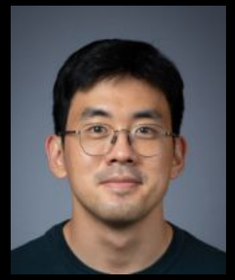
Supervised Kernel Thinning

Albert Gong



Kyuseong Choi

Raaz Dwivedi

Cornell University

CORNELL TECH

arxiv.org/abs/2410.13749

Kernel Methods for Supervised Learning

Data: n samples drawn i.i.d. from $\mathscr{D}$

Goal: Fit $y_i = f^\star(x_i) + \xi_i$ for $i = 1,2,\ldots,n$

1. Kernel **smoothing**
(Nadaraya-Watson regression)

$$\mathrm{NW}(x) = \frac{\sum_{i=1}^n y_i \mathbf{k}(x, x_i)}{\sum_{i=1}^n \mathbf{k}(x, x_i)}$$

2. Kernel ridge **regression**

$$\operatorname{argmin}_{f \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathscr{H}}^2$$

$$\mathrm{KRR}(x) = \sum_{i=1}^n \alpha_i \mathbf{k}(x, x_i)$$

$$\boldsymbol{\alpha} = (\mathbf{K} + n\lambda I)^{-1} \mathbf{y}$$

$$\mathbf{K} = [\mathbf{k}(x_i, x_j)]_{i,j=1}^n$$

$\mathbf{k}$ is a reproducing kernel

any symmetric ($\mathbf{k}(z, z') = \mathbf{k}(z', z)$) and positive semidefinite function

Computational Bottlenecks

Training: $\mathcal{O}(n)$ (to store points)

Inference: $\mathcal{O}(n)$ (per query)

1. Kernel **smoothing**
(Nadaraya-Watson regression)

$$\text{NW}(x) = \frac{\sum_{i=1}^{n} y_i \mathbf{k}(x, x_i)}{\sum_{i=1}^{n} \mathbf{k}(x, x_i)}$$

Efficient data structures [Kpotufe '09]
Coresets [Zheng & Phillips '17]
Nyström subsampling [Williams & Seeger '01, El Alaoui & Mahoney '15, Avron et al. '17]
Random Fourier features [Rahimi & Recht '07]
Divide-and-Conquer [Zhang et al. '15]
Iterative methods with preconditioning [Rudi et al. '17, Diaz et al. '23, Abedsoltan et al. '23]

Training: $\mathcal{O}(n^3)$ (to invert matrix)

Inference: $\mathcal{O}(n)$ (per query)

2. Kernel ridge **regression**

$$\text{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2$$

$$\text{KRR}(x) = \sum_{i=1}^{n} \alpha_i \mathbf{k}(x, x_i)$$

$$\boldsymbol{\alpha} = (\mathbf{K} + n\lambda I)^{-1} \mathbf{y}$$

$$\mathbf{K} = [\mathbf{k}(x_i, x_j)]_{i,j=1}^{n}$$

Our Strategy: Replace $\{(x_i, y_i)\}_{i=1}^{n}$ with $\{(x_i', y_i')\}_{i=1}^{n_{\text{out}}}$ $n_{\text{out}} = \sqrt{n}$

Training: ~~$\mathcal{O}(n)$~~ $\tilde{\mathcal{O}}(n)$

Inference: ~~$\mathcal{O}(n)$~~ $\mathcal{O}(\sqrt{n})$

1. Kernel **smoothing**
(Nadaraya-Watson regression)

$$\text{NW}(x) = \frac{\sum_{i=1}^{n} y_i \mathbf{k}(x, x_i)}{\sum_{i=1}^{n} \mathbf{k}(x, x_i)}$$

