Rehashing Kernel Evaluation in High Dimensions

Paris Siminelakis*
Ph.D. Candidate

Kexin Rong*, Peter Bailis, Moses Charikar, Phillip Levis
(Stanford University)




ICML @ Long Beach, California

June 11, 2019

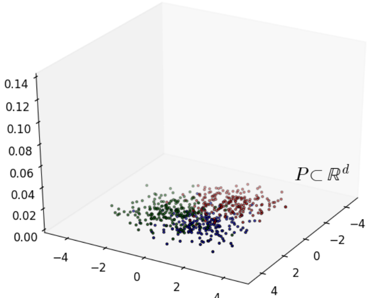
* equal contribution.

Kernel Density Function

$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

n points

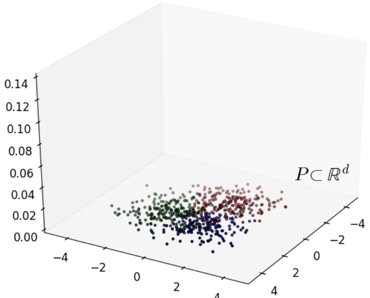


Kernel Density Function

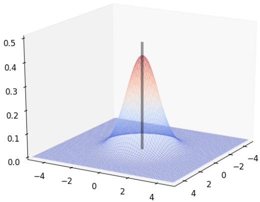
$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

n points



kernel *k*

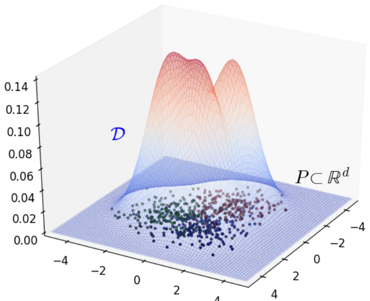


Kernel Density Function

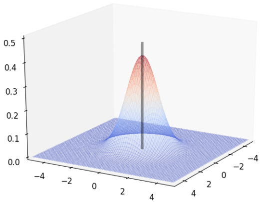
$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

n points



kernel k

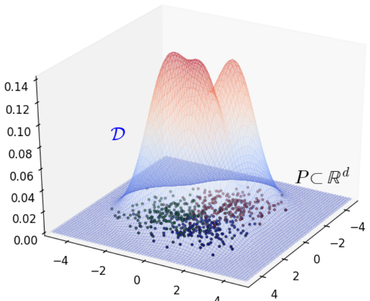


Kernel Density Function

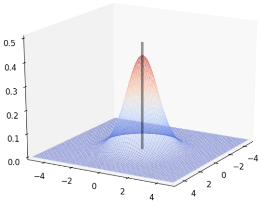
$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P(q) = \sum_{i=1}^{n} \left(\frac{1}{n}\right) k(x_i, q)$$

n points



kernel k



Kernel Density Evaluation

$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

Where is it used?

1. Non-parametric density estimation $\mathrm{KDF}_P(q)$
2. Kernel methods $f(x) = \sum_i \alpha_i \phi(\|x - x_i\|)$
3. Comparing point sets (distributions) with "Kernel Distance"

Kernel Density Evaluation

$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

Where is it used?

1. Non-parametric density estimation $\mathrm{KDF}_P(q)$
2. Kernel methods $f(x) = \sum_i \alpha_i \phi(\|x - x_i\|)$
3. Comparing point sets (distributions) with "Kernel Distance"

Kernel Density Evaluation

$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

Where is it used?

1. Non-parametric density estimation $\mathrm{KDF}_P(q)$
2. Kernel methods $f(x) = \sum_i \alpha_i \phi(\|x - x_i\|)$
3. Comparing point sets (distributions) with "Kernel Distance"

Kernel Density Evaluation

$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

Where is it used?

1. Non-parametric density estimation $\mathrm{KDF}_P(q)$
2. Kernel methods $f(x) = \sum_i \alpha_i \phi(\|x - x_i\|)$
3. Comparing point sets (distributions) with "Kernel Distance"

Kernel Density Evaluation

$P = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

Where is it used?

1. Non-parametric density estimation $\mathrm{KDF}_P(q)$
2. Kernel methods $f(x) = \sum_i \alpha_i \phi(\|x - x_i\|)$
3. Comparing point sets (distributions) with "Kernel Distance"

Evaluating at a single point requires $O(n)$

Kernel Density Evaluation

$P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, $u \geq 0$, query point q

$$\mathrm{KDF}_P^u(q) = \sum_{i=1}^{n} u_i k(x_i, q)$$

Where is it used?

1. Non-parametric density estimation $\mathrm{KDF}_P(q)$
2. Kernel methods $f(x) = \sum_i \alpha_i \phi(\|x - x_i\|)$
3. Comparing point sets (distributions) with "Kernel Distance"

How fast can we **approximate** KDF?

Methods for Fast Kernel Evaluation

$P \subset \mathbb{R}^d$, $\epsilon > 0 \Rightarrow (1 \pm \epsilon)$-approx to $\mu := \mathrm{KDF}_P(q)$ for any $q \in \mathbb{R}^d$

Methods for Fast Kernel Evaluation

$P \subset \mathbb{R}^d$, $\epsilon > 0 \Rightarrow (1 \pm \epsilon)$-approx to $\mu := \mathrm{KDF}_P(q)$ for any $q \in \mathbb{R}^d$

Space Partitions
$\log(1/\mu\epsilon)^{O(d)}$

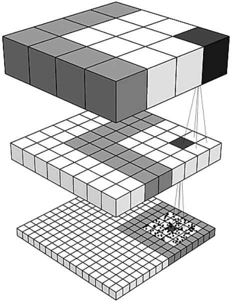
- FMM [Greengard, Rokhlin'87]
- Dual-Tree [Lee, Gray, Moore'06]
- FIG-Tree [Moriaru et al. NeurIPS'09]

