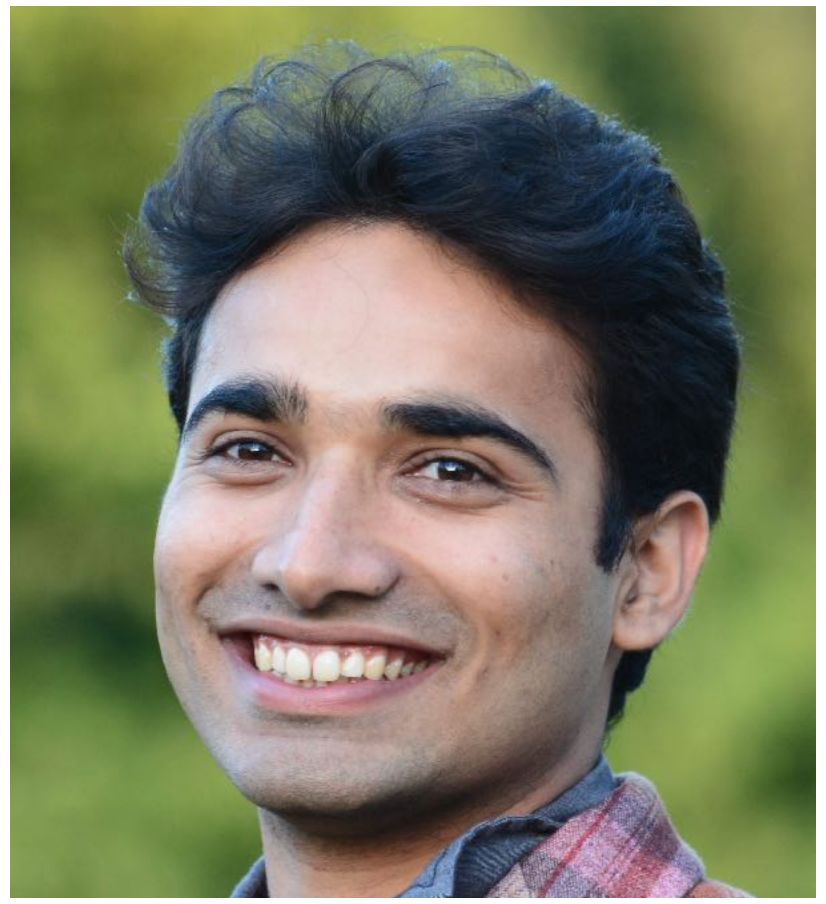
Generalized Kernel Thinning

Raaz Dwivedi, Lester Mackey
raaz@mit.edu



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

ICLR 2022

Motivation: Represent $\mathbb{P}^\star$ using a <u>few</u> high quality points $(x_i)_{i=1}^n$

- **I.I.D. sampling, and MCMC sampling** exhibit the **slow root-n Monte Carlo rate** $\left|\mathbb{P}^\star f - \mathbb{P}_n f\right| = \Theta(n^{-1/2})$, e.g., ~ 10^6 points for 0.1% error

- Computationally prohibitive for expensive f, and common fixes uniform thinning, or standard thinning—-choose every $t-$th point

- **Accuracy degrades with such thinning**–$\Theta(\sqrt{t/n})$ worst-case error–same as the slow rate with n/t points, e.g., $n^{-1/4}$ rate with $\sqrt{n}$ points

Minimax rates: $\Omega(n^{-1/2})$ for
- any compression scheme returning $\sqrt{n}$ points
- any approximation based on n i.i.d. points

Goal: Better than iid distribution compression

Given n points $(x_i)_{i=1}^{n}$ with empirical distribution $\mathbb{P}_{in} \triangleq \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i}$

Return a subset of size s with empirical distribution $\mathbb{P}_{out}$ such that

$$|\mathbb{P}_{in}f - \mathbb{P}_{out}f| = o(s^{-1/2})$$

for suitable function / function class

Many names:

- better than Monte Carlo points
- high quality coresets
- good prototypes

Prior strategies

- **Quasi Monte Carlo:** Better than Monte-Carlo error $\mathbb{P}^\star$ uniformly supported on $[0,1]^d$
- **Other strategies:**
    - Kernel herding, support points—-theory not well-understood for general kernels
    - Several schemes require $\mathbb{P}^\star$ with compact support or other restrictive assumptions
- **Kernel thinning [KT]: Near-optimal error** for
    - General target distributions on $\mathbb{R}^d$
    - Sufficiently smooth and decaying kernels on $\mathbb{R}^d$
- **This work: Generalize the KT results to arbitrary kernels on arbitrary domains!**