Training: ~~$\mathcal{O}(n^3)$~~ $\mathcal{O}(n^{3/2})$

Inference: ~~$\mathcal{O}(n)$~~ $\mathcal{O}(\sqrt{n})$

2. Kernel ridge **regression**

$$\text{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2$$

$$\text{KRR}(x) = \sum_{i=1}^{n} \alpha_i \mathbf{k}(x, x_i)$$

$$\boldsymbol{\alpha} = (\mathbf{K} + n\lambda I)^{-1} \mathbf{y}$$

$$\mathbf{K} = [\mathbf{k}(x_i, x_j)]_{i,j=1}^{n}$$

How to choose $\{(x_i', y_i')\}_{i=1}^{n_{\text{out}}}$? Supervised Kernel Thinning!

```
import goodpoints
X_thin, y_thin =
goodpoints.compresspp
(X,y,'rbf')
```

$\tilde{\mathcal{O}}(n_{\text{out}}^2)$, so near-linear when $n_{\text{out}} = n^{1/2}$

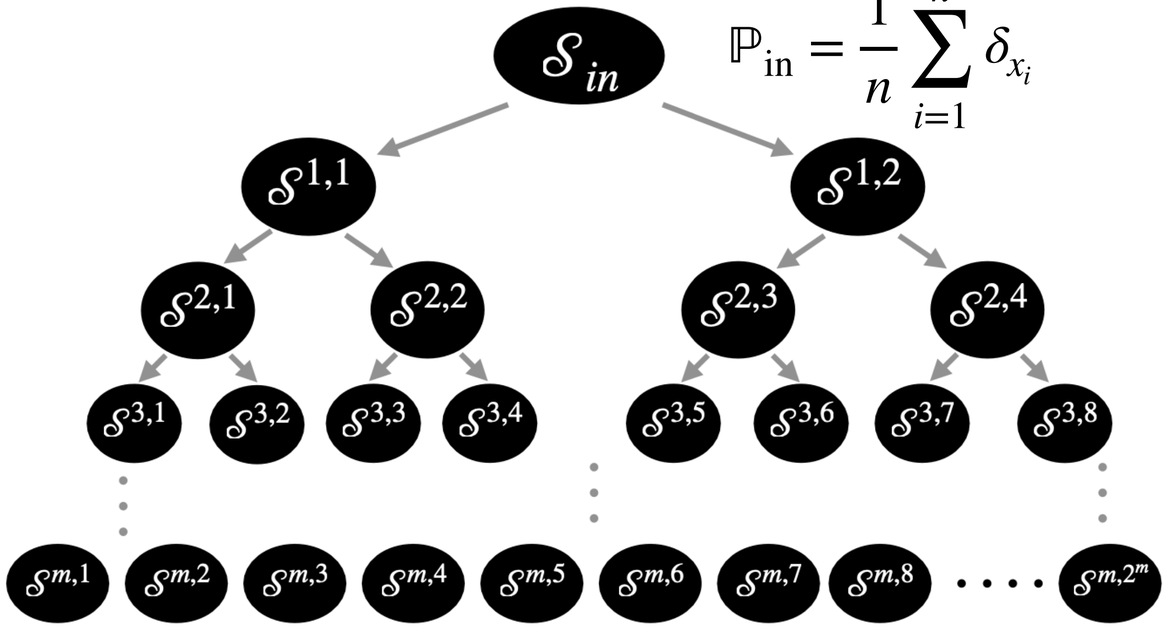
(Unsupervised) Kernel Thinning

Dwivedi & Mackey ’21, ’22, ’24;
Shetty-Dwivedi-Mackey ’22;
Domingo-Enrich-Dwivedi-Mackey ’23
Li-Dwivedi-Mackey ’24

Input: $\{x_i\}_{i=1}^n$ and kernel $\mathbf{k}$
Output: $\{x'_i\}_{i=1}^{n_{\text{out}}}$

Uniform (i.i.d.) subsampling:
$O(n_{\text{out}})$ time and $\Omega(n_{\text{out}}{}^{-1/2})$ error