Slow in high dim

Methods for Fast Kernel Evaluation

$P \subset \mathbb{R}^d$, $\epsilon > 0 \Rightarrow (1 \pm \epsilon)$-approx to $\mu := \mathrm{KDF}_P(q)$ for any $q \in \mathbb{R}^d$

Space Partitions
$\log(1/\mu\epsilon)^{O(d)}$



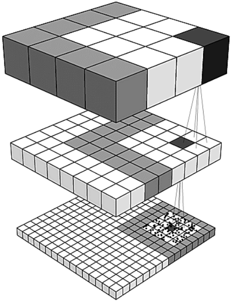
img: computer.org

Slow in high dim

Methods for Fast Kernel Evaluation

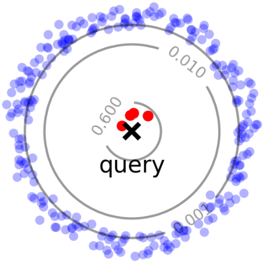
$P \subset \mathbb{R}^d$, $\epsilon > 0 \Rightarrow (1 \pm \epsilon)$-approx to $\mu := \mathrm{KDF}_P(q)$ for any $q \in \mathbb{R}^d$

Space Partitions
$\log(1/\mu\epsilon)^{O(d)}$



img: computer.org

Random Sampling
$1/\mu\epsilon^2$



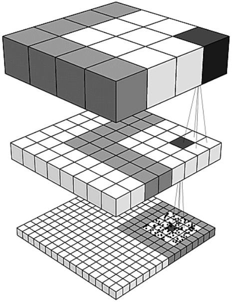
Slow in high dim

Linear in $1/\mu$

Methods for Fast Kernel Evaluation

$P \subset \mathbb{R}^d$, $\epsilon > 0 \Rightarrow (1 \pm \epsilon)$-approx to $\mu := \mathrm{KDF}_P(q)$ for any $q \in \mathbb{R}^d$

Space Partitions
$\log(1/\mu\epsilon)^{O(d)}$



img: computer.org

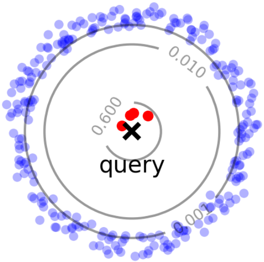
Hashing
$O(1/\sqrt{\mu}\epsilon^2)$

- Hashing-Based-Estimators [Charikar, **S**'17]

Similar idea:

- Locality Senstive Samplers [Spring, Shrivastava '17]

Random Sampling
$1/\mu\epsilon^2$



Slow in high dim

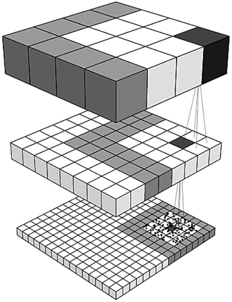
Sub-linear in $1/\mu$

Linear in $1/\mu$

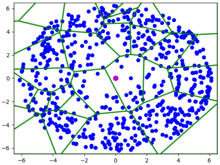
Methods for Fast Kernel Evaluation

$P \subset \mathbb{R}^d$, $\epsilon > 0 \Rightarrow (1 \pm \epsilon)$-approx to $\mu := \mathrm{KDF}_P(q)$ for any $q \in \mathbb{R}^d$

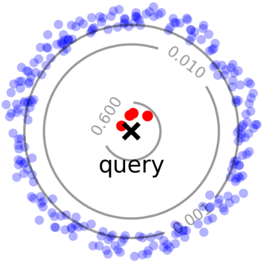
Space Partitions	Hashing	Random Sampling
$\log(1/\mu\epsilon)^{O(d)}$	$O(1/\sqrt{\mu}\epsilon^2)$	$1/\mu\epsilon^2$



img: computer.org



Importance Sampling via Randomized Space Partitions



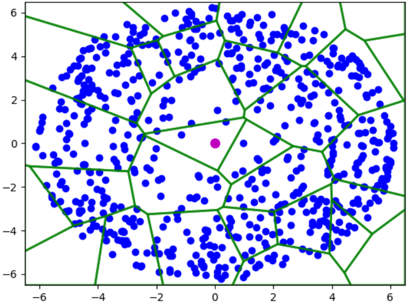
Slow in high dim

Sub-linear in $1/\mu$

Linear in $1/\mu$

Randomized Space Partitions

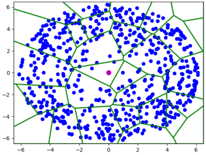
Distribution $\mathcal{H}$ over partitions $h : \mathbb{R}^d \to [M]$



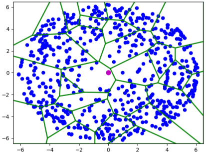
Randomized Space Partitions

Distribution $\mathcal{H}$ over partitions $h : \mathbb{R}^d \to [M]$

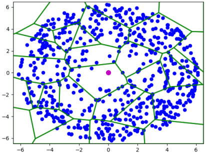
h_1



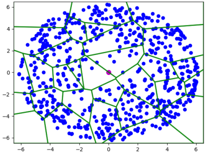
h_2



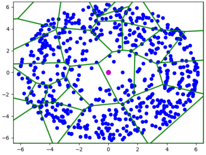
h_3



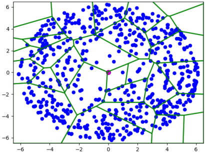
h_4



h_5



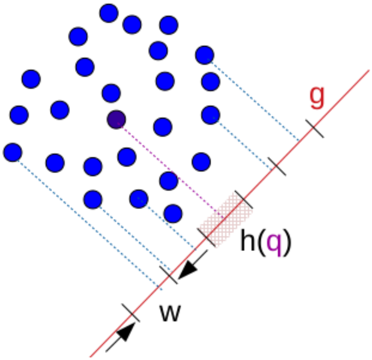
h_6



Locality Sensitive Hashing

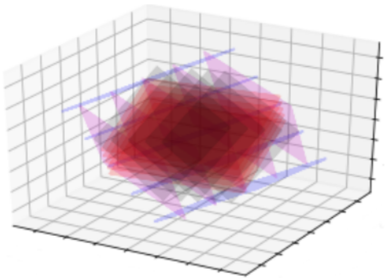
Partitions $\mathcal{H}$ such $\mathbb{P}_{h \sim \mathcal{H}}[h(x) = h(y)] = p(\|x - y\|)$

