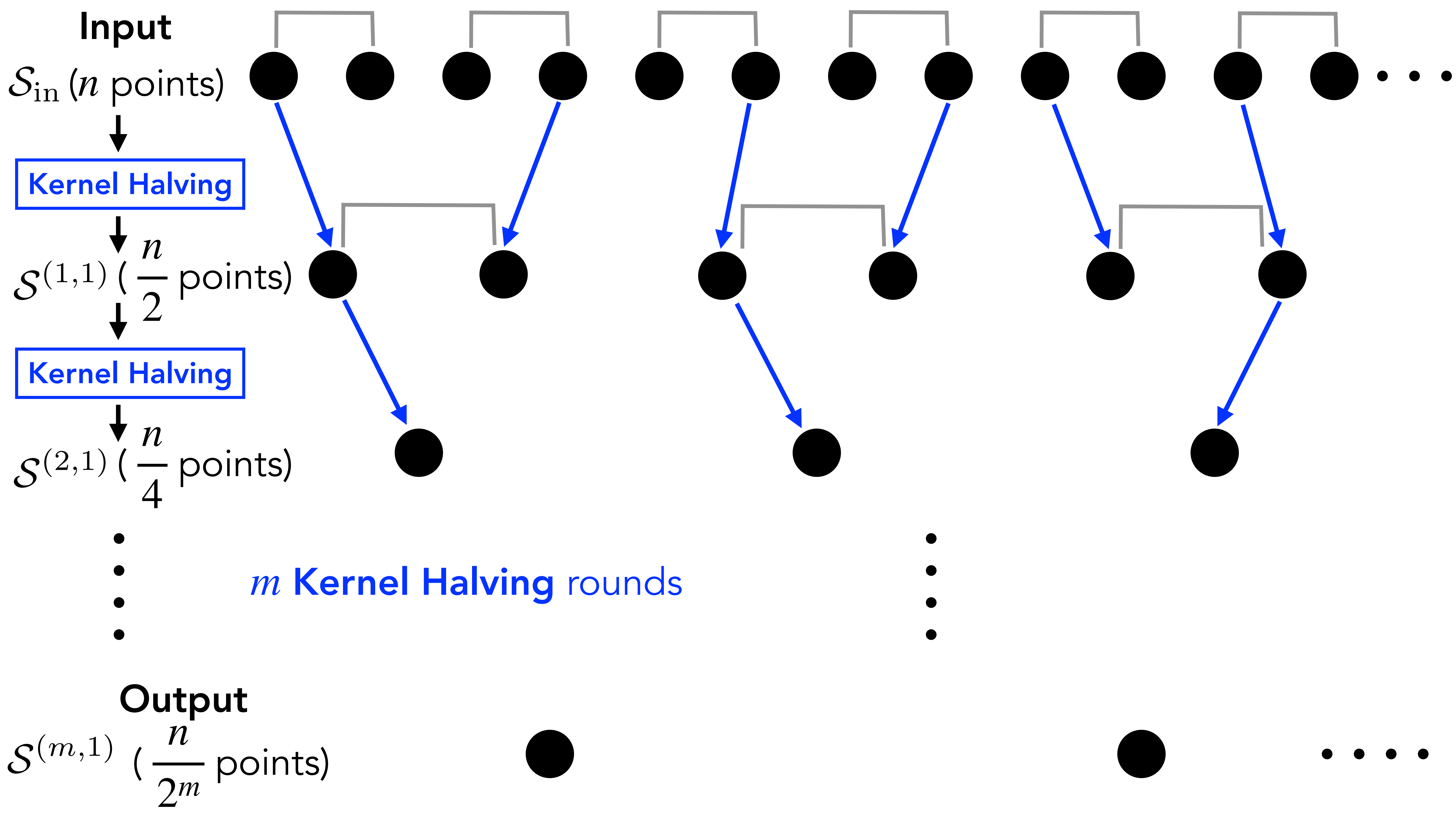
KT-Split

- Repeated rounds of splitting the parent coreset in two equal-sized children coresets
- Runs online, after seeing t input points, the bottom nodes have $t/2^m$ points



KT-Split

- One path on the tree is obtained by repeated **kernel halving**
- At each halving round, remaining points are paired, and one point is selected **non-uniformly** from each pair using a **new Hilbert space generalization of the self-balancing walk** of [Alweiss-Liu-Sawhney 2020]



