Advances in Distribution Compression

Lester Mackey

Microsoft Research New England

June 26, 2025

Joint work with Raaz Dwivedi, Marina Riabiz, Wilson Ye Chen, Jon Cockayne, Pawel Swietach, Steven A. Niederer, Chris J. Oates, Abhishek Shetty, Carles Domingo-Enrich, Lingxiao Li, Annabelle M. Carrell, and Albert Gong

Motivation: Computational Cardiology

Computational Cardiology: Developing multiscale *digital twins* of human hearts to non-invasively predict disease progression and therapy response [Niederer, Sacks, Girolami, and Willcox, 2021]

Single cell scale | Tissue scale | Organ scale

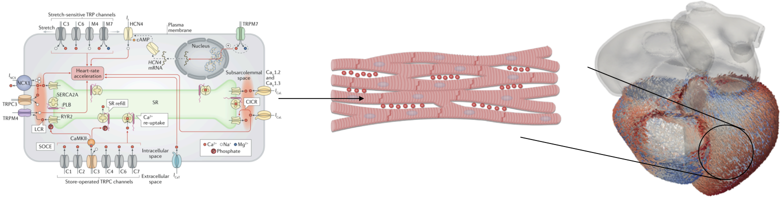


Figure credit: Marina Riabiz

Example (Heartbeats and arrhythmias)

- Whole-organ heartbeats are coordinated by calcium signaling in heart cells
- Dysregulation known to lead to life-threatening heart arrhythmias
- **Goal:** Model impact of calcium signaling dysregulation on heart function [Campos, Shiferaw, Prassl, Boyle, Vigmond, and Plank, 2015, Niederer, Lumens, and Trayanova, 2019, Colman, 2019]

Motivation: Computational Cardiology

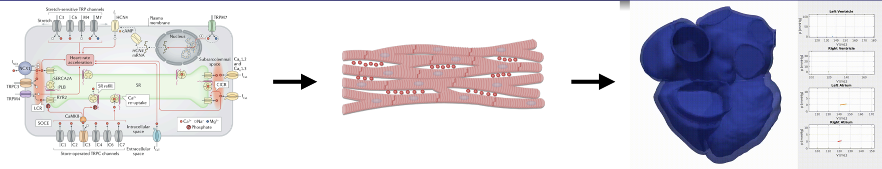


Figure credit: Augustin et al. 2020

Inferential Pipeline (Impact of calcium signaling dysregulation on heart function)

1. Estimate unknown calcium signaling model parameters from patient data
2. Capture uncertainty by sampling many likely parameter configurations
   - Run **Markov chain Monte Carlo (MCMC)** to (eventually) draw sample points from the posterior distribution $\mathbb{P}$ over unknown parameters
   - May require millions of sample points to adequately explore target distribution $\mathbb{P}$
3. Propagate uncertainty by simulating whole-heart model for each configuration
   - **Problem:** Each simulation requires thousands of CPU hours!

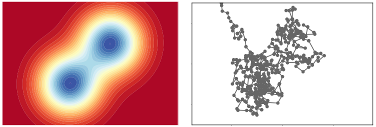
Questions: Can we accurately summarize $\mathbb{P}$ using many fewer points? If so, how?

Distribution Compression

Goal: Accurately summarize a distribution $\mathbb{P}$ using a small number of points

Standard solutions

- **i.i.d. sampling** directly from $\mathbb{P}$
- **MCMC** with Markov chain converging to $\mathbb{P}$



Benefits: Readily available and **eventually high-quality**

- Provide asymptotically exact sample estimates $\mathbb{P}_n f = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$ for intractable expectations $\mathbb{P}f = \mathbb{E}_{X\sim\mathbb{P}}[f(X)]$

Drawback: Samples are too large!

- Typical integration error $\mathbb{P}_n f - \mathbb{P}f = \Theta(n^{-1/2})$: need $n = 10000$ for 1% error
- Prohibitive for expensive downstream tasks and function evaluations

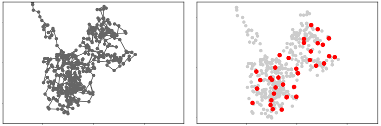
Idea: Directly compress the high-quality sample approximations $\mathbb{P}_n$

- Reduces general problem to approximating empirical distributions

Distribution Compression

Question: How do we effectively compress an empirical distribution $\mathbb{P}_n$?

Standard solutions



- **Uniform subsampling / i.i.d. sampling**
- **Standard thinning:** Keep every t-th point

Drawback: Large loss in accuracy, worst case integration error $= \Theta(\sqrt{t/n})$

- Compression from n to $\sqrt{n}$ points increases error from $\Theta(n^{-1/2})$ to $\Theta(n^{-1/4})$

Question: Can we do better?

Minimax lower bounds for worst-case integration error to $\mathbb{P}$

- $\Omega(n^{-1/2})$ for any compression procedure returning $\sqrt{n}$ points [Phillips and Tai, 2020]
- $\Omega(n^{-1/2})$ for any function of n i.i.d. points from $\mathbb{P}$ [Tolstikhin, Sriperumbudur, and Muandet, 2017]
- $\Theta(n^{-1/2} \log^{\frac{d-1}{2}} n)$ for best $\sqrt{n}$ points if $\mathbb{P} = \mathrm{Unif}([0,1]^d)$ [Novak and Wozniakowski, 2010]

This talk: Introduce a practical compression strategy – kernel thinning – that matches these lower bounds up to log factors, even for nonuniform and unbounded $\mathbb{P}$

Problem Setup

Given:

- Input points $\mathcal{S}_{\text{in}} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ with empirical distribution $\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$
  - Pre-generated by any algorithm (i.i.d. sampling, MCMC, quadrature, kernel herding)
- Target output size s (e.g., $s = \sqrt{n}$ for heavy compression)

Goal: Return coreset $\mathcal{S}_{\text{out}} \subset \mathcal{S}_{\text{in}}$ with $|\mathcal{S}_{\text{out}}| = s$, $\mathbb{Q} = \frac{1}{s}\sum_{x\in\mathcal{S}_{\text{out}}} \delta_x$, and $o(s^{-1/2})$ (better-than-i.i.d.) worst-case integration error between $\mathbb{P}_n$ and $\mathbb{Q}$

Maximum Mean Discrepancies

Goal: Return coreset $\mathcal{S}_{\text{out}} \subset \mathcal{S}_{\text{in}}$ with $|\mathcal{S}_{\text{out}}| = s$, $\mathbb{Q} = \frac{1}{s}\sum_{x \in \mathcal{S}_{\text{out}}} \delta_x$, and $o(s^{-1/2})$ worst-case integration error between $\mathbb{P}_n$ and $\mathbb{Q}$

Quality measure: Maximum mean discrepancy (MMD) [Gretton, Borgwardt, Rasch, Schölkopf, and Smola, 2012]

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_n, \mathbb{Q}) = \sup_{\|f\|_{\mathbf{k}} \le 1} |\mathbb{P}_n f - \mathbb{Q} f|$$

- Measures maximum discrepancy between input and coreset expectations over a class of real-valued test functions (unit ball of a reproducing kernel Hilbert space)
- Parameterized by a reproducing kernel $\mathbf{k}$: any symmetric $(\mathbf{k}(x, y) = \mathbf{k}(y, x))$ and positive semidefinite $(\sum_{i,l} c_i c_l \mathbf{k}(z_i, z_l) \ge 0, \forall z_i \in \mathbb{R}^d, c_i \in \mathbb{R})$ function
  - Gaussian: $\mathbf{k}(x, y) = e^{-\frac{1}{2}\|x-y\|_2^2}$, Inverse multiquadric: $\mathbf{k}(x, y) = \frac{1}{(1+\|x-y\|_2^2)^{1/2}}$
- Metrizes convergence in distribution for popular infinite-dimensional kernels (e.g., Gaussian, Matérn, B-spline, inverse multiquadric, sech, and Wendland)

Square-root Kernels

Definition (Square-root kernel)

A reproducing kernel $\mathbf{k}_{\mathrm{rt}}$ is a *square-root kernel* for $\mathbf{k}$ if

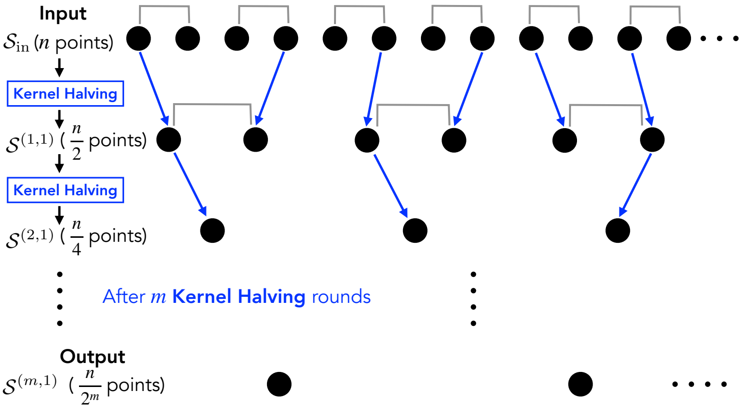
$$\mathbf{k}(x,y) = \int_{\mathbb{R}^d} \mathbf{k}_{\mathrm{rt}}(x,z)\mathbf{k}_{\mathrm{rt}}(y,z)dz.$$

