Near-Optimal Compression In Near-Linear Time

Raaz Dwivedi, Lester Mackey, Abhishek Shetty
raaz@mit.edu

MSRI Workshop, Berkeley
Mar 8, 2022
Computational cardiology

Modeling **digital twin heart** to predict therapy response in a *non-invasive way* requires single-cell modeling. A common methodology:

- Estimate single cell model using Bayesian set-up: Use millions of Markov chain Monte Carlo (MCMC) points to approximate posterior $\mathbb{P}^*$

- Propagate uncertainty at heart-level by passing these points to the whole-heart simulator

\[
\mathbb{P}^* f \triangleq \int f(x) d\mathbb{P}^*(x) \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i) \triangleq \mathbb{P}_n f
\]

$x_i = \text{MCMC sample for single cell model parameters}$

$f = \text{heart simulator}$

Niederer et al., 2011; Augustin et al., 2016; Strocchi et al., 2020
Computational cardiology

Modeling **digital twin heart** to predict therapy response in a *non-invasive way* requires single-cell modeling.
A common methodology:

- Estimate single cell model using Bayesian set-up: Use millions of Markov chain Monte Carlo (MCMC) points to approximate posterior $\mathbb{P}^*$

- Propagate uncertainty at heart-level by passing these points to the whole-heart simulator

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

$\mathbb{x_i} = \text{MCMC sample for single cell model parameters}$
$f = \text{heart simulator}$

1 Million MCMC samples ~ 2 weeks
**Single** evaluation of $f$ ~ 5 weeks

Can **NOT** use all million samples....!
Goal: Represent $\mathbb{P}^*$ using a few high quality points $(x_i^*)_{i=1}^n$

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

- Prohibitive for computationally expensive $f$
Goal: Represent $\mathbb{P}^*$ using a few high quality points $(x_i)_i^n$

Common solutions: I.I.D. sampling, and MCMC sampling exhibit a slow root-$n$ Monte Carlo rate $\mathbb{P}^* f - \mathbb{P}_n f = \Theta(n^{-1/2})$, e.g., $\sim 10^6$ points for 0.1% error

- Prohibitive for computationally expensive $f$

Data compression: Approximate $\mathbb{P}^*$ by compressing given $n$ points

- Uniform thinning, or standard thinning—choose every $t$–th point
Goal: Represent $\mathbb{P}^\star$ using a few high quality points $(x_i)_{i=1}^n$

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

- Prohibitive for computationally expensive $f$

Data compression: Approximate $\mathbb{P}^\star$ by compressing given $n$ points

- 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 Monte Carlo rate with $n/t$ points; e.g., $n^{-1/4}$ rate with $\sqrt{n}$ points
Goal: **Represent \( \mathbb{P}^\star \) using a few high quality points \((x_i)_i^{n}\)**

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

- Prohibitive for computationally expensive \( f \)

**Data compression:** Approximate \( \mathbb{P}^\star \) by compressing given \( n \) points

- 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 Monte Carlo rate with \( n/t \) points; e.g., \( n^{-1/4} \) rate with \( \sqrt{n} \) points

Can we do better?
Minimax lower bounds

There exists some $\mathbb{P}^*$ such that the worst-case integration error

- Is $\Omega(n^{-1/2})$ for any compression scheme returning $\sqrt{n}$ points

[Philips and Tai, 2020]
Minimax lower bounds

There exists some $P^*$ 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 $n$ i.i.d. points
  [Tolstikhin, Sriperumbudur, and Muandet, 2017]
This talk: **Kernel thinning-Compress++**

- **KT-Compress++**: A practical strategy based on two new algorithms to provide near-optimal compression in near-linear time
  - **Kernel thinning (KT)**: Provides near-optimal compression
  - **Compress++**: Provides **significantly reduced runtime** for generic thinning algorithms with **minimal worsening of error**
This talk: Kernel thinning-Compress++

- **KT-Compress++**: A practical strategy based on two new algorithms to provide near-optimal compression in near-linear time

  - **Kernel thinning (KT)**: Provides near-optimal compression

  - **Compress++**: Provides significantly reduced runtime for generic thinning algorithms with minimal worsening of error

- Overall, KT-Compress++ a solution for finding
  - better than Monte Carlo points
  - high quality coresets
  - good prototypes
Problem set-up