Kernel Thinning: A two-staged procedure

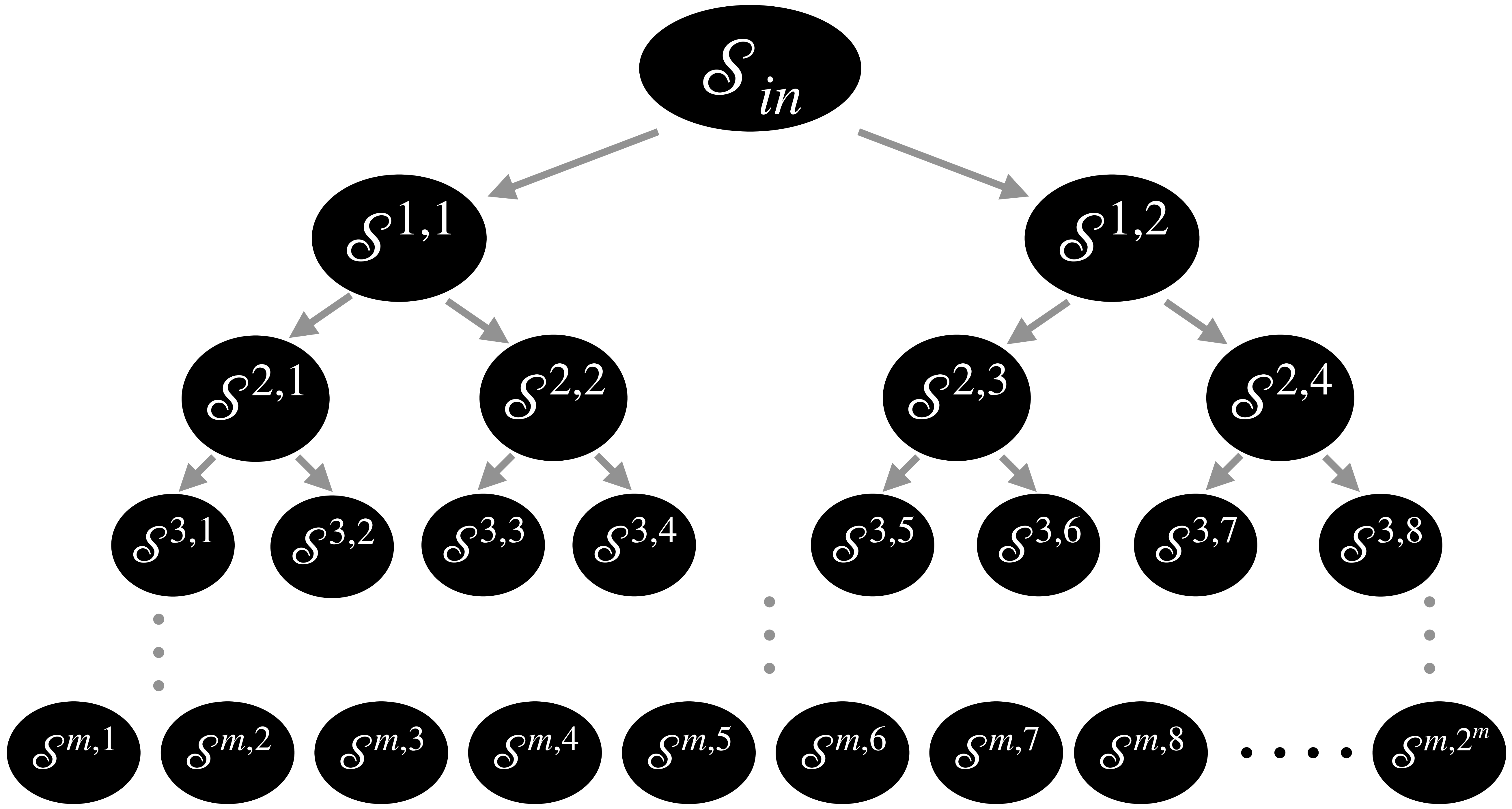
$x_1, x_2, \ldots, x_n$
reproducing kernel $\mathbf{k}$
$\mathbb{P}_{in} \triangleq \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i}$