$\mathbf{k}(x,y)=\kappa(x-y)$	$\kappa(z)$	**Square-root $\mathbf{k}_{\mathrm{rt}}$**
Gaussian(σ)	$\exp\left(-\frac{\|z\|_2^2}{2\sigma^2}\right)$	**Gaussian**$\left(\frac{\sigma}{\sqrt{2}}\right)$
Matérn(ν,γ)	$(\gamma\|z\|_2)^{\nu-\frac{d}{2}} K_{\nu-\frac{d}{2}}(\gamma\|z\|_2)$	**Matérn**$(\frac{\nu}{2},\gamma)$
B-spline$(2\beta+1)$	$\prod_{j=1}^d \circledast^{2\beta+2}\mathbf{1}_{[-\frac{1}{2},\frac{1}{2}]}(z_j)$	**B-spline**(β)

Theorem (L^∞ coresets for $\mathbf{k}_{\mathrm{rt}}$ are MMD coresets for $\mathbf{k}$ [Dwivedi and Mackey, 2024])

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_n,\mathbb{Q}) = \begin{cases} \mathcal{O}(\|\mathbb{P}_n\mathbf{k}_{\mathrm{rt}}-\mathbb{Q}\mathbf{k}_{\mathrm{rt}}\|_\infty) & \text{Compact support } \mathbf{k}_{\mathrm{rt}},\mathbb{P}_n \\ \mathcal{O}(\|\mathbb{P}_n\mathbf{k}_{\mathrm{rt}}-\mathbb{Q}\mathbf{k}_{\mathrm{rt}}\|_\infty \log(\frac{1}{\|\mathbb{P}_n\mathbf{k}_{\mathrm{rt}}-\mathbb{Q}\mathbf{k}_{\mathrm{rt}}\|_\infty})^{\frac{d+1}{2}}) & \text{Subexponential } \mathbf{k}_{\mathrm{rt}},\mathbb{P}_n \end{cases}$$

1. **Initialization:** KT-SPLIT
   - Partitions input $\mathcal{S}_{\text{in}}$ into balanced candidate coresets, each of size s



Kernel Halving [Dwivedi and Mackey, 2024]

Goal: Split $\mathcal{S}_{\text{in}}$ into two balanced coresets $\mathcal{S}_{\text{out}}, \mathcal{S}'_{\text{out}}$ of equal size

- Balance: $\mathbb{Q}\mathbf{k}_{\text{rt}} \approx \mathbb{Q}'\mathbf{k}_{\text{rt}} \Leftrightarrow \mathbb{P}_n\mathbf{k}_{\text{rt}} \approx \mathbb{Q}\mathbf{k}_{\text{rt}}$ for $\mathbb{Q}'\mathbf{k}_{\text{rt}} \triangleq \frac{1}{|\mathcal{S}'_{\text{out}}|}\sum_{x\in\mathcal{S}'_{\text{out}}} \mathbf{k}_{\text{rt}}(x,\cdot)$

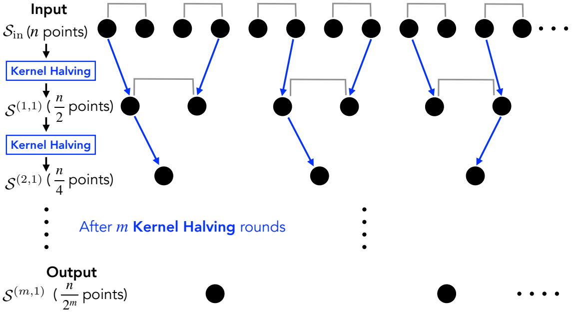
Uniformly random halving: $\|\mathbb{P}_n\mathbf{k}_{\text{rt}} - \mathbb{Q}\mathbf{k}_{\text{rt}}\|_\infty = \Omega(\frac{1}{\sqrt{n}})$ with high probability

Kernel halving: Check for balance before assigning points to coresets

- Hilbert space generalization of self-balancing walk of Alweiss, Liu, and Sawhney [2020]
- Start with empty coresets $\mathcal{S}_{\text{out}}, \mathcal{S}'_{\text{out}}$
- Assign input points $(x, x') = (x_1, x_2), \ldots, (x_{n-1}, x_n)$ to coresets two at a time:
  1. Try adding x to $\mathcal{S}_{\text{out}}$ and x' to $\mathcal{S}'_{\text{out}}$ and record $\alpha_{\text{heads}} = \|\mathbb{Q}\mathbf{k}_{\text{rt}} - \mathbb{Q}'\mathbf{k}_{\text{rt}}\|_{\mathbf{k}_{\text{rt}}}$
  2. Try adding x' to $\mathcal{S}_{\text{out}}$ and x to $\mathcal{S}'_{\text{out}}$ and record $\alpha_{\text{tails}} = \|\mathbb{Q}\mathbf{k}_{\text{rt}} - \mathbb{Q}'\mathbf{k}_{\text{rt}}\|_{\mathbf{k}_{\text{rt}}}$
  3. Final assignment: flip coin biased toward the more balanced option (the smaller α)
- **Theorem:** $\|\mathbb{P}_n\mathbf{k}_{\text{rt}} - \mathbb{Q}\mathbf{k}_{\text{rt}}\|_\infty = \mathcal{O}(\frac{\sqrt{d}\log(n)}{n})$ with high probability [Dwivedi and Mackey, 2024]

Kernel Thinning [Dwivedi and Mackey, 2024]

1. **Initialization:** KT-SPLIT
   - Partitions input $\mathcal{S}_{\text{in}}$ into balanced candidate coresets, each of size s



   - Non-uniform randomness ensures $\|\mathbb{P}_n\mathbf{k}_{\text{rt}} - \mathbb{Q}\mathbf{k}_{\text{rt}}\|_\infty$ small after each halving round
   - **Theorem:** $\mathrm{MMD}_{\mathbf{k}} = \begin{cases} \mathcal{O}(\sqrt{\frac{\log n}{n}}) & \text{Compact support } \mathbf{k}_{\text{rt}}, \mathbb{P}_n \\ \mathcal{O}(\frac{(\log n)^{\frac{d+1}{2}}\sqrt{\log\log n}}{\sqrt{n}}) & \text{Subexponential } \mathbf{k}_{\text{rt}}, \mathbb{P}_n \end{cases}$ with high prob.

     when $s = \sqrt{n}$ vs. $\Omega(n^{-\frac{1}{4}})$ for i.i.d. coreset [Dwivedi and Mackey, 2024]

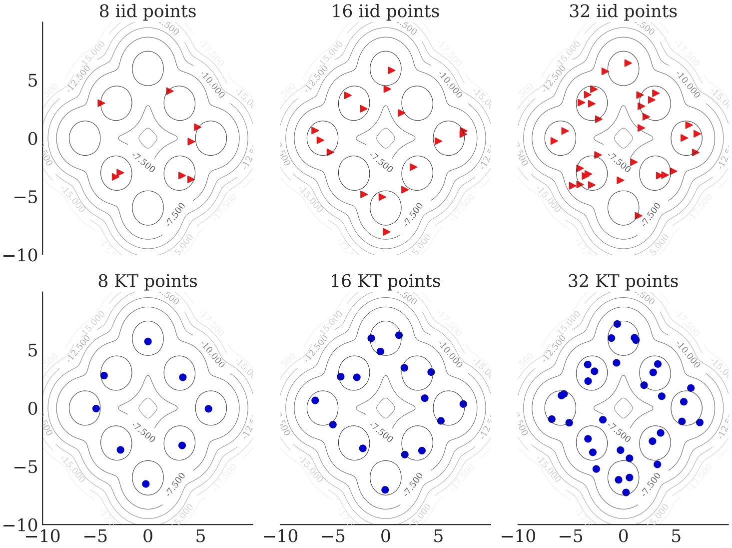
Kernel Thinning [Dwivedi and Mackey, 2024]

1. **Initialization:** KT-SPLIT
   - Partitions input $\mathcal{S}_{\text{in}}$ into balanced candidate coresets, each of size s
   - Non-uniform randomness ensures $\|\mathbb{P}_n\mathbf{k}_{\text{rt}} - \mathbb{Q}\mathbf{k}_{\text{rt}}\|_\infty$ small after each halving round
   - **Thm:** $\mathrm{MMD}_{\mathbf{k}} = \widetilde{\mathcal{O}}_d(s^{-1})$ for subexponential $\mathbf{k}_{\text{rt}}, \mathbb{P}_n$ vs. $\Omega(s^{-\frac{1}{2}})$ for i.i.d. [Dwivedi and Mackey, 2024]
2. **Refinement:** KT-SWAP
   - Selects candidate coreset closest to $\mathcal{S}_{\text{in}}$ in terms of $\mathrm{MMD}_{\mathbf{k}}$
   - Iteratively refines the coreset by replacing each coreset point in turn with the best alternative in $\mathcal{S}_{\text{in}}$, as measured by $\mathrm{MMD}_{\mathbf{k}}$