• **Input:**

Points \( (x_i)_{i=1}^n \) with empirical distribution \( P_{\text{in}} \equiv \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \)

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

• **Goal:**

Return a subset of input points with size \( s \), empirical distribution \( P_{\text{out}} \) with error rate \( o(s^{-1/2}) \), i.e., better than Monte Carlo rate
Reproducing kernel Hilbert space (RKHS)

- RKHS of $k$ is given by $\mathcal{H}_k \triangleq \text{span}\{k(x, \cdot), x \in \mathcal{X}\}$

- $\mathcal{H}_k$ is dense in the space of continuous functions for universal $k$ like

\[
\text{Gaussian } k(x, y) = \exp\left(-\frac{1}{2}\|x - y\|^2\right); \text{ IMQ } k(x, y) = \frac{1}{(1 + \|x - y\|^2)^{1/2}}
\]

$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a reproducing kernel if the matrix $K = (k(x_i, x_j))_{i,j=1}^n$ is a symmetric positive definite matrix for any $n$ and any $(x_1, \ldots, x_n)$
Kernel Thinning

\[ x_1, x_2, \ldots, x_n \]

kernel \( k \)

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

Kernel Thinning (KT)

Non-uniform sub-sample of size \( s \)

\[ y_1, \ldots, y_s \]

\[ \mathbb{P}_{KT} \triangleq \frac{1}{s} \sum_{i=1}^{s} \delta_{y_i} \]
Kernel Thinning: A two-staged procedure

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

\[ x_1, x_2, \ldots, x_n \]

kernel \( k \)

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

Non-uniform sub-sample of size \( s \)

\[ y_1, \ldots, y_s \]

\[ \mathbb{P}_{KT} \triangleq \frac{1}{s} \sum_{i=1}^{s} \delta_{y_i} \]
Kernel Thinning: A two-staged procedure

- **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
Kernel Thinning: Better than Monte Carlo rate for \( \mathbb{P}_{in} \)

With \( n \) input points, \( s \) output points, with high probability over the randomness in KT, for any fixed \( g \in \mathbb{H}_k \) we have

\[
| \mathbb{P}_{in} g - \mathbb{P}_{KT} g | \lesssim \frac{1}{s} \cdot ||g||_k \sqrt{\|k\|_\infty} (\log s + \log \log (n/s)) \ll O\left(\frac{1}{\sqrt{s}}\right) 
\]

Monte Carlo rate

for any kernel on any space!
Kernel Thinning: **Better than Monte Carlo rate for** $\mathbb{P}_{in}$

With $n$ input points, $\sqrt{n}$ output points, with high probability over the randomness in KT, **for any fixed** $g \in \mathbb{H}_k$ we have

$$|\mathbb{P}_{in}g - \mathbb{P}_{KT}g| \lesssim \frac{1}{\sqrt{n}} \cdot \|g\|_k \sqrt{\|k\|_\infty \log n} \ll \mathcal{O}\left(\frac{1}{n^{1/4}}\right)$$

**Monte Carlo rate** (standard thinning rate)

for any kernel on any space!
Kernel Thinning: Better than Monte Carlo rate for $P_{in}$

With $n$ input points, $\sqrt{n}$ output points, with high probability over the randomness in KT, for any fixed $g \in \mathbb{H}_k$ we have

$$|P_{in}g - P_{KT}g| \lesssim \frac{1}{\sqrt{n}} \cdot \|g\|_k \sqrt{\|k\|_\infty} \log n \ll O\left(\frac{1}{n^{1/4}}\right)$$

(Monte Carlo rate (standard thinning rate))

If $|P^*g - P_{in}g| = O\left(\sqrt{\frac{\log n}{n}}\right)$ then $|P^*g - P_{KT}g| = O\left(\sqrt{\frac{\log n}{n}}\right)$
Monte Carlo input + KT ⇒ Better than Monte Carlo output for $\mathbb{P}^*$

With $n$ input points, $\sqrt{n}$ output points, with high probability over the randomness in KT, for any fixed $g \in \mathbb{H}_k$ we have

$$|\mathbb{P}_{\text{in}}g - \mathbb{P}_{\text{KT}}g| \lesssim \frac{1}{\sqrt{n}} \cdot \|g\|_k \sqrt{\|k\|_{\infty}} \log n \ll \mathcal{O} \left( \frac{1}{n^{1/4}} \right)$$