Euclidean LSH [Datar, Immorlika, Indyk, Mirrokni'04]



Concatenate k hashes

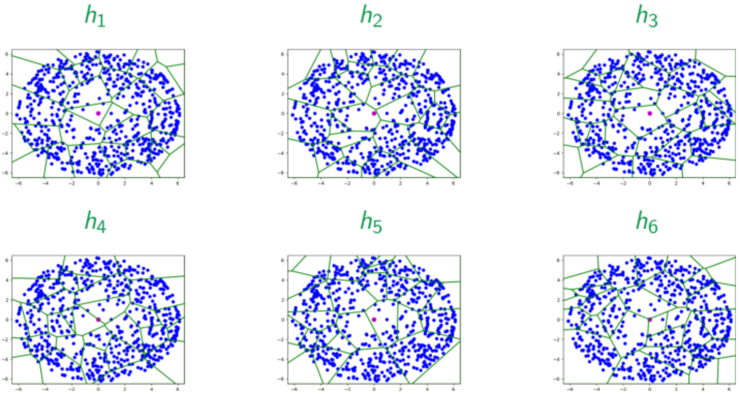
$p^k(\|x - y\|)$



Hashing-Based-Estimators

[Charikar, **S.** FOCS'17]

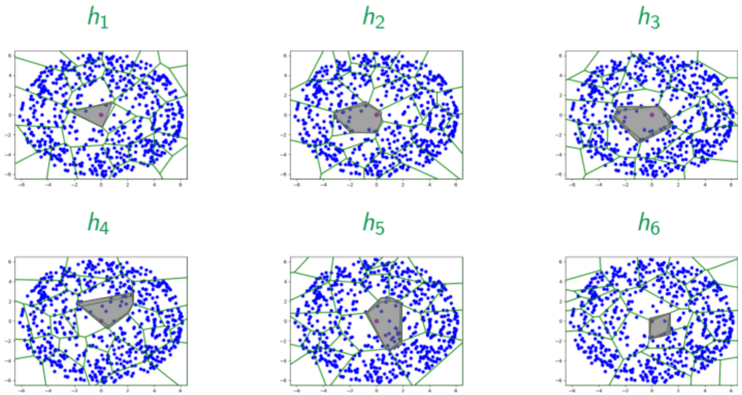
- **Preprocess:** Sample $h_1, \ldots, h_m \sim \mathcal{H}$ and evaluate on P
- **Query:** $H_t(q)$ hash-bucket for q in table t



Hashing-Based-Estimators

[Charikar, **S.** FOCS'17]

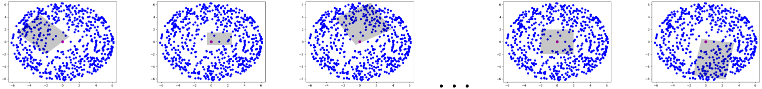
- **Preprocess:** Sample $h_1, \ldots, h_m \sim \mathcal{H}$ and evaluate on P
- **Query:** $H_t(q)$ hash-bucket for q in table t



Hashing-Based-Estimators

[Charikar, **S.** FOCS'17]

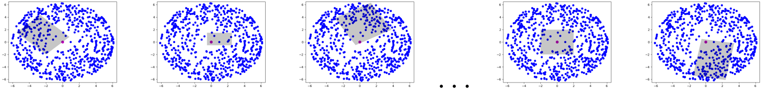
- **Preprocess:** Sample $h_1, \ldots, h_m \sim \mathcal{H}$ and evaluate on P
- **Query:** $H_t(q)$ hash-bucket for q in table t



Hashing-Based-Estimators

[Charikar, **S.** FOCS'17]

- **Preprocess:** Sample $h_1, \ldots, h_m \sim \mathcal{H}$ and evaluate on P
- **Query:** $H_t(q)$ hash-bucket for q in table t



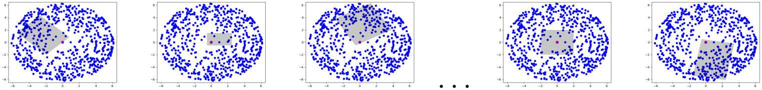
- **Estimator:** Sample random point X_t from $H_t(q)$ and return:

$$Z_m = \frac{1}{m}\sum_{t=1}^{m}\frac{1}{n}\frac{k(X_t, q)}{p(X_t, q)/|H_t(q)|}$$

Hashing-Based-Estimators

[Charikar, **S.** FOCS'17]

- **Preprocess:** Sample $h_1, \ldots, h_m \sim \mathcal{H}$ and evaluate on P
- **Query:** $H_t(q)$ hash-bucket for q in table t



- **Estimator:** Sample random point X_t from $H_t(q)$ and return:

$$Z_m = \frac{1}{m}\sum_{t=1}^{m}\frac{1}{n}\frac{k(X_t, q)}{p(X_t, q)/|H_t(q)|}$$

How many samples m? which LSH?

Hashing-Based-Estimators have Practical Limitations

Theorem [Charikar, S. FOCS'17]

For certain kernels HBE solves the kernel evaluation problem for $\mu \geq \tau$ using $O(1/\sqrt{\mu}\epsilon^2)$ samples and $O(n/\sqrt{\tau}\epsilon^2)$ space.