Recursive halving via Self-Balancing Hilbert Walk



Better-than-i.i.d. rates for a large class of kernels

$$|\mathbb{P}_{\text{in}} f - \mathbb{P}_{\text{out}} f| \lesssim \frac{\|f\|_{\mathbf{k}}\sqrt{\log n_{\text{out}}}}{n_{\text{out}}}$$

Used to prove near-minimax rates for kernel density estimation and integration tasks

+ Greedy post-processing $\longrightarrow$ $\mathbb{P}_{\text{out}} = \frac{1}{n_{\text{out}}}\sum_{i=1}^{n_{\text{out}}}\delta_{x'_i}$

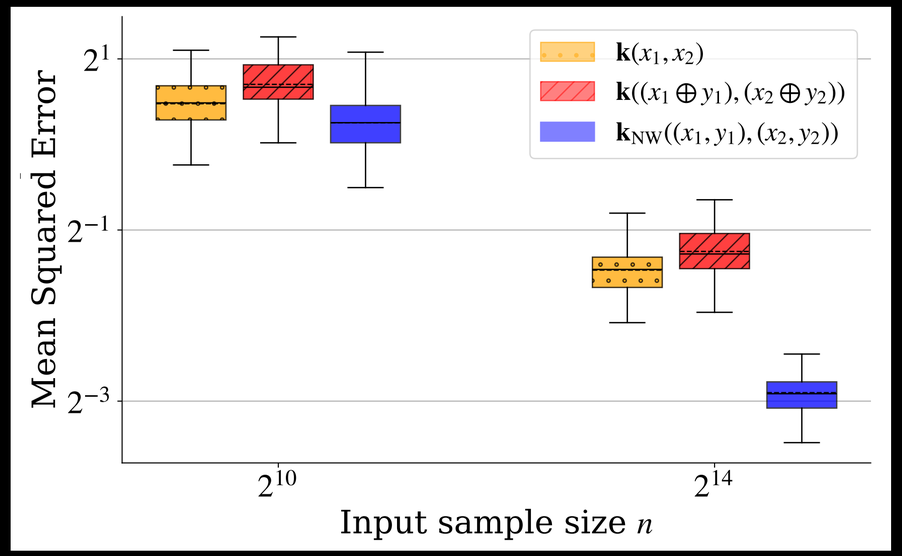
Runtime: $\mathcal{O}(n_{\text{out}}^2 \log^3 n_{\text{out}})$

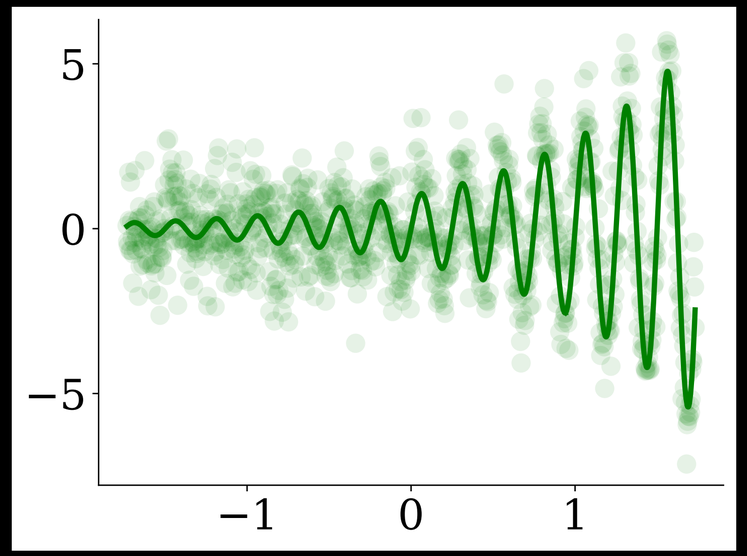
$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) \approx \frac{1}{n_{\text{out}}}\sum_{i=1}^{n_{\text{out}}} f(x_i')$$