Complexity

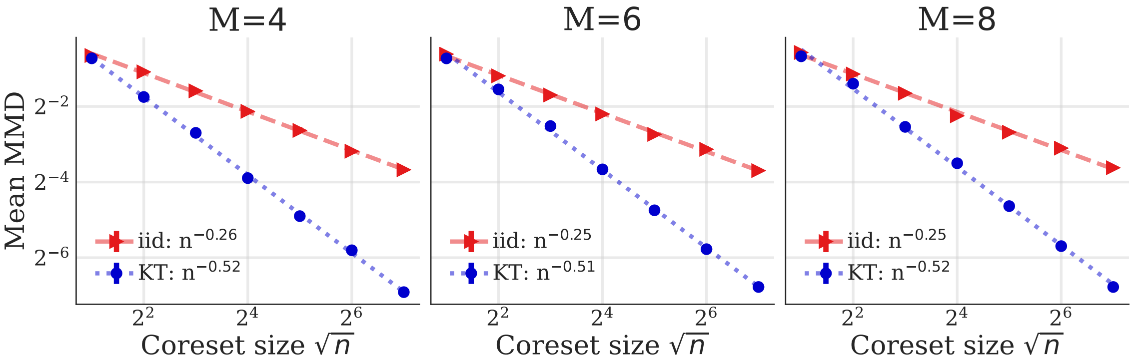
- Time: dominated by $\mathcal{O}(n^2)$ kernel evaluations
  - Reduces to $\mathcal{O}(n \log^3 n)$ for $s = \sqrt{n}$ using Compress++ of Shetty, Dwivedi, and Mackey [2022]
- Space: $\mathcal{O}(\min(nd, n^2))$
  - Reduces to $\mathcal{O}(\sqrt{n}d \log n)$ for $s = \sqrt{n}$ using Compress++

Kernel Thinning vs. i.i.d. Sampling: Mixture of Gaussians



- $\mathbb{P} = \frac{1}{M}\sum_{j=1}^{M}\mathcal{N}(\mu_j, \mathbf{I}_d)$
- $\mathbf{k}(x, y) = \exp(-\frac{1}{2\sigma^2}\|x - y\|_2^2)$ with $\sigma^2 = 2d$
- Even for small sample sizes, kernel thinning (KT) provides
  - Better stratification across components
  - Less clumping and fewer gaps within components

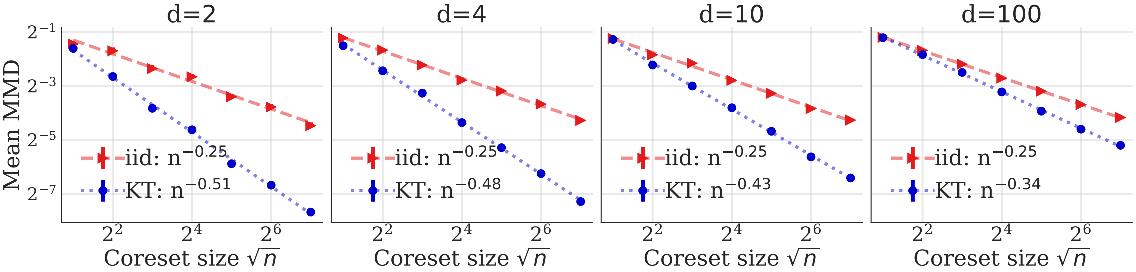
Kernel Thinning vs. i.i.d. Sampling: Mixture of Gaussians



Kernel thinning (KT) improves both rate of decay and order of magnitude of $\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}_{KT})$

- $\mathbb{P} = \frac{1}{M}\sum_{j=1}^{M} \mathcal{N}(\mu_j, \mathbf{I}_d)$, $d = 2$
- $\mathbf{k}(x, y) = \exp(-\frac{1}{2\sigma^2}\|x - y\|_2^2)$ with $\sigma^2 = 2d$

Kernel Thinning vs. i.i.d. Sampling: Higher Dimensions



Kernel thinning (KT) improves both rate of decay and order of magnitude of $\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}, \mathbb{Q}_{KT})$ even for high dimensions and small sample sizes

- $\mathbb{P} = \mathcal{N}(0, \mathbf{I}_d)$
- $\mathbf{k}(x, y) = \exp(-\frac{1}{2\sigma^2}\|x - y\|_2^2)$ with $\sigma^2 = 2d$

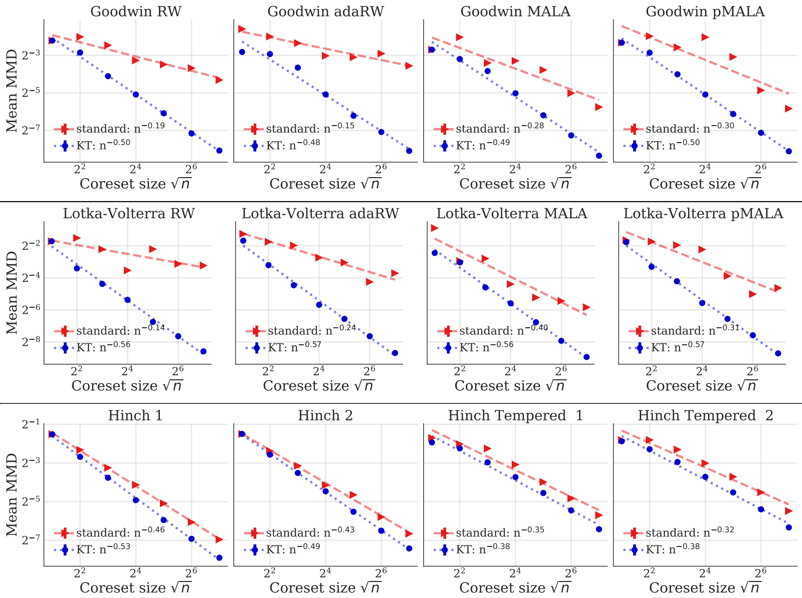
Kernel Thinning vs. Standard MCMC Thinning

Posterior inference for systems of ordinary differential equations (ODEs)

- $\mathbb{P} =$ posterior distribution of coupled ODE model parameters given observed data
- Goodwin model of oscillatory enzymatic control $(d = 4)$ [Goodwin, 1965]
- Lotka-Volterra model of oscillatory predator-prey evolution $(d = 4)$ [Lotka, 1925, Volterra, 1926]
- Hinch model of cardiac calcium signalling $(d = 38)$ [Hinch, Greenstein, Tanskanen, Xu, and Winslow, 2004]
  - Downstream goal: propagate model uncertainty through whole-heart simulation
  - Every sample point discarded via compression $=$ 1000 CPU hours saved

MCMC input points [Riabiz, Chen, Cockayne, Swietach, Niederer, Mackey, and Oates, 2021]

- Gaussian random walk (RW), adaptive RW (adaRW) [Haario, Saksman, and Tamminen, 1999]
  - 2 weeks of CPU time to generate each RW Hinch chain of length 4×10^6
- Metropolis-adjusted Langevin algorithm (MALA) [Roberts and Tweedie, 1996]
- Pre-conditioned MALA (pMALA) [Girolami and Calderhead, 2011]
- Discarded burn-in and standard thinned to form $\mathbb{P}_n$
- $\mathbf{k}(x, y) = \exp(-\frac{1}{2\sigma^2}\|x - y\|_2^2)$ with median heuristic σ^2 [Garreau, Jitkrittum, and Kanagawa, 2017]



KT improves rate of decay and magnitude of MMD, even when standard thinning is accurate

Something's Wrong

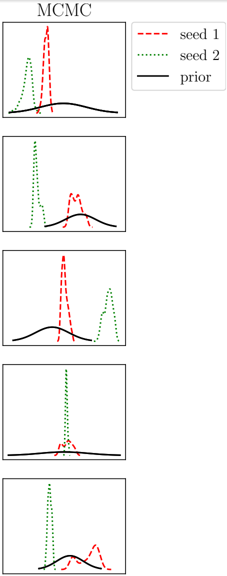
Problem: The Hinch Markov chains haven't mixed!

Solution: Use a more diffuse *tempered* posterior $\tilde{\mathbb{P}}$ for faster mixing

Problem: Tempering introduces a persistent bias

- MCMC points $\mathbb{P}_n$ will be summarizing the wrong distribution $\tilde{\mathbb{P}}$

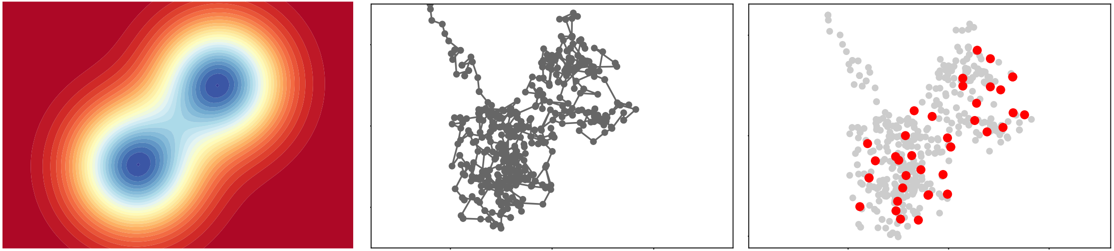
Question: Can we correct for such biases during compression?



Compression with Bias Correction

Question: Can we correct for distributional biases in $\mathbb{P}_n$ during compression?

- e.g., Biases due to off-target sampling, tempering, approximate MCMC, or burn-in



Difficulty: $\mathbb{P}_n$ alone is insufficient; need to measure distance to the true target $\mathbb{P}$

Measuring Distance to $\mathbb{P}$

Quality measure: Maximum mean discrepancy (MMD) [Gretton, Borgwardt, Rasch, Schölkopf, and Smola, 2012]

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P},\mathbb{Q}) = \sup_{\|f\|_{\mathbf{k}}\le 1} |\mathbb{P}f - \mathbb{Q}f| = \sqrt{(\mathbb{P}\times\mathbb{P})\mathbf{k} + (\mathbb{Q}\times\mathbb{Q})\mathbf{k} - 2(\mathbb{Q}\times\mathbb{P})\mathbf{k}}$$

Problem: Integration under $\mathbb{P}$ is typically intractable!