Kernel	LSH	Overhead
$e^{-\|\mathbf{x}-y\|^2}$	Ball Carving [Andoni, Indyk'06]	$e^{\tilde{O}(\log^{\frac{2}{3}}(n))}$
$e^{-\|\mathbf{x}-y\|}$	Euclidean [Datar et al'04]	$\sqrt{e}$
$\frac{1}{1+\|\mathbf{x}-y\|_2^t}$	Euclidean [Datar et al'04]	$3^{t/2}$

Hashing-Based-Estimators have Practical Limitations

Theorem [Charikar, S. FOCS'17]

For certain kernels HBE solves the kernel evaluation problem for $\mu \geq \tau$ using $O(1/\sqrt{\mu}\epsilon^2)$ samples and $O(n/\sqrt{\tau}\epsilon^2)$ space.

Practical Limitations:

1. Super-linear Space $\Rightarrow$ Not practical for massive datasets
2. Uses Adaptive procedure to estim. number of samples: $\Rightarrow$ large-constant + stringent requirements on hash functions.
3. Gaussian kernel Ball-Carving LSH very slow $e^{\tilde{O}(\log^{\frac{2}{3}}(n))}$

Hashing-Based-Estimators have Practical Limitations

Theorem [Charikar, S. FOCS'17]

For certain kernels HBE solves the kernel evaluation problem for $\mu \geq \tau$ using $O(1/\sqrt{\mu}\epsilon^2)$ samples and $O(n/\sqrt{\tau}\epsilon^2)$ space.

Practical Limitations:

1. Super-linear Space $\Rightarrow$ Not practical for massive datasets
2. Uses Adaptive procedure to estim. number of samples:
$\Rightarrow$ large-constant + stringent requirements on hash functions.
3. Gaussian kernel Ball-Carving LSH very slow $e^{\tilde{O}(\log^{\frac{2}{3}}(n))}$

Hashing-Based-Estimators have Practical Limitations

Theorem [Charikar, S. FOCS'17]

For certain kernels HBE solves the kernel evaluation problem for $\mu \geq \tau$ using $O(1/\sqrt{\mu}\epsilon^2)$ samples and $O(n/\sqrt{\tau}\epsilon^2)$ space.

Practical Limitations:

1. Super-linear Space $\Rightarrow$ Not practical for massive datasets
2. Uses Adaptive procedure to estim. number of samples:
   $\Rightarrow$ large-constant + stringent requirements on hash functions.
3. Gaussian kernel Ball-Carving LSH very slow $e^{\tilde{O}(\log^{\frac{2}{3}}(n))}$

Hashing-Based-Estimators have Practical Limitations

Theorem [Charikar, S. FOCS'17]

For certain kernels HBE solves the kernel evaluation problem for $\mu \geq \tau$ using $O(1/\sqrt{\mu}\epsilon^2)$ samples and $O(n/\sqrt{\tau}\epsilon^2)$ space.

Practical Limitations:

1. Super-linear Space $\Rightarrow$ Not practical for massive datasets
2. Uses Adaptive procedure to estim. number of samples:
   $\Rightarrow$ large-constant + stringent requirements on hash functions.
3. Gaussian kernel Ball-Carving LSH very slow $e^{\tilde{O}(\log^{\frac{2}{3}}(n))}$

Q: Practical HBE + preserve theoretical **guarantees**?

Overcoming practical Limitations of HBE

[Charikar, **S.** FOCS'17]

Practical Limitations:

1. super-linear space!
2. Adaptive procedure has large constant overhead.
3. Gaussian Kernel Ball-Carving LSH is slow.

[This work ICML'19]

Resolve by:

1. Sketching (sub-linear space)
2. Improved Adaptive procedure + New Analysis
3. Practical HBE for Gaussian Kernel via Eulcidean LSH

Overcoming practical Limitations of HBE

[Charikar, **S.** FOCS'17]

Practical Limitations:

1. super-linear space!
2. Adaptive procedure has large constant overhead.
3. Gaussian Kernel Ball-Carving LSH is slow.

[This work ICML'19]

Resolve by:

1. Sketching (sub-linear space)
2. Improved Adaptive procedure + New Analysis
3. Practical HBE for Gaussian Kernel via Eulcidean LSH

Overcoming practical Limitations of HBE

[Charikar, **S.** FOCS'17]

Practical Limitations:

1. super-linear space!
2. Adaptive procedure has large constant overhead.
3. Gaussian Kernel Ball-Carving LSH is slow.

[This work ICML'19]

Resolve by:

1. Sketching (sub-linear space)
2. Improved Adaptive procedure + New Analysis
3. Practical HBE for Gaussian Kernel via Eulcidean LSH

Overcoming practical Limitations of HBE

[Charikar, **S.** FOCS'17]

Practical Limitations:

1. super-linear space!
2. Adaptive procedure has large constant overhead.
3. Gaussian Kernel Ball-Carving LSH is slow.

[This work ICML'19]

Resolve by:

1. Sketching (sub-linear space)
2. Improved Adaptive procedure + New Analysis
3. Practical HBE for Gaussian Kernel via Eulcidean LSH

Overcoming practical Limitations of HBE

[Charikar, **S.** FOCS'17]

Practical Limitations:

1. super-linear space!
2. Adaptive procedure has large constant overhead.
3. Gaussian Kernel Ball-Carving LSH is slow.

[This work ICML'19]

Resolve by:

1. Sketching (sub-linear space)
2. Improved Adaptive procedure + New Analysis
3. Practical HBE for Gaussian Kernel via Eulcidean LSH

[S.*, Rong*, Bailis, Charikar, Levis ICML'19]

First Practical and **Provably** Accurate Algorithm for Gaussian Kernel in High Dimensions

Going back a step

Q1: Practical HBE + preserve theoretical **guarantees**?

Going back a step

Q1: Practical HBE + preserve theoretical **guarantees**?