Monte Carlo rate (standard thinning rate)

If $|\mathbb{P}^*g - \mathbb{P}_{\text{in}}g| = \mathcal{O} \left( \sqrt{\frac{\log n}{n}} \right)$ then $|\mathbb{P}^*g - \mathbb{P}_{\text{KT}}g| = \mathcal{O} \left( \sqrt{\frac{\log n}{n}} \right)$

- This is the Monte Carlo rate for input!
- Easily satisfied for input from iid sampling, MCMC, quadrature methods …
Intuition: KT finds “diverse and representative” points
Worst-case error

• For points in $\mathbb{R}^d$, the worst-case error—Maximum Mean Discrepancy (MMD) error—in the reproducing kernel Hilbert space (RKHS) satisfies

$$\sup_{\|g\|_k \leq 1} |\mathbb{P}_{in} g - \mathbb{P}_{KT} g|$$
Worst-case error: $\widetilde{O}(n^{-1/2})$ with $\sqrt{n}$ points for decaying $k$

- For points in $\mathbb{R}^d$, the **worst-case error**—Maximum Mean Discrepancy (MMD) error—in the reproducing kernel Hilbert space (RKHS) satisfies

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

- For output size $s$, the MMD error is $\widetilde{O}(1/s)$

Assuming similar tails for $P_{in}$
Monte Carlo input + KT $\Rightarrow$ Better than Monte Carlo output for $\mathbb{P}^*$

- For points in $\mathbb{R}^d$, the worst-case error—Maximum Mean Discrepancy (MMD) error—in the reproducing kernel Hilbert space (RKHS) satisfies whenever $\sup_{\|g\|_k \leq 1} |\mathbb{P}^* g - \mathbb{P}_{KT} g| \lesssim_d \begin{cases} 
n^{-1/2} \sqrt{\log n} & \text{(Compactly supported; e.g., B-spline $k$)} 
n^{-1/2} \sqrt{\log d/2 + 1} \frac{n \log \log n}{n} & \text{(Sub-Gaussian tails; e.g., Gaussian $k$)} 
n^{-1/2} \sqrt{\log d + 1} \frac{n \log \log n}{n} & \text{(Sub-exponential tails; e.g., Matern $k$)} \end{cases}$

Assuming similar tails for $\mathbb{P}_{in}$.
Comparison with related work

- finding good approximations to $\mathbb{P}^*$ by thinning, reweighting or directly
Related work: $\sqrt{n}$ points with $O(n^{-1/4})$ 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]
  - 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: $\sqrt{n}$ points with $o(n^{-1/4})$ MMD

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

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

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

- **$(k, \mathbb{P}^*)$ 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 $O(\sqrt{\log n/n})$ integration-error for any fixed function in the RKHS for any kernel on any space (iid sampling gives $\Omega(n^{-1/4})$ error)

2. $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$-worst-case/MMD error for decaying kernels

3. Valid for non-uniform target distributions with unbounded support

4. Valid for infinite-dimensional smooth/decaying kernels

5. Valid for generic input points including iid/MCMC/quadrature etc with mild conditions

6. Requires only kernel evaluations to implement

7. Matches MMD lower bounds up to log factors

8. Matches $L^\infty$-error lower bounds up to log factors
KT vs iid: Gaussian $\mathbb{P}^\star$ in $\mathbb{R}^d$

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

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

(Worst-case error in the unit ball of Gaussian RKHS)
KT on MCMC samples from computational cardiology

In this setting with $d = 38$, standard thinning is already good (the chain is mixing slowly), but KT provides further improvement! Each point saves 1000s of CPU hours!!