when f lies in the RKHS of $\mathbf{k}$

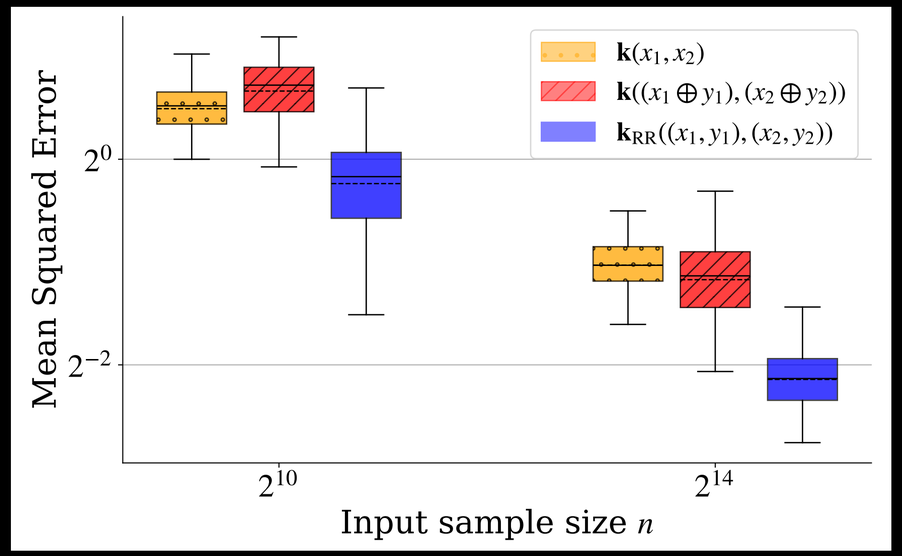
Which kernel should we use?

1. Kernel smoothing





2. Kernel ridge regression



Baseline #1: Apply KT on $\{x_i\}_{i=1}^{n}$

Baseline #2: Apply KT on $\{x_i \oplus y_i\}_{i=1}^{n}$

What does naïve KT miss?

$$\mathrm{NW}(\,\cdot\,) = \frac{\frac{1}{n}\sum_{i=1}^{n} y_i \mathbf{k}(\,\cdot\,, x_i)}{\frac{1}{n}\sum_{i=1}^{n} \mathbf{k}(\,\cdot\,, x_i)} = \frac{\mathbb{P}_{\mathrm{in}}(y\mathbf{k})(\,\cdot\,)}{\mathbb{P}_{\mathrm{in}}\mathbf{k}(\,\cdot\,)}$$

1. But $y \cdot \mathbf{k}$ is not in the RKHS of $\mathbf{k}$
2. However, it is in the RKHS of $y_1 y_2 \cdot \mathbf{k}(x_1, x_2)$!
3. And both numerator and denominator functions lie in the RKHS of

$$y_1 y_2 \cdot \mathbf{k}(x_1, x_2) + \mathbf{k}(x_1, x_2)$$

KT-NW is better than uniform subsampling

Informal Theorem. When

- p_X has compact support,
- $f^\star$ is β Holder for $\beta \in (0,2]$,
- $\mathbf{k}$ is a reproducing kernel with compact support.