Yes: Sketching, Adaptive procedure, Euclidean LSH

Going back a step

Q1: Practical HBE + preserve theoretical **guarantees**?

Yes: Sketching, Adaptive procedure, Euclidean LSH

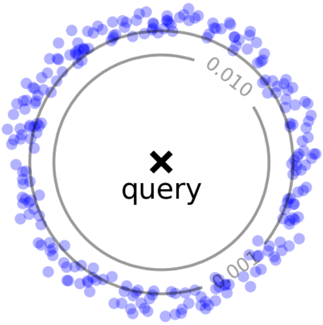
Q2: Is it always better to use?

Worst-case bounds can be misleading

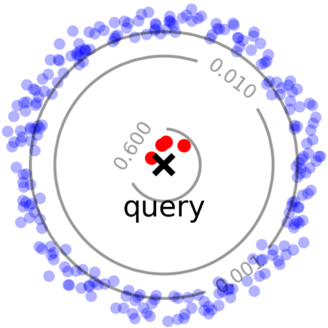
Worst-case bounds do not always reflect reality

Random Sampling

good: O(1) samples



bad: $O(1/\mu)$ samples



Going back a step

Q1: Practical HBE + preserve theoretical **guarantees**?

Yes: Sketching, Adaptive procedure, Euclidean LSH

Q2: Is it always better to use?

Going back a step

Q1: Practical HBE + preserve theoretical **guarantees**?

Yes: Sketching, Adaptive procedure, Euclidean LSH

Q2: Is it always better to use?

No: worst-case insufficient to predict performance on a **dataset**.

Going back a step

Q1: Practical HBE + preserve theoretical **guarantees**?

Yes: Sketching, Adaptive procedure, Euclidean LSH

Q2: Is it always better to use?

No: worst-case insufficient to predict performance on a **dataset**.

[This work ICML'19]
Diagnostic tools to estimate dataset-specific performance even without evaluating HBE

Outline of the rest of the talk

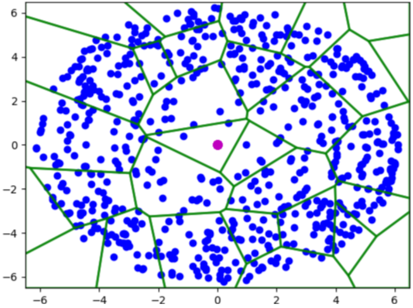
1. Sketching
2. Diagnostic tools
3. Experimental evaluation

Sketching

How to sketch the KDF?

Recall: HBE samples a single point from each hash table.

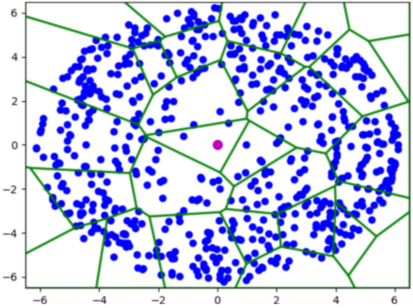
Goal: "simulate" HBE on full sample by applying on "Sketch"



How to sketch the KDF?

Recall: HBE samples a single point from each hash table.

Goal: "simulate" HBE on full sample by applying on "Sketch"



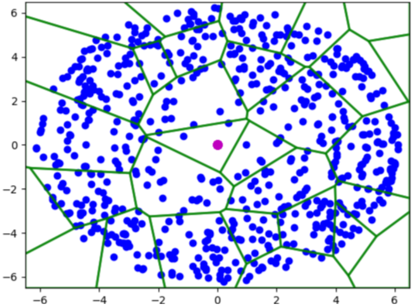
Two approaches:

1. Random points:
   $\Rightarrow$ some buckets might have 0 points in sketch.
2. point from each bucket:
   $\Rightarrow$ might need a large number of points

How to sketch the KDF?

Recall: HBE samples a single point from each hash table.

Goal: "simulate" HBE on full sample by applying on "Sketch"



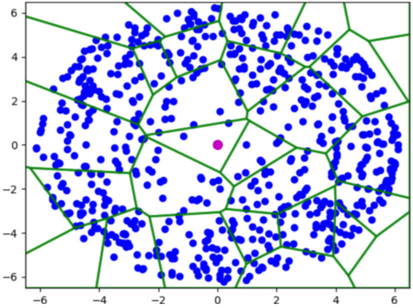
Two approaches:

1. Random points:
   $\Rightarrow$ some buckets might have 0 points in sketch.
2. point from each bucket:
   $\Rightarrow$ might need a large number of points

How to sketch the KDF?

Recall: HBE samples a single point from each hash table.

Goal: "simulate" HBE on full sample by applying on "Sketch"



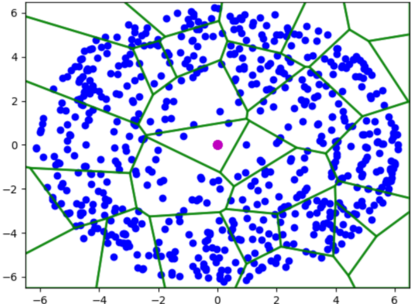
Two approaches:

1. Random points:
   ⇒ some buckets might have 0 points in sketch.
2. point from each bucket:
   ⇒ might need a large number of points

How to sketch the KDF?

Recall: HBE samples a single point from each hash table.

Goal: "simulate" HBE on full sample by applying on "Sketch"



Two approaches:

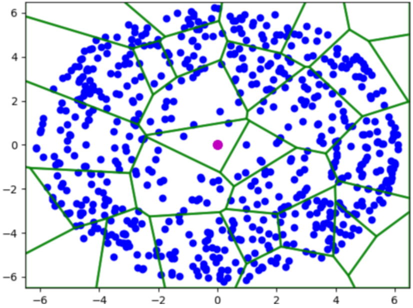
1. Random points:
   $\Rightarrow$ some buckets might have 0 points in sketch.
2. point from each bucket:
   $\Rightarrow$ might need a large number of points

Idea: interpolate between uniform points vs uniform over buckets!