*MCMC samples taken from Riabiz-Chen-Cockayne-Swietach-Niederer-Mackey-Oates, 2021*
KT drawback: $n^2$ runtime with $n$ input points

Y-axis = Runtime in log-scale (Runtime dominated by kernel evaluations)

Y-axis = Runtime in linear scale (seconds)
Compress++: Reducing runtime

![Graphs showing single core runtime vs input size for different dimensions](image-url)
Compress++: Reducing runtime with minimal loss in accuracy!!
Compress++: A recursive strategy to reduce runtime for generic thinning algorithms

Thinning algorithm with $n^2$ runtime and sub-Gaussian error $e(n)$

$ightarrow$ Compress++

$\sqrt{n}$-thinning algorithm with $n \log^3 n$ runtime and sub-Gaussian error $4e(n)$

Compress($S$, $g$, ALG):

- If $\text{size}(S) == 1$, Return $S$
- Else:
  - (i) Call Compress separately on 4 equal splits of $S$
  - (ii) Concatenate the 4 outputs from step (i)
  - (iii) Return Halved output of step (ii) using ALG
Summary:
KT-Compress++ provides near-optimal compression in near-linear time

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

$n$ points $\rightarrow$ Kernel Thinning (KT) $\rightarrow \sqrt{n}$ points with $\widetilde{O}\left(\frac{1}{\sqrt{n}}\right)$ error in $n^2$ time

$n$ points $\rightarrow$ KT-Compress++ $\rightarrow \sqrt{n}$ points with $\widetilde{O}\left(\frac{1}{\sqrt{n}}\right)$ error in $n \log^3 n$ time

The $\widetilde{O}$ notation hides logarithmic factors.

**Python**

```
pip install goodpoints
```

[Kernel Thinning, COLT 2021]
https://arxiv.org/abs/2105.05842

[Generalized Kernel Thinning, ICLR 2022]
https://arxiv.org/abs/2110.01593

[Distribution Compression In Near-linear Time, ICLR 2022]
Additional slides
Details of kernel thinning
**KT: A two-staged algorithm**

- **Input:** Kernel $k$, 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 $\text{MMD}_k$ 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:** $\Theta(n^2)$ 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
• 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]

![Diagram of KT-Split algorithm](image-url)
Algorithm 2: Self-balancing Hilbert Walk

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

\(\psi_0 \leftarrow 0 \in \mathcal{H}\)

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 \(|\alpha_i| > a_i\):

\[ \psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / a_i \]

else:

\[ \eta_i \leftarrow 1 \text{ with probability } \frac{1}{2} \left( 1 - \alpha_i / a_i \right) \text{ and } \eta_i \leftarrow -1 \text{ otherwise} \]

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

end

return \(\psi_n\), 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 \((a_i)_{i=1}^n\)

\[ \psi_0 \leftarrow 0 \in \mathcal{H} \]

for \(i = 1, 2, \ldots, n\) do

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

\[ \text{if } |\alpha_i| > a_i: \]

\[ \psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i/a_i \]

We choose \(a_i\) such that this step **does not** occur with high probability

\[ \text{else:} \]

\[ \eta_i \leftarrow 1 \text{ with probability } \frac{1}{2}(1 - \alpha_i/a_i) \text{ and } \eta_i \leftarrow -1 \text{ otherwise} \]

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

end

**return** \(\psi_n\), 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 \((\alpha_i)_{i=1}^n\)

\[
\psi_0 \leftarrow 0 \in \mathcal{H} \\
\text{for } i = 1, 2, \ldots, n \text{ do} \\
\quad \alpha_i \leftarrow \langle \psi_{i-1}, f_i \rangle_\mathcal{H} \quad // \text{Compute Hilbert space inner product} \\
\quad \text{if } |\alpha_i| > \alpha_i: \quad \text{We choose } \alpha_i \text{ such that this step does not occur with high probability} \\
\quad \quad \psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / \alpha_i \quad \text{else:} \\
\quad \quad \eta_i \leftarrow 1 \text{ with probability } \frac{1}{2} (1 - \alpha_i / \alpha_i) \quad \text{and } \eta_i \leftarrow -1 \text{ otherwise} \\
\quad \psi_i \leftarrow \psi_{i-1} + \eta_i f_i \\
\text{end} \\
\text{return } \psi_n, \text{ combination of signed input functions}

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

\[
\frac{1}{n} \psi = \frac{1}{n} \sum_{x \in \mathcal{S}_{\text{in}}} k(x, \cdot) - \frac{2}{n} \sum_{x \in \mathcal{S}_{\text{out}}} k(x, \cdot) = \mathbb{P}_{\text{in}} k - \mathbb{P}_{\text{out}} k
\]
Algorithm 2: Self-balancing Hilbert Walk