Algorithm 2: Self-balancing Hilbert Walk

Input: sequence of functions $(f_i)_{i=1}^n$ in Hilbert space $\mathcal{H}$, threshold sequence $(\mathfrak{a}_i)_{i=1}^n$

$\psi_0 \leftarrow \mathbf{0} \in \mathcal{H}$

for $i = 1, 2, \ldots, n$ **do**

  $\alpha_i \leftarrow \langle \psi_{i-1}, f_i \rangle_{\mathcal{H}}$ // Compute Hilbert space inner product

  if $|\alpha_i| > \mathfrak{a}_i$**:**

    $\psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / \mathfrak{a}_i$

  else:

    $\eta_i \leftarrow 1$ with probability $\frac{1}{2}(1 - \alpha_i/\mathfrak{a}_i)$ and $\eta_i \leftarrow -1$ otherwise

    $\psi_i \leftarrow \psi_{i-1} + \eta_i f_i$

end

return ψ_n, combination of signed input functions

Algorithm 2: Self-balancing Hilbert Walk

Input: sequence of functions $(f_i)_{i=1}^n$ in Hilbert space $\mathcal{H}$, threshold sequence $(\mathfrak{a}_i)_{i=1}^n$

$\psi_0 \leftarrow \mathbf{0} \in \mathcal{H}$

for $i = 1, 2, \ldots, n$ **do**

  $\alpha_i \leftarrow \langle \psi_{i-1}, f_i \rangle_{\mathcal{H}}$ // Compute Hilbert space inner product

  if $|\alpha_i| > \mathfrak{a}_i$**:**

    $\psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / \mathfrak{a}_i$ ← We choose $\mathfrak{a}_i$ such that this step **does not occur** with high probability

  else:

    $\eta_i \leftarrow 1$ with probability $\frac{1}{2}(1 - \alpha_i/\mathfrak{a}_i)$ and $\eta_i \leftarrow -1$ otherwise

    $\psi_i \leftarrow \psi_{i-1} + \eta_i f_i$

end

return ψ_n, combination of signed input functions

Algorithm 2: Self-balancing Hilbert Walk

Input: sequence of functions $(f_i)_{i=1}^n$ in Hilbert space $\mathcal{H}$, threshold sequence $(\mathfrak{a}_i)_{i=1}^n$

$\psi_0 \leftarrow \mathbf{0} \in \mathcal{H}$

for $i = 1, 2, \ldots, n$ **do**

  $\alpha_i \leftarrow \langle \psi_{i-1}, f_i \rangle_{\mathcal{H}}$ // Compute Hilbert space inner product

  if $|\alpha_i| > \mathfrak{a}_i$**:**

    $\psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / \mathfrak{a}_i$ ← We choose $\mathfrak{a}_i$ such that this step **does not occur** with high probability

  else:

    $\eta_i \leftarrow 1$ with probability $\frac{1}{2}(1 - \alpha_i/\mathfrak{a}_i)$ and $\eta_i \leftarrow -1$ otherwise

    $\psi_i \leftarrow \psi_{i-1} + \eta_i f_i$

end

return ψ_n, combination of signed input functions

- Exact **Kernel halving**: When $f_i = \mathbf{k}_{rt}(2i, \cdot\,) - \mathbf{k}_{rt}(x_{2i-1}, \cdot\,)$, exactly half of input points ($\mathcal{S}_{out}$) given -1 sign after $n/2$ steps

$$\frac{1}{n}\psi_n = \frac{1}{n}\sum_{x\in\mathcal{S}_{\text{in}}} \mathbf{k}_{\text{rt}}(x, \cdot) - \frac{2}{n}\sum_{x\in\mathcal{S}_{\text{out}}} \mathbf{k}_{\text{rt}}(x, \cdot) = \mathbb{P}_{\text{in}}\mathbf{k}_{\text{rt}} - \mathbb{P}_{\text{out}}\mathbf{k}_{\text{rt}}$$

Algorithm 2: Self-balancing Hilbert Walk

Input: sequence of functions $(f_i)_{i=1}^n$ in Hilbert space $\mathcal{H}$, threshold sequence $(\mathfrak{a}_i)_{i=1}^n$