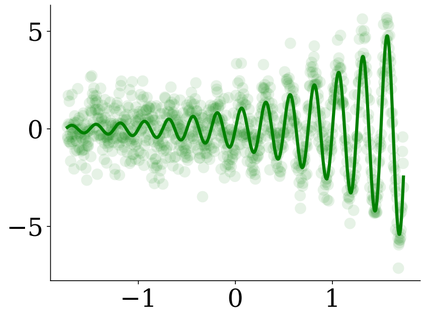
Then

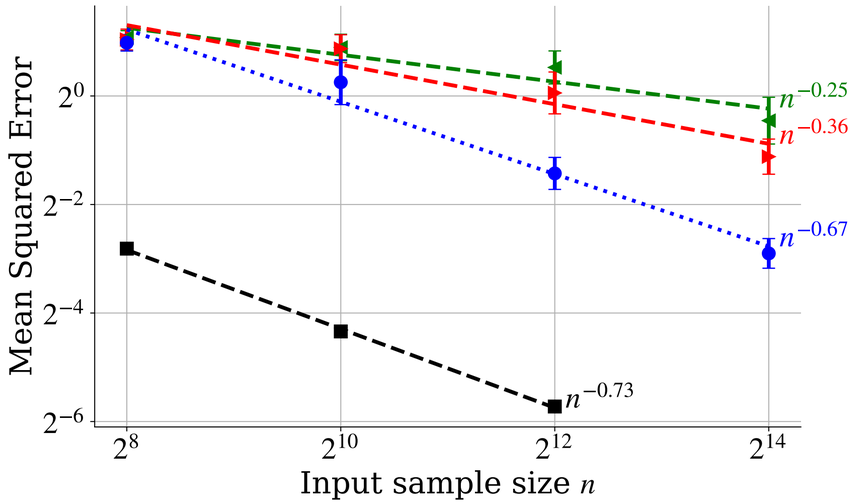
$$\mathbb{E}[\hat{f}_{\mathrm{KT}}(X) - f^\star(X)]^2 \lesssim n^{-\frac{\beta}{\beta+d}}$$

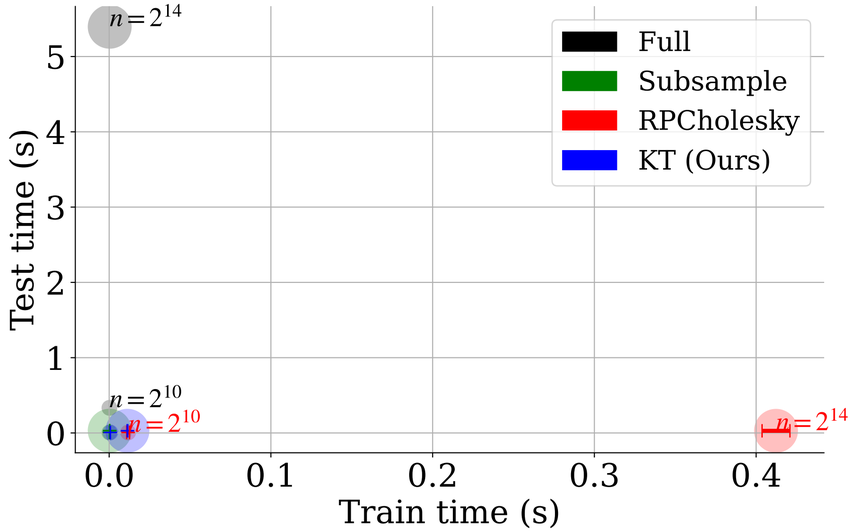
Minimax rate: $n^{-\frac{2\beta}{2\beta+d}}$

Uniform subsampling rate: $n^{-\frac{\beta}{2\beta+d}}$

Our results can be relaxed to sub-exponential and heavy-tails when $\beta > d/2$





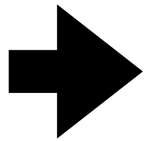


Wendland(0) kernel with cross validation

Designing the right kernel for kernel ridge regression

$$\text{KRR}(x) = \sum_{i=1}^{n} \alpha_i \mathbf{k}(x, x_i),$$

$$\text{where } \boldsymbol{\alpha} = (\mathbf{K} + n\lambda \mathbf{I})^{-1} \boldsymbol{y}$$



$$\text{KT-KRR}(x) = \sum_{i=1}^{n_{\text{out}}} \alpha'_i \mathbf{k}(x, x'_i),$$

$$\text{where } \boldsymbol{\alpha}' = (\mathbf{K}' + n\lambda' \mathbf{I})^{-1} \boldsymbol{y}'$$

Can this be expressed as a simple function average?