$\Rightarrow$ $\mathbb{P}\mathbf{k}$ and $\mathrm{MMD}_{\mathbf{k}}(\mathbb{P},\mathbb{Q})$ cannot be computed in practice for most kernels

Idea: Only consider kernels $\mathbf{k}_{\mathbb{P}}$ with $\mathbb{P}\mathbf{k}_{\mathbb{P}}$ known *a priori* to be 0

- Then MMD computation only depends on $\mathbb{Q}$!

Kernel Stein Discrepancies

Idea: Consider $\mathrm{MMD}_{\mathbf{k}_{\mathbb{P}}}$ with $\mathbb{P}\mathbf{k}_{\mathbb{P}}$ known *a priori* to be 0

Kernel Stein discrepancy (KSD)

[Chwialkowski, Strathmann, and Gretton, 2016, Liu, Lee, and Jordan, 2016, Gorham and Mackey, 2017]

- $\mathbf{k}_{\mathbb{P}}(x,y) = \sum_{j=1}^{d} \frac{1}{p(x)p(y)} \nabla_{x_j} \nabla_{y_j} (p(x)\mathbf{k}(x,y)p(y))$ [Oates, Girolami, and Chopin, 2017]
  - $\mathbb{P}$ has differentiable Lebesgue density p
  - $\mathbf{k}$ is a bounded base kernel with bounded continuous derivatives
- $\mathbb{P}\mathbf{k}_{\mathbb{P}} = 0$ whenever $\nabla \log p$ is integrable [Gorham and Mackey, 2017]
- Depends on $\mathbb{P}$ through $\nabla \log p$: computable when normalization constant unknown

$\Rightarrow$ Kernel Stein discrepancy $\mathrm{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P},\mathbb{Q})$ is computable!

Theorem (KSD controls convergence in distribution [Gorham and Mackey, 2017, Chen, Barp, Briol, Gorham, Girolami, Mackey, and Oates, 2019])

Consider the base kernel $\mathbf{k}(x,y) = (c^2 + \|\Gamma(x-y)\|_2^2)^{-1/2}$ for any $c > 0$ and positive definite Γ. If $\mathbb{P}$ has strongly log concave tails and Lipschitz $\nabla \log p$, then $\mathbb{Q}_s \Rightarrow \mathbb{P}$ whenever $\mathrm{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P},\mathbb{Q}_s) \to 0$.

Stein Kernel Thinning: Stein Thinning + Kernel Thinning

(1) Stein thinning: Greedily minimize KSD using points from $\mathcal{S}_{\text{in}} = \{x_1, \dots, x_n\}$

[Riabiz, Chen, Cockayne, Swietach, Niederer, Mackey, and Oates, 2021]

- Choose initial approximation $\mathbb{P}_1 = \delta_{y_1}$ with

$$y_1 \in \operatorname{argmin}_{y \in \mathcal{S}_{\text{in}}} \operatorname{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P}, \delta_y) = \operatorname{argmin}_{y \in \mathcal{S}_{\text{in}}} \mathbf{k}_{\mathbb{P}}(y, y)$$

- Iteratively construct $\mathbb{P}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i}$ with

$$\begin{aligned} y_n &\in \operatorname{argmin}_{y \in \mathcal{S}_{\text{in}}} \operatorname{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P}, \tfrac{n-1}{n}\mathbb{P}_{n-1} + \tfrac{1}{n}\delta_y) \\ &= \operatorname{argmin}_{y \in \mathcal{S}_{\text{in}}} \mathbf{k}_{\mathbb{P}}(y, y) + 2\sum_{i=1}^{n-1} \mathbf{k}_{\mathbb{P}}(y_i, y) \end{aligned}$$

- Same point x_i can be selected multiple times
- Runtime $= \mathcal{O}(n^2)$ after n steps

(2) Kernel thinning: Compress debiased approximation $\mathbb{P}_n$ to obtain coreset $\mathbb{Q}$

Stein Kernel Thinning Guarantees

Theorem (Stein KT guarantee [Li, Dwivedi, and Mackey])

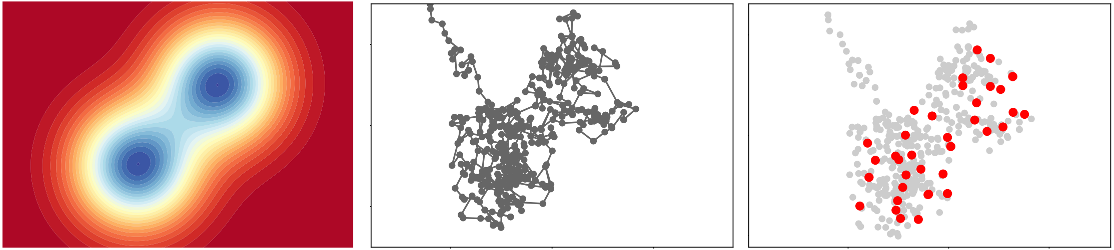
$$\mathrm{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P}, \mathbb{Q}) \leq \inf_{w \in \Delta_{n-1}} \mathrm{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P}, \textstyle\sum_{i=1}^{n} w_i \delta_{x_i}) + \widetilde{\mathcal{O}}_d(n^{-1/2})$$

with high probability for $s = \sqrt{n}$ and slow-growing inputs and $\mathbf{k}_{\mathbb{P}}$.

Takeaway: Stein KT performs nearly as well as best simplex reweighting of $\mathcal{S}_{\text{in}}$

$\Rightarrow$ Nearly as well as Markov chain with burn-in removed!

$\Rightarrow$ Nearly as well as off-target sample after optimal importance sampling reweighting!



Stein Kernel Thinning Guarantees

Takeaway: Stein KT performs nearly as well as best simplex reweighting of $\mathcal{S}_{\text{in}}$

$\Rightarrow$ Nearly as well as Markov chain with burn-in removed!

$\Rightarrow$ Neary as well as off-target sample after optimal importance sampling reweighting!

Theorem (Stein KT corrects off-target sampling [Li, Dwivedi, and Mackey])

If $\mathcal{S}_{\text{in}}$ drawn $i.i.d.$ from $\tilde{\mathbb{P}}$ with tails no lighter than $\mathbb{P}$ (i.e., $\mathbb{E}_{\tilde{\mathbb{P}}}[\frac{d\mathbb{P}}{d\tilde{\mathbb{P}}}(X)^3 \mathbf{k}_{\mathbb{P}}(X,X)^4] < \infty$),

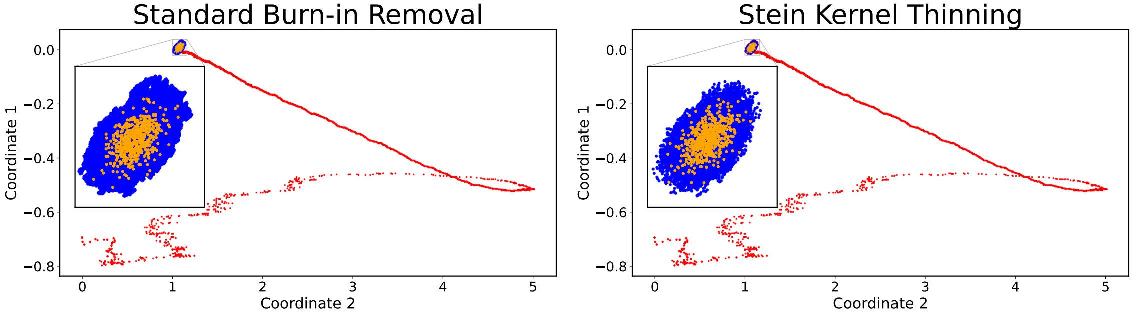
$$\mathrm{MMD}_{\mathbf{k}_{\mathbb{P}}}(\mathbb{P}, \mathbb{Q}) = \widetilde{O}_d(n^{-1/2}) \text{ in probability,}$$

for $s = \sqrt{n}$ and slow-growing $\mathbf{k}_{\mathbb{P}}$.