$\psi_0 \leftarrow \mathbf{0} \in \mathcal{H}$

for $i = 1, 2, \ldots, n$ **do**

  $\alpha_i \leftarrow \langle \psi_{i-1}, f_i \rangle_{\mathcal{H}}$ // Compute Hilbert space inner product

  if $|\alpha_i| > \mathfrak{a}_i$**:**

    $\psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / \mathfrak{a}_i$

  else:

    $\eta_i \leftarrow 1$ with probability $\frac{1}{2}(1 - \alpha_i/\mathfrak{a}_i)$ and $\eta_i \leftarrow -1$ otherwise

    $\psi_i \leftarrow \psi_{i-1} + \eta_i f_i$

end

return ψ_n, combination of signed input functions

- **Balance**: If $\mathbf{k}_{rt}$ is a reproducing kernel $\psi_n(z)/n$ is $\mathcal{O}(n^{-1} \cdot \sqrt{\log n})$-sub-Gaussian for all z

If η_i were chosen i.i.d., the sub-Gaussian parameter is $\Omega(n^{-1/2})$

Why the square-root kernel $\mathbf{k}_{rt}$?

- **A deterministic result:**

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \leq c_d R^{\frac{d}{2}} \|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty + 2\tau_{\mathbf{k}_{rt}}(R) + 2\|\mathbf{k}\|_\infty^{\frac{1}{2}} \max(\tau_{\mathbb{P}}(R), \tau_{\mathbb{Q}}(R))$$

Why the square-root kernel $\mathbf{k}_{rt}$?

- **A deterministic result:**

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \leq c_d R^{\frac{d}{2}} \|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty + 2\tau_{\mathbf{k}_{rt}}(R) + 2\|\mathbf{k}\|_\infty^{\frac{1}{2}} \max(\tau_{\mathbb{P}}(R), \tau_{\mathbb{Q}}(R))$$

L^∞**-error:** $\|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty \triangleq \sup_{z \in \mathbb{R}^d} |\mathbb{P}\mathbf{k}_{rt}(z) - \mathbb{Q}\mathbf{k}_{rt}(z)|$

Why the square-root kernel $\mathbf{k}_{rt}$?

- **A deterministic result:**

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P},\mathbb{Q}) \le c_d R^{\frac{d}{2}} \|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty + 2\tau_{\mathbf{k}_{rt}}(R) + 2\|\mathbf{k}\|_\infty^{\frac{1}{2}} \max(\tau_{\mathbb{P}}(R), \tau_{\mathbb{Q}}(R))$$

L^∞**-error:** $\|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty \triangleq \sup_{z\in\mathbb{R}^d} |\mathbb{P}\mathbf{k}_{rt}(z) - \mathbb{Q}\mathbf{k}_{rt}(z)|$

Tail behaviors: $\tau_{\mathbb{P}}(R) \triangleq \mathbb{P}(\|X\|_2 \ge R)$, $\quad \tau^2_{\mathbf{k}_{rt}}(R) \triangleq \sup_x \int_{\|y\|_2 \ge R} \mathbf{k}^2_{rt}(x, x-y)dy$

Why the square-root kernel $\mathbf{k}_{rt}$?

- **A deterministic result:**

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}) \le c_d R^{\frac{d}{2}} \|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty + 2\tau_{\mathbf{k}_{rt}}(R) + 2\|\mathbf{k}\|_\infty^{\frac{1}{2}} \max(\tau_{\mathbb{P}}(R), \tau_{\mathbb{Q}}(R))$$

L^∞**-error:** $\|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty \triangleq \sup_{z \in \mathbb{R}^d} |\, \mathbb{P}\mathbf{k}_{rt}(z) - \mathbb{Q}\mathbf{k}_{rt}(z) \,|$

Tail behaviors: $\tau_{\mathbb{P}}(R) \triangleq \mathbb{P}(\|X\|_2 \ge R)$, $\quad \tau_{\mathbf{k}_{rt}}^2(R) \triangleq \sup_x \int_{\|y\|_2 \ge R} \mathbf{k}_{rt}^2(x, x-y) dy$

- Apply with $\mathbb{P} = \mathbb{P}_{in}$ and $\mathbb{Q} = \mathbb{P}_{out}$ for repeated kernel halving (i.e., repeated self-balancing walk with $\mathbf{k}_{rt}$) $\Rightarrow \|\mathbb{P}\mathbf{k}_{rt} - \mathbb{Q}\mathbf{k}_{rt}\|_\infty = \widetilde{O}(n^{-1/2})$

Why the square-root kernel $\mathbf{k}_{rt}$?