→ **Kernel Thinning (KT)** →

Non-uniform sub-sample of size s

$y_1, \ldots, y_s$
$\mathbb{P}_{KT} \triangleq \frac{1}{s}\sum_{i=1}^{s}\delta_{y_i}$

- **Stage 1:** $m = \frac{1}{2}\log_2(n/s)$ recursive rounds of **non-uniform splitting** the parent coreset in two equal-sized children coresets
- **Stage 2:** Point-by-point **refinement** of the best child coreset



Original KT: Better than Monte Carlo error for smooth $\mathbf{k}$

$x_1, x_2, \ldots, x_n$

reproducing kernel $\mathbf{k}$

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

→ **Kernel Thinning (KT)** →

Non-uniform sub-sample of size s

$y_1, \ldots, y_s$

$\mathbb{P}_{KT} \triangleq \frac{1}{s}\sum_{i=1}^{s} \delta_{y_i}$

- Define $\mathbf{k}_{\alpha-rt} = \widehat{(\widehat{\mathbf{k}})^{\alpha}}$ where ^ denotes Fourier transform
- **Original KT:** $\alpha = 1/2$ to run the algorithm & provide better than Monte Carlo rate for the worst-case integration error in the reproducing kernel Hilbert space of $\mathbf{k}$
- Allows only sufficiently smooth kernels like Gaussian, inverse multiquadrics, …

Generalized kernel thinning

- Define $\mathbf{k}_{\alpha-rt} = \widehat{(\widehat{\mathbf{k}})^{\alpha}}$ where ^ denotes Fourier transform
- Allows $\alpha \in [1/2, 1]$ to run the algorithm
- Any fractional power or even the kernel itself can be used!
- Provides better than Monte Carlo guarantees for single function for arbitrary kernels on arbitrary domains!

Generalized kernel thinning

	Root KT [Original Algorithm]		
Kernel used in algorithm	$\mathbf{k}_{1/2-rt}$		
Single-function error	Same as worst-case error		
Worst-case integration error for $\mathbb{P}^\star$ supported on $\mathbb{R}^d$	$\widetilde{\mathcal{O}}(n^{-1/2})$ for sufficiently smooth $\mathbf{k}$		

Generalized kernel thinning

	Root KT [Original Algorithm]	Target KT	
Kernel used in algorithm	$\mathbf{k}_{1/2-rt}$	$\mathbf{k}$	
Single-function error	Same as worst-case error	$\sqrt{\frac{\log n}{n}}$ For arbitrary k on arbitrary domain	
Worst-case integration error for $\mathbb{P}^\star$ supported on $\mathbb{R}^d$	$\widetilde{\mathcal{O}}(n^{-1/2})$ for sufficiently smooth $\mathbf{k}$	$\sqrt{\frac{\log^{ad+b} n}{n}}$ Analytic $\mathbf{k}$ $\sqrt{\frac{n^{d/m}}{n}}$ m-times differentiable $\mathbf{k}$	

Generalized kernel thinning

https://arxiv.org/pdf/2110.01593.pdf https://github.com/microsoft/goodpoints

	Root KT [Original Algorithm]	Target KT	KT+ [Best of both worlds]
Kernel used in algorithm	$\mathbf{k}_{1/2-rt}$	$\mathbf{k}$	$\mathbf{k} + \mathbf{k}_{\alpha-rt}$ $\alpha \in (1/2,1)$
Single-function error	Same as worst-case error	$\sqrt{\frac{\log n}{n}}$ **For arbitrary k on arbitrary domain**	$\sqrt{\frac{\log n}{n}}$
Worst-case integration error for $\mathbb{P}^\star$ supported on $\mathbb{R}^d$	$\widetilde{\mathcal{O}}(n^{-1/2})$ for sufficiently smooth $\mathbf{k}$	$\sqrt{\frac{\log^{ad+b} n}{n}}$ Analytic $\mathbf{k}$ $\sqrt{\frac{n^{d/m}}{n}}$ m-times differentiable $\mathbf{k}$	**Min(Target KT Error, α-Root KT)**