- Result extends to geometrically ergodic Markov chains targeting $\tilde{\mathbb{P}}$

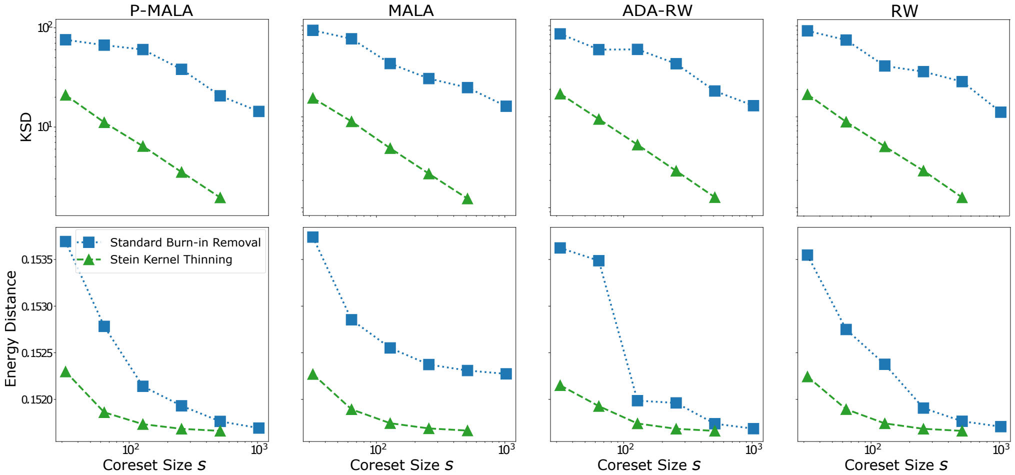
Stein KT in Action: Correcting for Burn-in

Goodwin model of oscillatory enzymatic control



- First two coordinates of `P-MALA` MCMC output
- Before selecting coresets, burn-in removal uses 6 Markov chains to discard burn-in
- Stein KT identifies the same high-density region with 1 chain

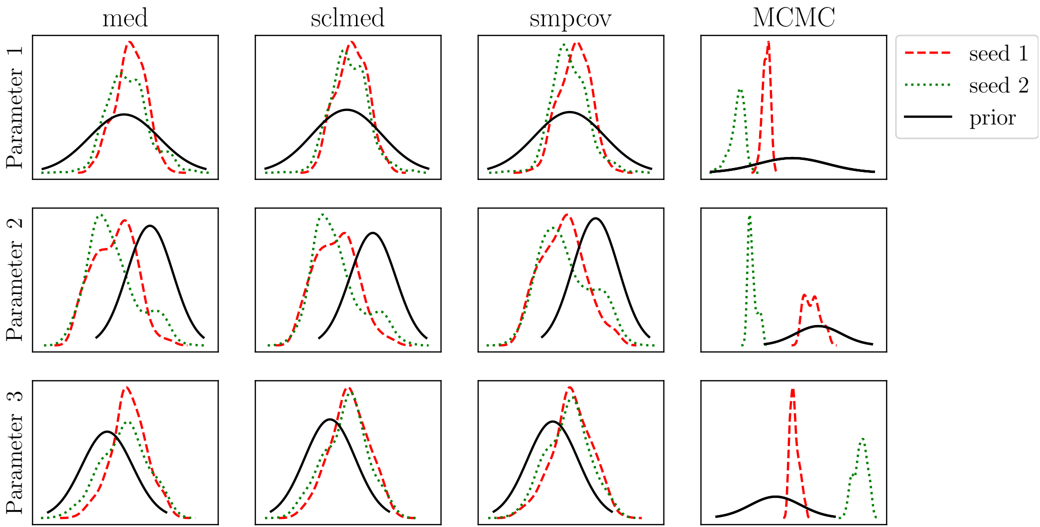
Goodwin model of oscillatory enzymatic control



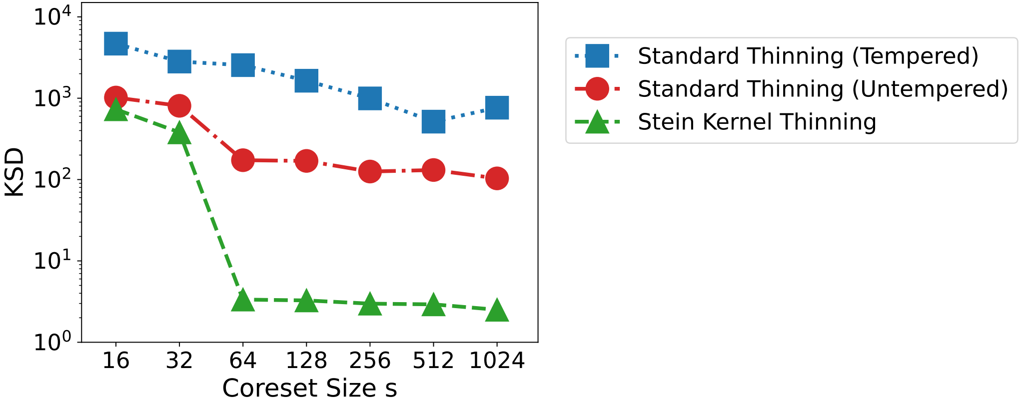
Stein KT outperforms standard burn-in removal in terms of KSD and energy distance

Stein KT in Action: Correcting for Tempering

Hinch model of cardiac calcium signalling: Tempering improves mixing



Hinch model of cardiac calcium signalling



- Untempered standard thinning yields poor summary due to poor mixing
- Tempered thinning without bias correction is even worse (due to tempering bias)
- Tempering + Stein KT bias correction improves approximation to $\mathbb{P}$

Conclusions

Summary

- New tools for summarizing a probability distribution more effectively than $\mathrm{i.i.d.}$ sampling or standard MCMC thinning
- Kernel thinning compresses an n point summary into a $\sqrt{n}$ point summary with better-than-$\mathrm{i.i.d.}$ approximation error
- Stein kernel thinning simultaneously compresses and reduces biases due to off-target sampling, tempering, or burn-in
- Compress++ speeds up thinning algorithms without ruining their quality

Kernel Thinning and **Compress++**

Papers:
- arxiv.org/abs/2105.05842
- arxiv.org/abs/2110.01593
- arxiv.org/abs/2111.07941

Stein Thinning and **Stein KT**

Papers:
- arxiv.org/abs/2005.03952
- arxiv.org/abs/2404.12290

Package: github.com/microsoft/goodpoints

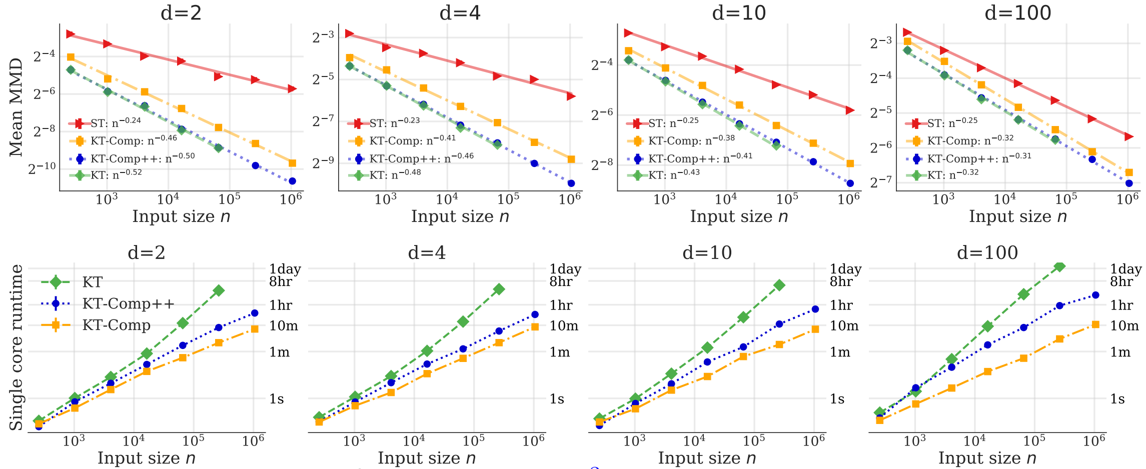
Generalized Kernel Thinning [Dwivedi and Mackey, 2022]

Question: Do you really need a square-root kernel?

1. KT-SPLIT with target kernel $\mathbf{k}$ yields
   - Similar or better MMD guarantees for smooth kernels (like Gaussian, IMQ, & sinc) and kernels with fast eigenvalue decay [Carrell, Gong, Shetty, Dwivedi, and Mackey, 2025]
   - Dimension-free $\mathcal{O}(\frac{\sqrt{\log s}}{s})$ single-function integration error for any $\mathbf{k}$ and $\mathbb{P}$
2. KT-SPLIT with fractional power kernel $\mathbf{k}_\alpha$ yields
   - Improved MMD for kernels without $\mathbf{k}_{\mathrm{rt}}$ (like Laplace and non-smooth Matérn)
3. KT-SPLIT with $\mathbf{k} + \mathbf{k}_\alpha$ yields all of the above simultaneously!
   - We call this **kernel thinning+ (KT+)**

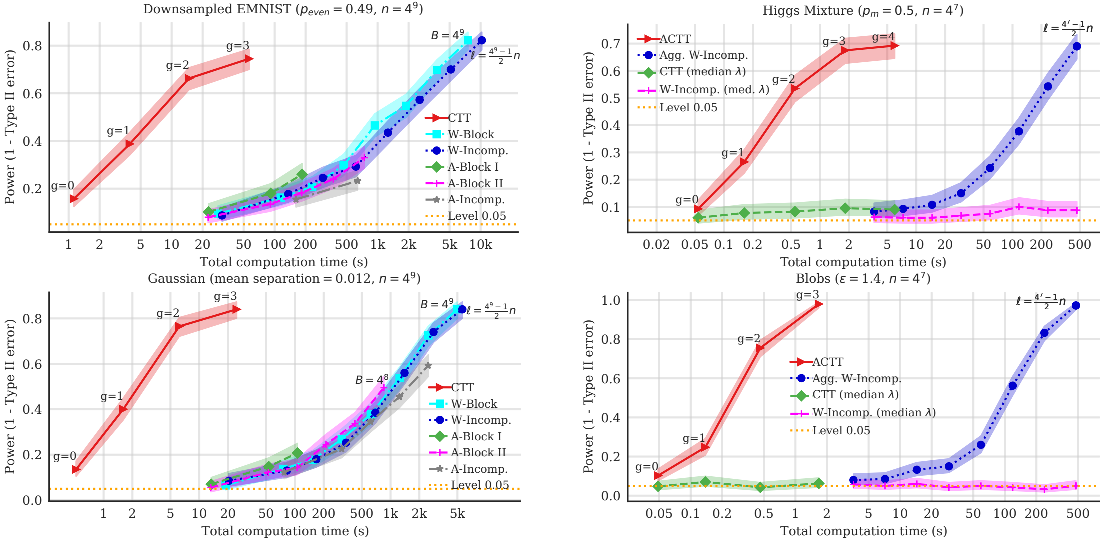
Distribution Compression in Near-linear Time [Shetty, Dwivedi, and Mackey, 2022]

Question: Can we speed up thinning algorithms without ruining their quality?