Idea: Thin the Training Loss

$$\frac{1}{n}\sum_{i=1}^{n} f^2(x_i) - 2y_i \cdot f(x_i) + y_i^2 \quad \Rightarrow \quad \frac{1}{n_{\text{out}}}\sum_{i=1}^{n_{\text{out}}} f^2(x_i') - 2y_i' \cdot f(x_i') + (y_i')^2$$

But which kernel should we choose?

f^2 lies in RKHS of $\mathbf{k}^2(x_1, x_2)$

$y \cdot f(x)$ lies in RKHS of $y_1 y_2 \cdot \mathbf{k}(x_1, x_2)$ $\Longrightarrow$ Loss function lies in the RKHS of $\mathbf{k}^2(x_1, x_2) + y_1 y_2 \cdot \mathbf{k}(x_1, x_2) + (y_1 y_2)^2$

y^2 lies in RKHS of $(y_1 y_2)^2$

Guarantees for KT-Kernel Ridge Regression

For rank m kernel $\mathbf{k}$, $n_{\text{out}} = \sqrt{n}$

$$\mathbb{E}[\hat{f}_{\text{KT}}(X) - f^\star(X)]^2 \lesssim \frac{m}{n}\|f^\star\|_2^2$$

Minimax rate: $\sigma^2\frac{m}{n}$

Uniform subsampling rate: $\sigma^2\frac{m}{\sqrt{n}}$

KT-KRR is nearly minimax in runtime $O(n^{3/2})$ — a quadratic speedup over full KRR

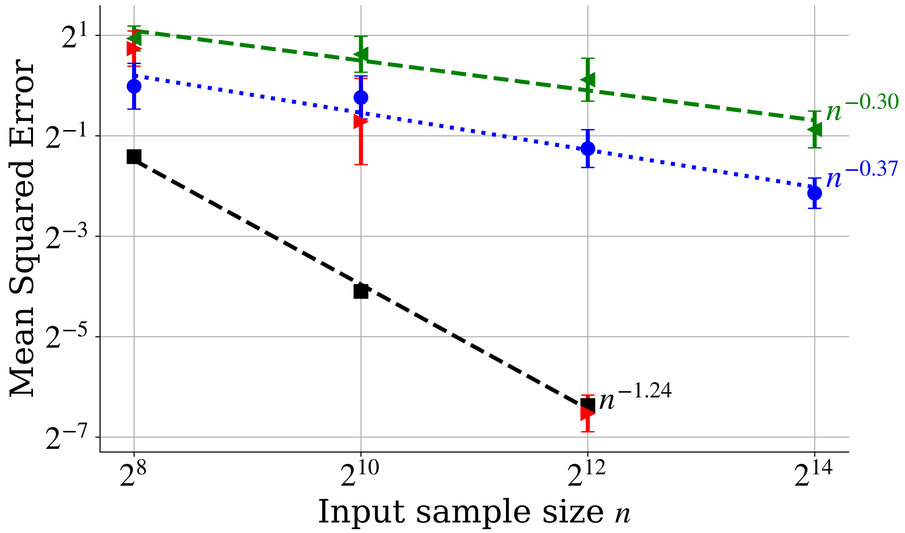
Additive guarantee for KT

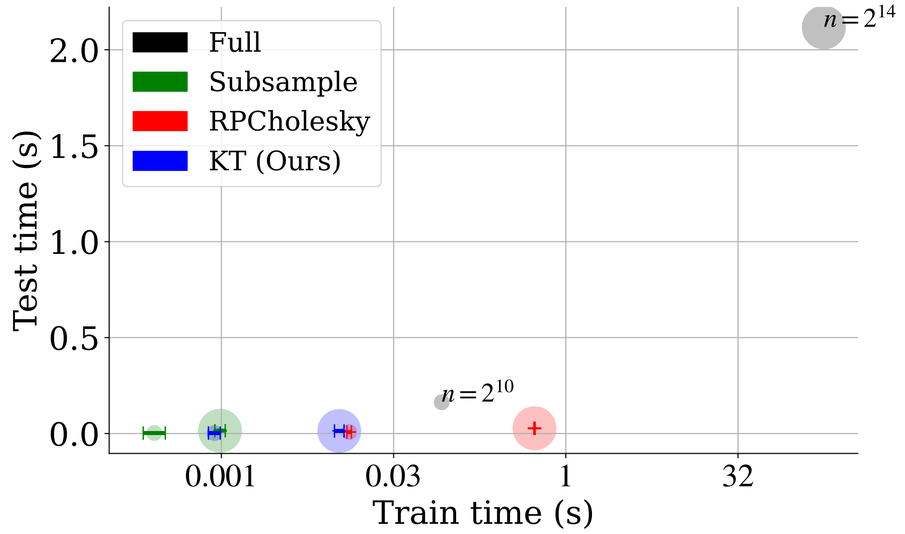
$$\|\mathbb{P}_{\text{in}} f - \mathbb{P}_{\text{out}} f\| \lesssim \frac{\|f\|_{\mathbf{k}}\sqrt{\log n_{\text{out}}}}{n_{\text{out}}}$$