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

\[ \psi_0 \leftarrow 0 \in \mathcal{H} \]

for \(i = 1, 2, \ldots, n\) do

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

if \(|\alpha_i| > a_i:\)

\[ \psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / a_i \]

else:

\[ \eta_i \leftarrow 1 \quad \text{with probability} \quad \frac{1}{2} (1 - \alpha_i / a_i) \quad \text{and} \quad \eta_i \leftarrow -1 \quad \text{otherwise} \]

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

end

**return** \(\psi_n\), combination of signed input functions

\[ \bullet \ \textbf{Balance}: \text{If } k \text{ is a reproducing kernel, for all } g \in \mathbb{H}, \]

\[ \langle \psi_n, g \rangle_k = \mathbb{P}_{\text{in}} g - \mathbb{P}_{\text{out}} g \text{ is } \mathcal{O}(n^{-1} \cdot \sqrt{\log n \cdot \|g\|_k}) \text{-sub-Gaussian} \]

If \(\eta_i\) were chosen i.i.d., the sub-Gaussian parameter is \(\Omega(n^{-1/2})\)
Details of Compress++
Compress++: A simple two-stage algorithm

\[ n \text{ points, parameter } g, \]
halving algorithm HALVE \[ \rightarrow \]
Compress \[ \rightarrow \]
\[ 2^g \sqrt{n} \text{ points} \]

For example:
HALVE = Kernel thinning by a factor of 2
THIN = Kernel thinning by a factor of \(2^g\)
(other algorithms can be used too!)
Compress++: A simple two-stage algorithm

\[ n \text{ points, parameter } g, \]
halving algorithm HALVE

\[ \text{Compress} \]

\[ 2^g \sqrt{n} \text{ points} \]

\[ \text{Compress}(\mathcal{S}, g, \text{ALG}): \]
- If \( \text{size}(\mathcal{S}) = 4^g \), Return \( \mathcal{S} \)
- Else:
  (i) Call Compress separately on 4 equal splits of \( \mathcal{S} \)
  (ii) Concatenate the 4 outputs from step (i)
  (iii) Return Halved output of step (ii) using ALG

\[ 2^g \text{ thinning algorithm THIN} \]

\[ \sqrt{n} \text{ points} \]
Compress++: Informal guarantee

Under some mild conditions, with $g = \log \log n + 1$, we have

- **Sub-Gaussian error inflation by at most 4:**
  
  If $\text{MMD}_k(\mathbb{P}_n, \mathbb{P}_{\text{HALVE}}) \sim e_1(n)$ and $\text{MMD}_k(\mathbb{P}_n, \mathbb{P}_{\text{THIN}}) \sim e_2(n)$ then
  
  $\text{MMD}_k(\mathbb{P}_n, \mathbb{P}_{\text{Compress++}}) \sim 4 \max(e_1(n), e_2(n))$

\[
\text{MMD}_k(\mathbb{P}_{\text{in}}, \mathbb{P}_{\text{out}}) = \sup_{\|g\|_k \leq 1} |\mathbb{P}_{\text{in}} g - \mathbb{P}_{\text{out}} g|
\]
Compress++: Informal guarantee

Under some mild conditions, with $g = \log \log n + 1$, we have

- **Sub-Gaussian error inflation by at most 4:**
  
  If $\text{MMD}_k(\mathbb{P}_n, \mathbb{P}_{\text{HALVE}}) \sim e_1(n)$ and $\text{MMD}_k(\mathbb{P}_n, \mathbb{P}_{\text{THIN}}) \sim e_2(n)$ then
  
  $\text{MMD}_k(\mathbb{P}_n, \mathbb{P}_{\text{Compress++}}) \sim 4 \max(e_1(n), e_2(n))$

- **Quadratic reduction in runtime:**
  
  If runtime of HALVE and THIN with $n$ points is $\mathcal{O}(n^\tau)$ then the runtime of Compress++ with $n$ points is $\mathcal{O}(n^{\tau/2})$ if $\tau > 2$ and $\mathcal{O}(n \log^3 n)$ if $\tau = 2$. 
KT vs Compress ($g = 0$) vs Compress++ ($g = 4$)

The input algorithms Halve and Thin to Compress++ are derived from KT.
Results for Compress++ with kernel herding (Herd)
Lower bounds
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
Single function +
Additional MCMC Experiments
Better error for functions inside and outside of RKHS