Compress++ reduces n^2 runtime to $n \log^3 n$, applies to any halving algorithm, and inflates error by at most a factor of 4

Compress Then Test [Domingo-Enrich, Dwivedi, and Mackey, 2023]



Compress Then Test (▶ above) yields powerful kernel tests in near-linear time

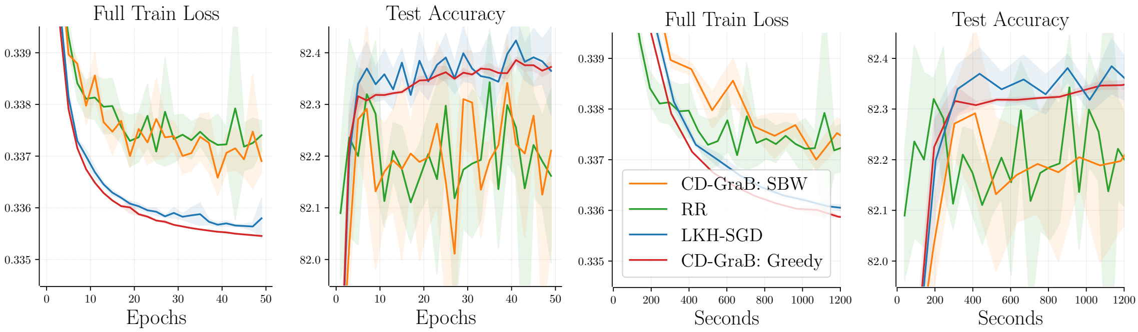
Compress Then Attend [Carrell, Gong, Shetty, Dwivedi, and Mackey, 2025]

Attention Algorithm	Top-1 Accuracy (%)	Layer 1 Runtime (ms)	Layer 2 Runtime (ms)
Exact	**82.55 ± 0.00**	**18.48 ± 0.12**	**1.40 ± 0.01**
Performer	80.56 ± 0.30	2.54 ± 0.01	0.60 ± 0.01
Reformer	81.47 ± 0.06	7.84 ± 0.03	1.53 ± 0.01
KDEformer	82.00 ± 0.07	5.39 ± 0.03	2.28 ± 0.03
Scatterbrain	82.05 ± 0.08	6.86 ± 0.02	1.55 ± 0.03
Thinformer (Ours)	**82.18 ± 0.05**	**2.06 ± 0.01**	**0.54 ± 0.00**

$$\text{Attention} = \text{softmax}\underbrace{\left(\begin{bmatrix}\text{query}_1\\ \text{query}_2\\ \vdots\\ \text{query}_n\end{bmatrix}\begin{bmatrix} | & | & & | \\ \text{key}_1 & \text{key}_2 & \cdots & \text{key}_n \\ | & | & & | \end{bmatrix}\right)}_{n\times n}\begin{bmatrix}\text{value}_1\\ \text{value}_2\\ \vdots\\ \text{value}_n\end{bmatrix}$$

Thinformer provides a fast, high-quality approximation to attention in Transformers

Compress Then Sort [Carrell, Gong, Shetty, Dwivedi, and Mackey, 2025]



Inspired by gradient balancing (CD-GraB) [Cooper, Guo, Pham, Yuan, Ruan, Lu, and De Sa, 2023], Kernel Halving SGD accelerates model training by reordering stochastic gradients

Conclusions

Summary

- New tools for summarizing a probability distribution more effectively than i.i.d. sampling or standard MCMC thinning
- Kernel thinning compresses an n point summary into a $\sqrt{n}$ point summary with better-than-i.i.d. approximation error
- Stein kernel thinning simultaneously compresses and reduces biases due to off-target sampling, tempering, or burn-in
- Compress++ speeds up thinning algorithms without ruining their quality
- CTT, Thinformer, and KH-SGD accelerate testing, Transformers, and training

Code:
- github.com/microsoft/goodpoints
- github.com/microsoft/thinformer
- github.com/microsoft/deepctt
- github.com/microsoft/khsgd

Future Directions

Many opportunities for future development

1. Value of swapping
   - KT-SWAP refinement stage typically leads to significant quality improvements over KT-SPLIT alone. Can we establish stronger guarantees for KT-SWAP?
2. Faster debiasing
   - Low-rank SKT [Li, Dwivedi, and Mackey] matches Stein KT guarantees in $\mathcal{O}(n^{1.5})$ time. Can we improve runtime further?
3. Weighted compression
   - For applications that support weights, can we establish stronger guarantees for weighted coresets?
   - e.g., weighted Stein Recombination and Stein Cholesky coresets can match SKT guarantees with as few as $s = \mathrm{polylog}(n)$ points instead of $s = \sqrt{n}$ [Li, Dwivedi, and Mackey].
4. Other metrics
   - For which other metrics is (significantly) better-than-i.i.d. compression achievable?

Related Work on MMD Coresets

Uniform distribution $\mathbb{P}$ on $[0,1]^d$: L^2 discrepancy MMD, s points

- Quasi-Monte Carlo [Chen, Skriganov, et al., 2002]: $\mathcal{O}(s^{-1}\log^{\frac{d-1}{2}} s)$
- Online Haar strategy [Dwivedi, Feldheim, Gurel-Gurevich, and Ramdas, 2019]: $\mathcal{O}(s^{-1}\log^{2d} s)$

Order $s^{-\frac{1}{2}}$ MMD coresets for general $\mathbb{P}$

- i.i.d. [Tolstikhin, Sriperumbudur, and Muandet, 2017], geometrically ergodic MCMC [Dwivedi and Mackey, 2024]
- Kernel herding [Chen, Welling, and Smola, 2010, Lacoste-Julien, Lindsten, and Bach, 2015], Stein points MCMC [Chen, Barp, Briol, Gorham, Girolami, Mackey, and Oates, 2019], Greedy sign selection [Karnin and Liberty, 2019]

Finite-dimensional linear kernels on $\mathbb{R}^d$: $\mathcal{O}(\sqrt{d}s^{-1}\log^{2.5} s)$, s points

- Discrepancy construction [Harvey and Samadi, 2014]: does not cover infinite-dimensional $\mathbf{k}$

Unknown coreset quality

- Super-sampling with a reservoir [Paige, Sejdinovic, and Wood, 2016]: coreset quality not analyzed
- Support points [Mak and Joseph, 2018]
  - Optimal s coreset has $o(s^{-\frac{1}{2}})$ energy distance MMD but no construction given
  - Practical convex-concave procedures not analyzed or shown to be optimal

References I

R. Alweiss, Y. P. Liu, and M. Sawhney. Discrepancy minimization via a self-balancing walk. *arXiv preprint arXiv:2006.14009*, 2020.

F. O. Campos, Y. Shiferaw, A. J. Prassl, P. M. Boyle, E. J. Vigmond, and G. Plank. Stochastic spontaneous calcium release events trigger premature ventricular complexes by overcoming electrotonic load. *Cardiovascular Research*, 107(1):175–183, 2015.

A. M. Carrell, A. Gong, A. Shetty, R. Dwivedi, and L. Mackey. Low-rank thinning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=iAkg2nVmvN.

W. W. L. Chen, M. M. Skriganov, et al. Explicit constructions in the classical mean squares problem in irregularities of point distribution. *Journal fur die Reine und Angewandte Mathematik*, 545:67–96, 2002.

W. Y. Chen, A. Barp, F.-X. Briol, J. Gorham, M. Girolami, L. Mackey, and C. Oates. Stein point Markov chain Monte Carlo. In *International Conference on Machine Learning*, pages 1011–1021. PMLR, 2019.

Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, page 109–116, Arlington, Virginia, USA, 2010. AUAI Press. ISBN 9780974903965.

K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness of fit. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

M. A. Colman. Arrhythmia mechanisms and spontaneous calcium release: Bi-directional coupling between re-entrant and focal excitation. *PLoS Computational Biology*, 15(8), 2019.

A. F. Cooper, W. Guo, K. Pham, T. Yuan, C. F. Ruan, Y. Lu, and C. De Sa. Coordinating distributed example orders for provably accelerated training. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

C. Domingo-Enrich, R. Dwivedi, and L. Mackey. Compress then test: Powerful kernel testing in near-linear time. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR, 25–27 Apr 2023.

R. Dwivedi and L. Mackey. Generalized kernel thinning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/pdf?id=IfNu7Dr-3fQ.