New: Multiplicative guarantee for KT

$$(1 - \frac{1}{n_{\text{out}}})^2 \leq \frac{\frac{1}{n_{\text{out}}}\sum_{i=1}^{n_{\text{out}}} f^2(x_i')}{\frac{1}{n}\sum_{i=1}^{n} f^2(x_i)} \leq (1 + \frac{1}{n_{\text{out}}})^2$$

Results for KT-Kernel Ridge Regression





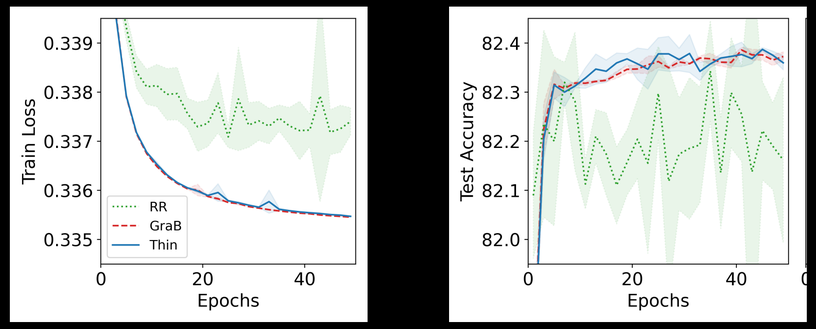
Gaussian kernel with cross validation

Method	Test Error (%)	Training Time (s)
RPCholesky	19.99 ± 0.00	3.46 ± 0.03
FALKON	19.99 ± 0.00	5.06 ± 0.02
CG	20.35 ± 0.00	6.16 ± 0.03
ST-KRR	22.71 ± 0.30	0.09 ± 0.00
KT-KRR (Ours)	22.00 ± 0.21	1.79 ± 0.00

SUSY dataset $(d = 18, n = 5 \times 10^6)$

Summary: Find function averages in your problem!

- **Supervised Kernel Thinning**: Thin-then-fit to speed up regression
  - Key idea: Use KT with an appropriate kernel
- Upcoming: **Low-Rank Thinning** to speed up SGD, Transformers
  - Key idea: KT is adaptive to low-rank structures



KT speeds up training time for SGD

Attention Algorithm	Top-1 Acc. (%)†	Attn. Layer 1 (ms)⋆	Attn. Layer 2 (ms)⋆
Exact	82.55 ± 0	21.00 ± 0.07	1.50 ± 0.01
Performer	80.53 ± 0.38	3.50 ± 0.01	0.61 ± 0.01
Reformer	81.47 ± 0.05	11.64 ± 0.03	2.29 ± 0.02
ScatterBrain	82.01 ± 0.08	9.47 ± 0.05	1.81 ± 0.03
KDEformer	82.04 ± 0.04	6.36 ± 0.08	2.36 ± 0.03
Compressformer (Ours)	82.21 ± 0.03	4.82 ± 0.03	0.76 ± 0.01

KT speeds up attention in Transformers

Thank you!