How to sketch the KDF?

Recall: HBE samples a single point from each hash table.

Goal: "simulate" HBE on full sample by applying on "Sketch"



Two approaches:

1. Random points:
   ⇒ some buckets might have 0 points in sketch.
2. point from each bucket:
   ⇒ might need a large number of points

Idea: interpolate between uniform points vs uniform over buckets!

Solution: hashing+ non-uniform sampling

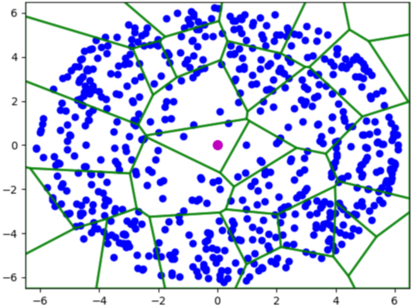
Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

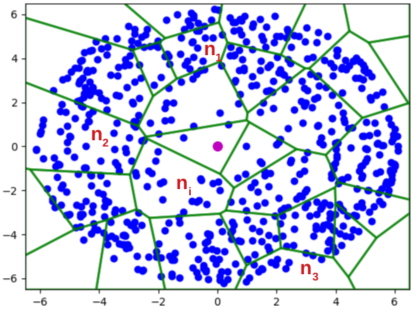
Sample h_0 evaluate on P



Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



$S \leftarrow \emptyset$.

for $j = 1, \ldots,$ SketchSize:

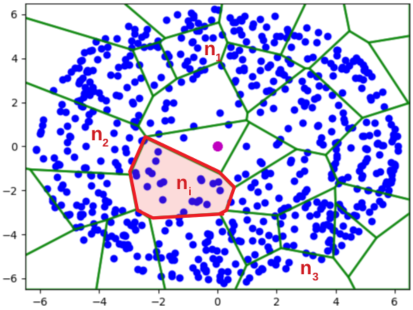
- Sample bucket i prob. $\propto n_i{}^{\gamma}$
- Sample a random point J from bucket i: $S \leftarrow S \cup \{J\}$
- Weight it so that $\mathbb{E}_J[\hat{w}_J k(q, x_J)] \propto \mathrm{KDF}_P(q)$

return $(\hat{w}, S)$

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



$S \leftarrow \emptyset$.

for $j = 1, \ldots,$ SketchSize:

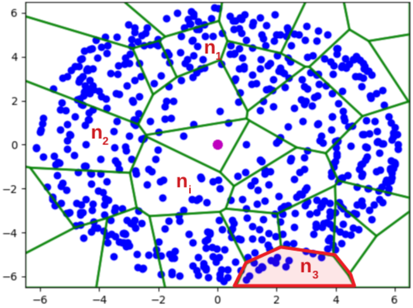
- Sample bucket i prob. $\propto {n_i}^{\gamma}$
- Sample a random point J from bucket i: $S \leftarrow S \cup \{J\}$
- Weight it so that $\mathbb{E}_J[\hat{w}_J k(q, x_J)] \propto \mathrm{KDF}_P(q)$

return $(\hat{w}, S)$

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



$S \leftarrow \emptyset$.

for $j = 1, \ldots,$ SKETCHSIZE:

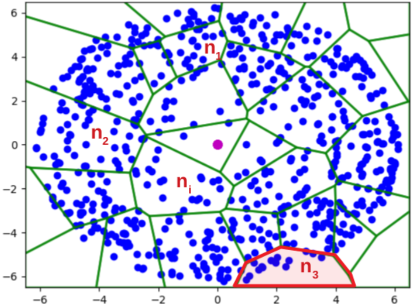
- Sample bucket i prob. $\propto {n_i}^{\gamma}$
- Sample a random point J from bucket i: $S \leftarrow S \cup \{J\}$
- Weight it so that $\mathbb{E}_J[\hat{w}_J k(q, x_J)] \propto \mathrm{KDF}_P(q)$

return $(\hat{w}, S)$

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



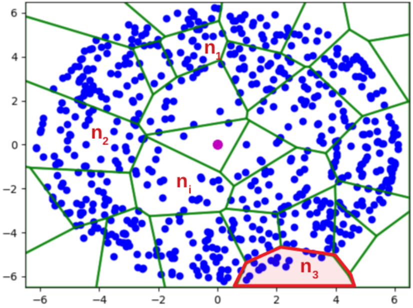
Theorem: $O(1/\tau)$ points suffice.

- Approx. any density $\mu \geq \tau$.
- Reduce space from $O(n/\sqrt{\tau})$ to $O(1/\sqrt{\tau^3})$
- Contains a point from any bucket with $\geq n \cdot \tau$ points

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



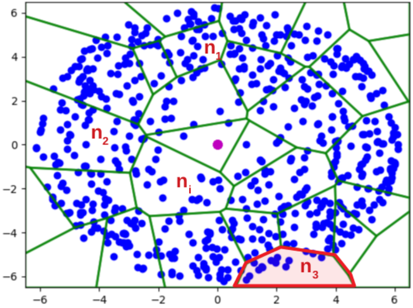
Theorem: $O(1/\tau)$ points suffice.

- Approx. any density $\mu \geq \tau$.
- Reduce space from $O(n/\sqrt{\tau})$ to $O(1/\sqrt{\tau^3})$
- Contains a point from any bucket with $\geq n \cdot \tau$ points

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



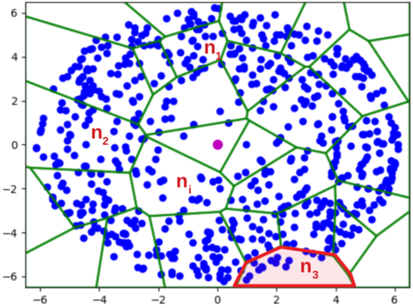
Theorem: $O(1/\tau)$ points suffice.