- Thus, we have

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_{in}, \mathbb{P}_{out}) \lesssim R^{d/2} \cdot \widetilde{O}(n^{-1/2})$$

where R such that

$$\max(\tau_{\mathbb{P}_{in}}(R), \tau_{\mathbf{k}_{rt}}(R)) \lesssim n^{-1/2}$$

Sufficient choices:
$R = \mathcal{O}(1)$ for compact case
$R = \mathcal{O}(\log n)$ for sub-exp decay

Summary: KT provides better than iid compression

n points → **Standard-$\sqrt{n}$ Thinning** → $\sqrt{n}$ points with $n^{-1/4}$ error (Monte Carlo rate)

n points → **Kernel Thinning** → $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ error

- → Like Quasi Monte Carlo but applicable to more general $\mathbb{P}^{\star}$
- → Nearly minimax integration error
- → Nearly optimal L^{∞}-error

python: `pip install kernelthinning`

GitHub: `rzrsk/kernel_thinning`

arXiv.org: `https://arxiv.org/abs/2105.05842`

Additional slides

Coming this Fall!

- **Dimension-independent** single function guarantees with KT…
- **Near-linear time** variant of kernel thinning…

Thinning to a larger size

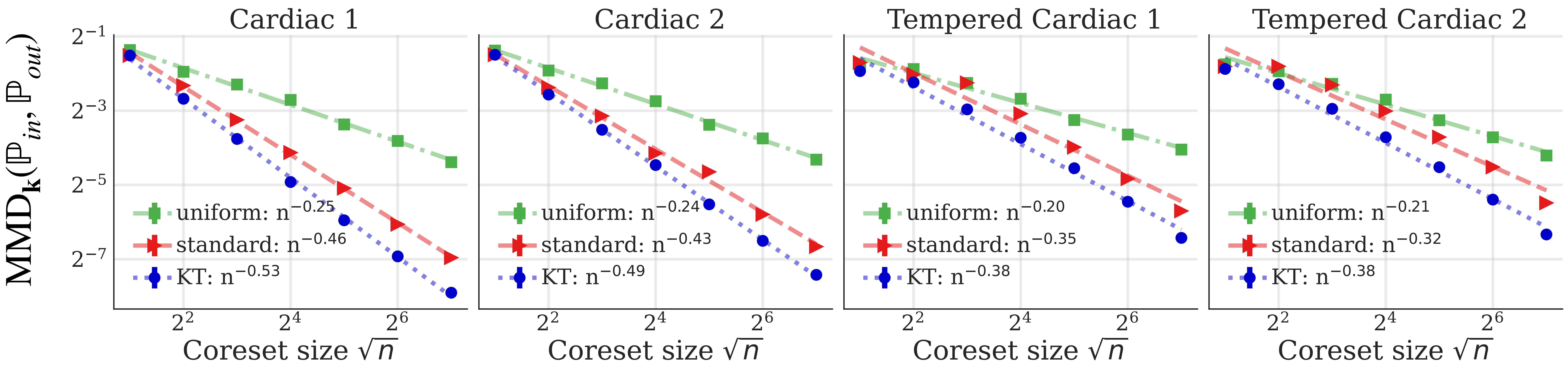
n points → **Standard-t Thinning** → $\frac{n}{t}$ points with $\sqrt{\frac{t}{n}}$ error

(Monte Carlo rate)