R. Dwivedi and L. Mackey. Kernel thinning. *Journal of Machine Learning Research*, 25(152):1–77, 2024.

R. Dwivedi, O. N. Feldheim, O. Gurel-Gurevich, and A. Ramdas. The power of online thinning in reducing discrepancy. *Probability Theory and Related Fields*, 174 (1):103–131, 2019.

D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

References II

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

B. C. Goodwin. Oscillatory behavior in enzymatic control process. *Advances in Enzyme Regulation*, 3:318–356, 1965.

J. Gorham and L. Mackey. Measuring sample quality with kernels. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375–395, 1999.

N. Harvey and S. Samadi. Near-optimal herding. In *Conference on Learning Theory*, pages 1165–1182, 2014.

R. Hinch, J. Greenstein, A. Tanskanen, L. Xu, and R. Winslow. A simplified local control model of calcium-induced calcium release in cardiac ventricular myocytes. *Biophysical journal*, 87(6):3723–3736, 2004.

S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian. Comparing distributions and shapes using the kernel distance. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 47–56, 2011.

Z. Karnin and E. Liberty. Discrepancy, coresets, and sketches in machine learning. In *Conference on Learning Theory*, pages 1975–1993. PMLR, 2019.

S. Lacoste-Julien, F. Lindsten, and F. Bach. Sequential kernel herding: Frank-Wolfe optimization for particle filtering. In *Artificial Intelligence and Statistics*, pages 544–552. PMLR, 2015.

L. Li, R. Dwivedi, and L. Mackey. Debiased distribution compression.

Q. Liu, J. D. Lee, and M. I. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

A. J. Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.

Y. Lu, W. Guo, and C. M. De Sa. Grab: Finding provably better data permutations than random reshuffling. *Advances in Neural Information Processing Systems*, 35:8969–8981, 2022.

S. Mak and V. R. Joseph. Support points. *The Annals of Statistics*, 46(6A):2562–2592, 2018.

S. A. Niederer, J. Lumens, and N. A. Trayanova. Computational models in cardiology. *Nature Reviews Cardiology*, 16(2):100–111, 2019.

References III

S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox. Scaling digital twins from the artisanal to the industrial. *Nature Computational Science*, 1(5):313–320, 2021.

E. Novak and H. Wozniakowski. Tractability of multivariate problems, volume ii: Standard information for functionals, european math. *Soc. Publ. House, Zürich*, 3, 2010.

C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society, Series B*, 79(3):695–718, 2017.

B. Paige, D. Sejdinovic, and F. Wood. Super-sampling with a reservoir. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 567–576, 2016.

J. M. Phillips. ε-samples for kernels. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1622–1632. SIAM, 2013.

J. M. Phillips and W. M. Tai. Improved coresets for kernel density estimates. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2718–2727. SIAM, 2018.

J. M. Phillips and W. M. Tai. Near-optimal coresets of kernel density estimates. *Discrete & Computational Geometry*, 63(4):867–887, 2020.

M. Riabiz, W. Chen, J. Cockayne, P. Swietach, S. A. Niederer, L. Mackey, and C. Oates. Optimal thinning of MCMC output. *To appear: Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2021.

G. O. Roberts and R. L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

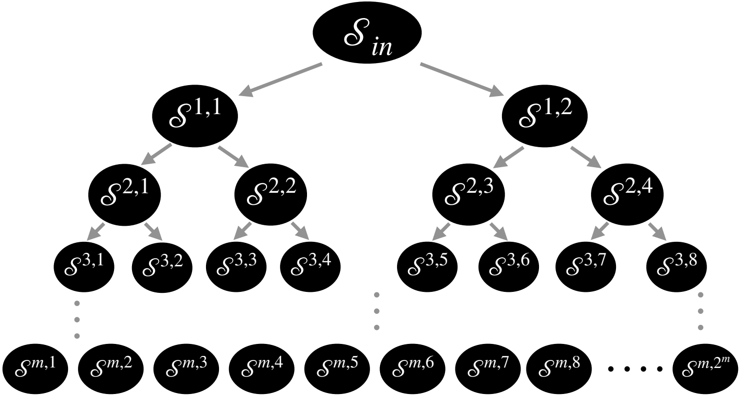
A. Shetty, R. Dwivedi, and L. Mackey. Distribution compression in near-linear time. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/pdf?id=lzupY5zjaU9.

W. M. Tai. New nearly-optimal coreset for kernel density estimation. *arXiv preprint arXiv:2007.08031*, 2020.

I. Tolstikhin, B. K. Sriperumbudur, and K. Muandet. Minimax estimation of kernel mean embeddings. *The Journal of Machine Learning Research*, 18(1): 3002–3048, 2017.

V. Volterra. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. 1926.

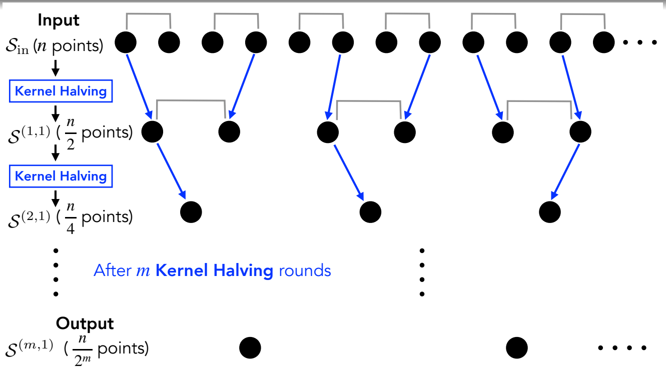
KT-SPLIT



KT-SPLIT partitions the input $\mathcal{S}_{\text{in}}$ recursively, first dividing the input sequence in half, then halving those halves into quarters, and so on

- Runs online: after i input points processed have output coresets of size $\frac{i}{2^m}$

KT-SPLIT



Each output coreset $\mathcal{S}^{(m,\ell)}$ is the result of repeated **kernel halving**

- On each halving round, remaining points are paired, and one point from each pair is selected using a new Hilbert space generalization of the self-balancing walk of Alweiss, Liu, and Sawhney [2020]
- Selection rule ensures that $\mathbb{P}_n\mathbf{k}_{\mathrm{rt}} - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}$ remains small with high probability

Kernel Halving with a Self-Balancing Hilbert Walk

Algorithm: Self-balancing Hilbert Walk [Dwivedi and Mackey, 2024]

Input: sequence of functions $(f_i)_{i=1}^{n/2}$ in Hilbert space $\mathcal{H}$, threshold sequence $(\mathfrak{a}_i)_{i=1}^{n/2}$

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

for $i = 1, 2, \ldots, n/2$ **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 $\mathfrak{a}_i$ to avoid this case 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 $\psi_{n/2}$, sum of signed input functions // $\psi_{n/2} = \sum_{i=1}^{n/2} \eta_i f_i$ with high probability

1. **Kernel Halving:** If $f_i = \mathbf{k}_{\mathrm{rt}}(x_{2i-1}, \cdot) - \mathbf{k}_{\mathrm{rt}}(x_{2i}, \cdot)$, half of input points $\mathcal{S}_{\mathrm{out}}$ given sign 1
   $\Rightarrow \frac{1}{n}\psi_{n/2} = \mathbb{P}_n \mathbf{k}_{\mathrm{rt}} - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}$ with $\mathbb{Q} = \frac{2}{n}\sum_{x \in \mathcal{S}_{\mathrm{out}}} \delta_x$
2. **Balance:** If $\mathcal{H} = \mathbf{k}_{\mathrm{rt}}$ RKHS, $\mathbb{P}_n \mathbf{k}_{\mathrm{rt}}(x) - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}(x)$ is $\mathcal{O}(\sqrt{\log(n)}/n)$ sub-Gaussian, $\forall x$
   - In contrast, i.i.d. signs η_i give $\mathbb{P}_n \mathbf{k}_{\mathrm{rt}}(x) - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}(x) = \Omega(1/\sqrt{n})$

Why the Square-root Kernel $\mathbf{k}_{\mathrm{rt}}$?

Theorem (L^∞ coresets for $(\mathbf{k}_{\mathrm{rt}}, \mathbb{P}_n)$ are MMD coresets for $(\mathbf{k}, \mathbb{P}_n)$ [Dwivedi and Mackey, 2024])

For any scalars $R, a, b \geq 0$ with $a + b = 1$, we have

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

for $v_d \triangleq \pi^{d/4}/\Gamma(d/2+1)^{1/2}$.