- Approx. any density $\mu \geq \tau$.
- Reduce space from $O(n/\sqrt{\tau})$ to $O(1/\sqrt{\tau^3})$
- Contains a point from any bucket with $\geq n \cdot \tau$ points

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



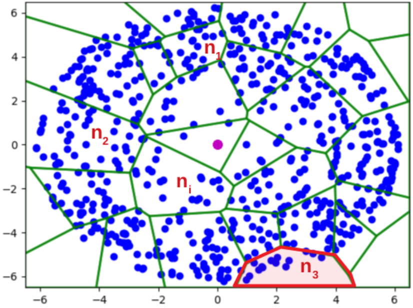
Theorem: $O(1/\tau)$ points suffice.

- Approx. any density $\mu \geq \tau$.
- Reduce space from $O(n/\sqrt{\tau})$ to $O(1/\sqrt{\tau^3})$
- Contains a point from any bucket with $\geq n \cdot \tau$ points

Sketching Kernel Density Function

Hashing-Based-Sketch (HBS): hashing + non-uniform sampling.

Sample h_0 evaluate on P



Theorem: $O(1/\tau)$ points suffice.

- Approx. any density $\mu \geq \tau$.
- Reduce space from $O(n/\sqrt{\tau})$ to $O(1/\sqrt{\tau^3})$
- Contains a point from any bucket with $\geq n \cdot \tau$ points

Sub-linear space: e.g $\tau = \frac{1}{\sqrt{n}}$ we get $n^{5/4} \rightarrow n^{3/4}$

Diagnostic tools

Variance of Unbiased Estimators

Unbiased estimators: Random Sampling, HBE

Metric of interest is average relative variance:

$$\mathbb{E}_{q\sim P}\left[\frac{\mathbb{V}[Z(q)]}{\mathbb{E}[Z(q)]^2}\right] \propto \text{“Sample Complexity”}$$

Diagnostic Procedure

1. Sample a number T of random queries from P.
2. For each $\Rightarrow$ **upper bound** Relative Variance
3. Average for each method of interest over T queries.

Estimate mean and bound Variance

Bounding the variance

Variance is a "quadratic polynomial" of $w_i = k(q, x_i)$

$$\mathbb{V}[Z] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

Bounding the variance

Variance is a "quadratic polynomial" of $w_i = k(q, x_i)$

$$\mathbb{V}[Z] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

Random Sampling (RS)

$$\mathbb{E}[k^2(q, X)] = \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2$$

$$V_{ij} = 1$$

Bounding the variance

Variance is a "quadratic polynomial" of $w_i = k(q, x_i)$

$$\mathbb{V}[Z] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

Random Sampling (RS)

$$\mathbb{E}[k^2(q, X)] = \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2$$

$$V_{ij} = 1$$

HBE collision prob. $p(x, y)$

$$\mathbb{E}[Z^2] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

$$V_{ij} = \frac{\min\{p(q,x_i), p(q,x_j)\}}{p^2(q,x_i)}$$

Bounding the variance

Variance is a "quadratic polynomial" of $w_i = k(q, x_i)$

$$\mathbb{V}[Z] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

Random Sampling (RS)

$$\mathbb{E}[k^2(q, X)] = \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2$$

$$V_{ij} = 1$$

HBE collision prob. $p(x, y)$

$$\mathbb{E}[Z^2] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

$$V_{ij} = \frac{\min\{p(q,x_i), p(q,x_j)\}}{p^2(q,x_i)}$$

Evaluating variance naively requires $O(n)$ or $O(n^2)$ per query

Bounding the variance

Variance is a "quadratic polynomial" of $w_i = k(q, x_i)$

$$\mathbb{V}[Z] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

Random Sampling (RS)

$$\mathbb{E}[k^2(q, X)] = \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2$$

$$V_{ij} = 1$$

HBE collision prob. $p(x, y)$

$$\mathbb{E}[Z^2] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

$$V_{ij} = \frac{\min\{p(q,x_i), p(q,x_j)\}}{p^2(q,x_i)}$$

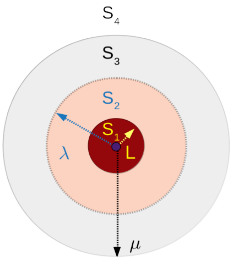
Q: Efficient alternative?

Data-dependent Variance Bounds

Variance is a "quadratic polynomial" of $w_i = k(q, x_i)$

$$\mathbb{V}[Z] \leq \frac{1}{n^2} \sum_{i,j=1}^{n} w_i^2 V_{ij}$$

Decompose in 4 sets



$\rightarrow$ For two sets $S_\ell, S_{\ell'}$:

$$\sum_{i \in S_\ell, j \in S_{\ell'}} w_i^2 V_{ij}$$

$$\leq \sup_{i \in S_\ell, j \in S_{\ell'}} \left\{ \frac{w_i}{w_j} V_{ij} \right\} \mu_\ell \mu_{\ell'}$$

(Hölder)

$\rightarrow$ **Diagnostic**

1. bnd $\binom{4}{2}$ terms
2. Evaluate on subsample S_0
3. Produced by RS and *Adapt. Algorithm*