(Gaussian $k$ with $\sigma^2 = 2d$ and standard Gaussian $P^*$)

\[ \|P - P_{out}\|_{X_1}, d = 2 \]
\[ \|P - P_{out}\|_{k(X')}, d = 2 \]
\[ \|P_{in} - P_{out}\|_{CF}, d = 2 \]
Better error for functions inside and outside of RKHS

(Gaussian $k$ with $\sigma^2 = 2d$ and standard Gaussian $\mathbb{P}^*$)
Better error for functions inside and outside of RKHS

(Gaussian $k$ with $\sigma^2 = 2d$ and standard Gaussian $\mathbb{P}^*$)
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]

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

MCMC samples taken from Riabiz-Chen-Cockayne-Swietach-Niederer-Mackey-Oates, 2021
Results for MCMC experiments
Details for KT result
Target KT MMD rates: $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ error

- More generally

$$\widetilde{O}\left(\frac{1}{\sqrt{n}}\right)$$ error rate for analytic kernels

$$\widetilde{O}\left(\frac{n^{d/2m}}{\sqrt{n}}\right)$$ for $m$-times differentiable kernels

- We state explicit constants with dependence on kernel hyper-parameters in the paper
## Generalized kernel thinning

<table>
<thead>
<tr>
<th>KT-Split kernel</th>
<th>Root KT</th>
<th>Target KT</th>
<th>KT+ [Best of both worlds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{rt}$</td>
<td>$k$</td>
<td>$k + k_{\alpha-rt}$</td>
<td></td>
</tr>
</tbody>
</table>

**Single-function error**
- Same as MMD error
- $\sqrt{\frac{\log n}{n}}$ for arbitrary $k$
- On arbitrary domain

**MMD error**
- See slide
- $\sqrt{\frac{\log^{ad+b} n}{n}}$ for analytic $k$
- $\sqrt{\frac{n^{dlm}}{m}}$ for $m$-times differentiable $k$
- $\text{Min}(\text{Target KT Error, } \alpha \text{-Root KT})$

\[
k(x, y) = \int k_{rt}(x, z)k_{rt}(z, y)dz \quad \text{and} \quad k_{\alpha-rt} = \hat{\mathbf{k}}^\alpha \text{ where } \hat{} \text{ denotes Fourier transform}
\]
Target KT or KT+: Better than Monte Carlo rate

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

smooth decaying \( k \)

Kernel Thinning (KT)

Non-uniform sub-sample of size \( s \)

\[ y_1, \ldots, y_s \]

For any fixed \( g \in \mathbb{H}_k \), with probability \( 1 - \delta \) over the randomness in KT, we have

\[
| \mathbb{P}_{in}g - \mathbb{P}_{KT}g | \leq \frac{1}{s} \cdot \| g \|_k \sqrt{\frac{8}{3} \| k \|_\infty \log \left( \frac{4}{\delta} \right)} \log \left( \frac{6s \log(n/s)}{\delta} \right)
\]

Much faster than the Monte Carlo rate for standard/uniform thinning \( \mathcal{O} \left( \frac{1}{\sqrt{s}} \right) \)
Target KT or KT+: Better than Monte Carlo rate

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

smooth decaying \( k \)

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

Kernel

Thinning

(KT)

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

\[ y_1, \ldots, y_{\sqrt{n}} \]

\[ P_{KT} := \frac{1}{\sqrt{n}} \sum_{i=1}^{\sqrt{n}} \delta_{y_i} \]

For any fixed \( g \in \mathbb{H}_k \), with probability \( 1 - \delta \) over the randomness in KT, we have

\[
|P_{in}g - P_{KT}g| \leq \frac{1}{\sqrt{n}} \cdot \|g\|_k \left( 8 \frac{\|k\|_\infty \log \left( \frac{4}{\delta} \right) \log \left( \frac{6\sqrt{n \log \sqrt{n}}}{\delta} \right)}{3} \right)
\]

Much faster than the Monte Carlo rate for standard/uniform thinning \( \mathcal{O}\left( \frac{1}{n^{1/4}} \right) \)
Properties of MMD

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

\[
\text{MMD}_k(P_{\text{in}}, P_{\text{out}}) = \sup_{\|g\|_k \leq 1} |P_{\text{in}}g - P_{\text{out}}g|
\]

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

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

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