- **L^∞ error:** $\|\mathbb{P}_n \mathbf{k}_{\mathrm{rt}} - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}\|_\infty \triangleq \sup_{x \in \mathbb{R}^d} |\mathbb{P}_n \mathbf{k}_{\mathrm{rt}}(x) - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}(x)|$
- **Tail decay of $(\mathbb{P}_n, \mathbb{Q}, \mathbf{k}_{\mathrm{rt}})$:** $\tau_{\mathbb{P}_n}(R) \triangleq \mathbb{P}_n(\|X\|_2 \geq R)$
- **Effective radius:** Want $\tau_{\mathbf{k}_{\mathrm{rt}}}(aR), \tau_{\mathbb{P}_n}(bR), \tau_{\mathbb{Q}}(bR) = \mathcal{O}(\frac{1}{\sqrt{n}})$
  - $R = \mathcal{O}(1)$ for compact support, $R = \mathcal{O}(\log(n))$ for sub-exponential decay
- When $(\mathbb{P}_n, \mathbb{Q}, \mathbf{k}_{\mathrm{rt}})$ are compactly supported, $\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_n, \mathbb{Q}) = \mathcal{O}(\|\mathbb{P}_n \mathbf{k}_{\mathrm{rt}} - \mathbb{Q}\mathbf{k}_{\mathrm{rt}}\|_\infty)$

L^∞ Coresets from Kernel Halving

Theorem (L^∞ guarantees for kernel halving [Dwivedi and Mackey, 2024])

With high probability,

1. **Kernel halving yields a 2-thinned L^∞ coreset $\mathbb{Q}_{\mathrm{KH}}^{(1)}$ satisying**
$$\|\mathbb{P}_n \mathbf{k}_{\mathrm{rt}} - \mathbb{Q}_{\mathrm{KH}}^{(1)} \mathbf{k}_{\mathrm{rt}}\|_\infty \leq \|\mathbf{k}_{\mathrm{rt}}\|_\infty \cdot \frac{2}{n} \mathfrak{M}_{\mathbf{k}_{\mathrm{rt}}}(\mathbb{P}_n)$$
2. **Repeated kernel halving yields a 2^m-thinned L^∞ coreset $\mathbb{Q}_{\mathrm{KH}}^{(m)}$ satisfying**
$$\|\mathbb{P}_n \mathbf{k}_{\mathrm{rt}} - \mathbb{Q}_{\mathrm{KH}}^{(m)} \mathbf{k}_{\mathrm{rt}}\|_\infty \leq \|\mathbf{k}_{\mathrm{rt}}\|_\infty \cdot \frac{2^m}{n} \mathfrak{M}_{\mathbf{k}_{\mathrm{rt}}}(\mathbb{P}_n)$$

- $\mathfrak{M}_{\mathbf{k}_{\mathrm{rt}}}(\mathbb{P}_n) = \mathcal{O}(\sqrt{\log n})$ for compactly supported $(\mathbb{P}, \mathbf{k}_{\mathrm{rt}})$ and $\mathcal{O}(\log n)$ in general
- With $m = \frac{1}{2}\log_2(n)$ rounds, yields $\sqrt{n}$ points with $\mathcal{O}(n^{-\frac{1}{2}}\log(n))$ L^∞ error
  - An equal-sized i.i.d. sample has $\Omega(n^{-\frac{1}{4}})$ L^∞ error
- **Near-optimal:** any procedure outputting $\sqrt{n}$ points must suffer $\Omega(n^{-\frac{1}{2}})$ L^∞ error for some $\mathbb{P}_n$ [Phillips and Tai, 2020, Thm. 3.1]

MMD Coresets from Kernel Thinning

Theorem (MMD guarantee for kernel thinning [Dwivedi and Mackey, 2024])

Kernel thinning returns a coreset $\mathbb{Q}_{KT}$ with $\sqrt{n}$ points satisfying, with high probability,

$$\mathrm{MMD}_{\mathbf{k}}(\mathbb{P}_n, \mathbb{Q}_{KT}) = \begin{cases} \mathcal{O}(\sqrt{\frac{\log n}{n}}) & \text{for compact support } (\mathbb{P}, \mathbf{k}_{\mathrm{rt}}) \text{ (e.g., B-spline } \mathbf{k}) \\ \mathcal{O}(\frac{(\log n)^{\frac{d+2}{4}}\sqrt{\log\log n}}{\sqrt{n}}) & \text{for sub-Gaussian } (\mathbb{P}, \mathbf{k}_{\mathrm{rt}}) \text{ (e.g., Gaussian } \mathbf{k}) \\ \mathcal{O}(\frac{(\log n)^{\frac{d+1}{2}}\sqrt{\log\log n}}{\sqrt{n}}) & \text{for sub-exponential } (\mathbb{P}, \mathbf{k}_{\mathrm{rt}}) \text{ (e.g., Matérn } \mathbf{k}) \end{cases}$$

- An equal-sized i.i.d. sample has $\Omega(n^{-\frac{1}{4}})$ MMD
- Sub-exponential guarantees resemble the classical $\mathcal{O}(\frac{(\log n)^{\frac{d-1}{2}}}{\sqrt{n}})$ quasi-Monte Carlo error rates for uniform $\mathbb{P}$ on $[0,1]^d$ but apply to more general distributions on $\mathbb{R}^d$
- See the paper for non-asymptotic bounds with explicit constants and $\frac{n}{2^m}$ points

Related Work on L^∞ Coresets

L^∞ coresets for $\mathbb{P}_n$: $o(n^{-\frac{1}{4}})$ L^∞ error, $\sqrt{n}$ points

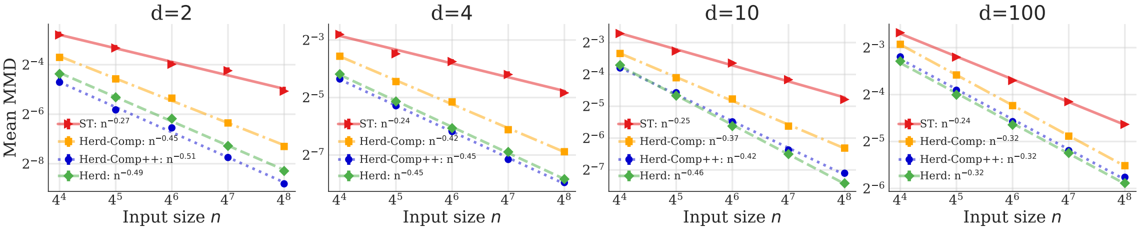
- Series of breakthroughs due to [Joshi, Kommaraji, Phillips, and Venkatasubramanian, 2011, Phillips, 2013, Phillips and Tai, 2018, 2020, Tai, 2020]

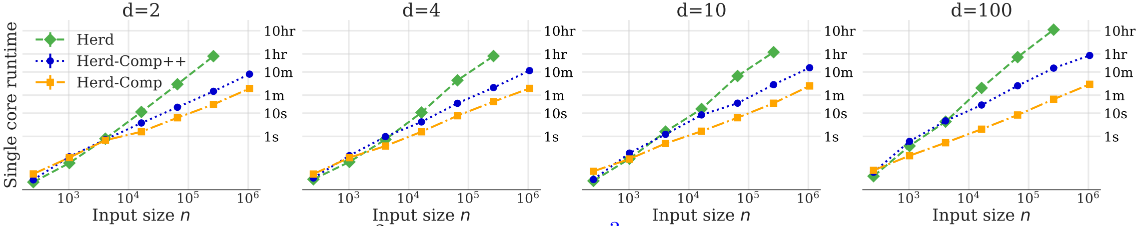
Best known L^∞ guarantees (for coreset of size $\sqrt{n}$)

- Phillips and Tai [2020]: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\sqrt{\log n})$ error, $\Omega(n^4)$ time, $\Omega(n^2)$ space
- Tai [2020] (Gaussian $\mathbf{k}$): $\mathcal{O}(2^d n^{-\frac{1}{2}}\sqrt{\log(d\log n)})$ error, $\Omega(\max(d^{5d}, n^4))$ time
- Both are offline and require rebalancing after approximate halving steps
- This work: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\log n)$ error, $\mathcal{O}(n^2)$ time, $\mathcal{O}(nd)$ space, online, exact halving
    - Sub-Gaussian $(\mathbf{k}_{\mathrm{rt}}, \mathbb{P})$: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\sqrt{\log n \log\log n})$ error
    - Compact support $(\mathbf{k}_{\mathrm{rt}}, \mathbb{P})$: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\sqrt{\log n})$ error

Distribution Compression in Near-linear Time [Shetty, Dwivedi, and Mackey, 2022]

Question: Can we speed up thinning algorithms without ruining their quality?





Compress++ reduces n^2 runtime to $n \log^3 n$, applies to any thinning algorithm (e.g., kernel herding), and inflates error by at most a factor of 4

Distribution Compression in Near-linear Time [Shetty, Dwivedi, and Mackey, 2022]

Algorithm 2: $\textsc{Compress}$: Given n points return thinned coreset of size $\sqrt{n}$

Input: halving algorithm $\textsc{Halve}$, point sequence $\mathcal{S}_{\text{in}}$ of size n

if $n = 1$ **then** **return** $\mathcal{S}_{\text{in}}$

Partition $\mathcal{S}_{\text{in}}$ into four arbitrary subsequences $\{\mathcal{S}_i\}_{i=1}^4$ each of size $n/4$

for $i = 1, 2, 3, 4$ **do**

  $\widetilde{\mathcal{S}}_i \leftarrow \textsc{Compress}(\mathcal{S}_i, \textsc{Halve})$ // return coresets of size $\sqrt{\frac{n}{4}}$

end

$\widetilde{\mathcal{S}} \leftarrow \textsc{Concatenate}(\widetilde{\mathcal{S}}_1, \widetilde{\mathcal{S}}_2, \widetilde{\mathcal{S}}_3, \widetilde{\mathcal{S}}_4)$ // coreset of size $2\sqrt{n}$

return $\textsc{Halve}(\widetilde{\mathcal{S}})$ // coreset of size $\sqrt{n}$

Error guarantees rely on unbiased halving ($\mathbb{E}[\mathbb{P}_{\textsf{Halve}}\mathbf{k} \mid \mathcal{S}_{\text{in}}] = \mathbb{P}_{in}\mathbf{k}$)

- Achieved for any halving algorithm by symmetrization: return either the outputted half or its complement with equal probability