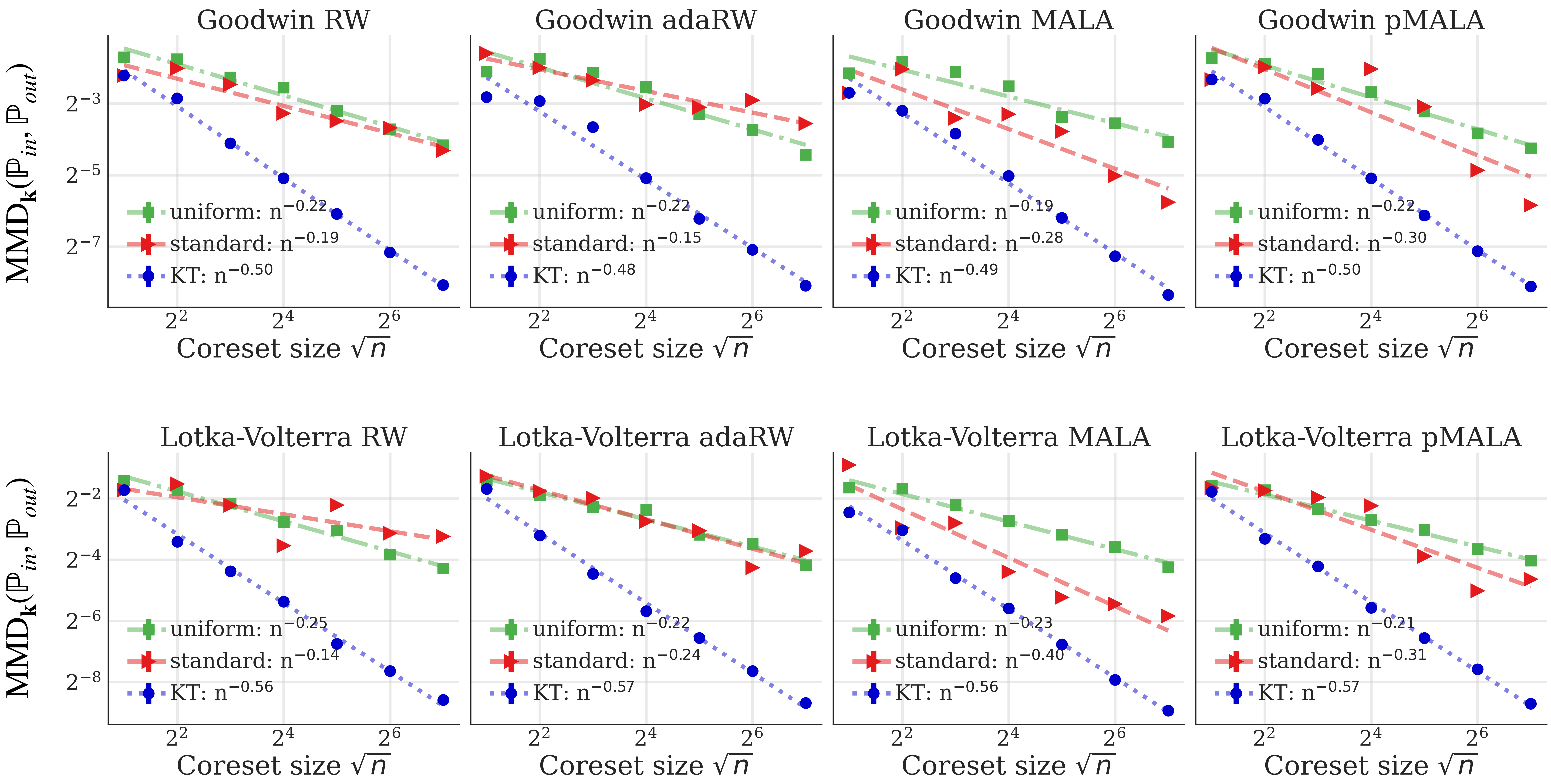
n points → **Kernel-t Thinning** → $\frac{n}{t}$ points with $\widetilde{O}\left(\frac{t}{n}\right)$ error

w.r.to input

Another look at results for cardiac data



Another look at MCMC results



Lower bounds

- For smooth kernels, there exists a target $\mathbb{P}$, such that a coreset of size $\sqrt{n}$ suffers an MMD error of $\min(\sqrt{\frac{d}{n}}, n^{-1/4})$. [Philips and Tai 2020]

- For characteristic kernels, there exists a target $\mathbb{P}$, such that any estimator based on n i.i.d. input points must suffer at least $n^{-1/2}$ MMD error. [Tolstikhin et al. 2017]

Both bounds apply to Gaussian and Matérn kernels