Evaluation

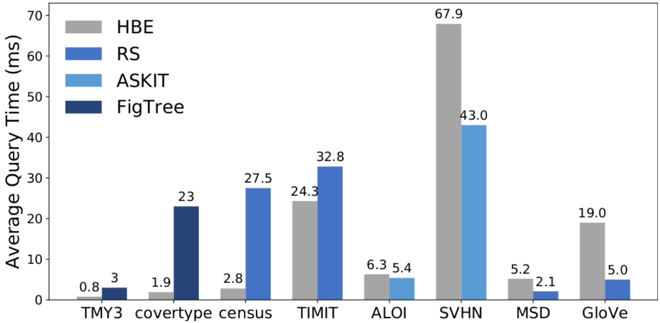
Algorithms for Kernel Evaluation

- Random Sampling (RS):
  sensitive to ***range*** *of kernel values (distances).*
- Hashing-Based-Estimators (HBE):
  *sensitive to "****correlations****" (dense distant clusters)*
  [Charikar, S. FOCS'2017][This work ICML'2019]
- Fast Improved Gauss Transform (FIGTree):
  *sensitive to # "clusters" (****directions****) at certain distance*
  [Morariu,Srinivasan,Raykar, Duraiswami, Davis, NeurIPS'2009]
- Approximate Skeletonization via Treecodes (ASKIT)
  sensitive to **"medium" distance scale***/size clusters*
  [March, Xiao, Biros, SIAM JSC 2015]

Compare performance on Real-world datasets

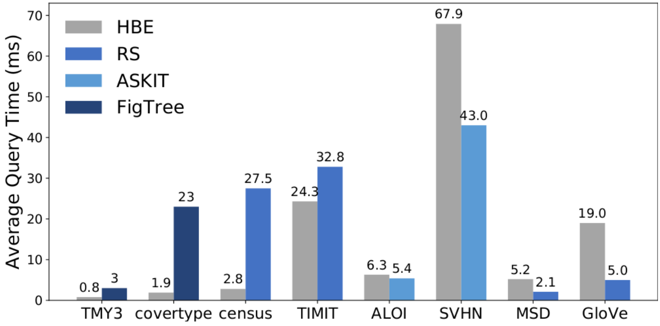
Comparison on Real-world Datasets

HBE is consistently best or second-best method



Comparison on Real-world Datasets

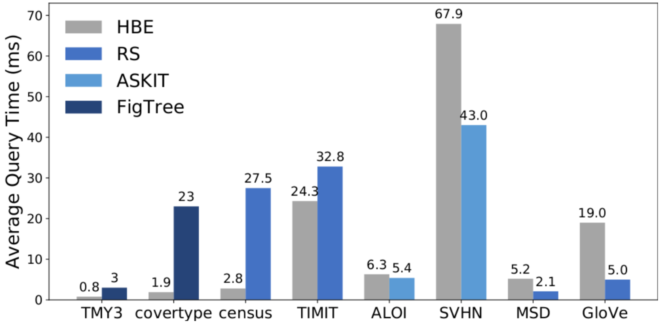
HBE is consistently best or second-best method

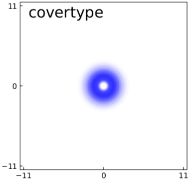


Diagnostic correctly (21/22) choses between RS and HBE

Comparison on Real-world Datasets

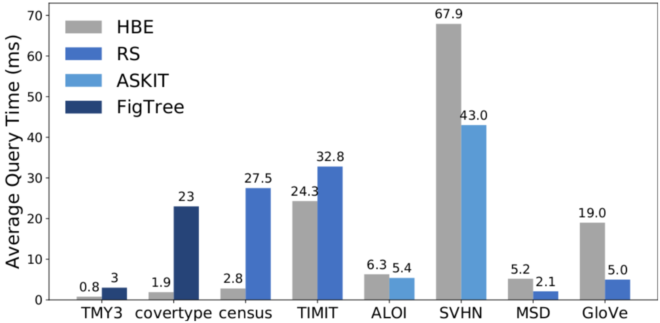
HBE is consistently best or second-best method

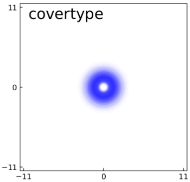


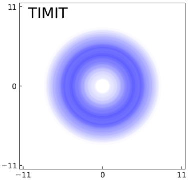


Comparison on Real-world Datasets

HBE is consistently best or second-best method

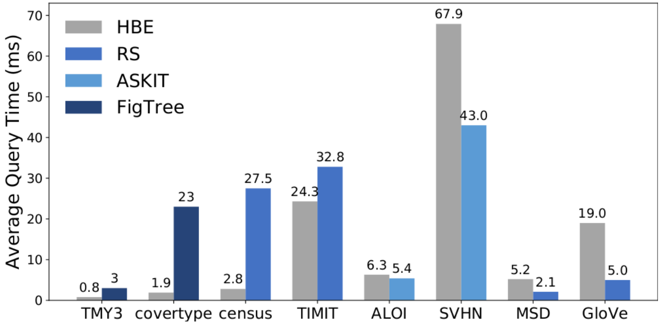


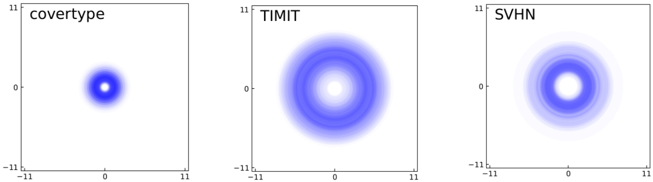




Comparison on Real-world Datasets

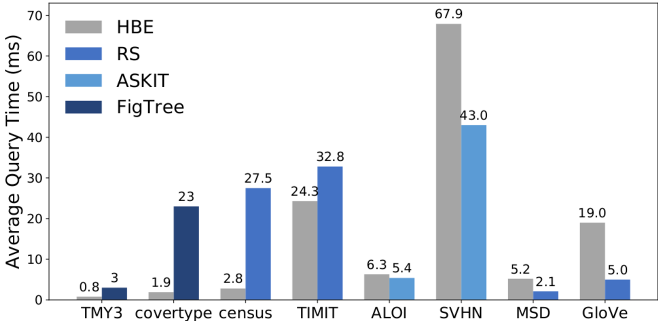
HBE is consistently best or second-best method

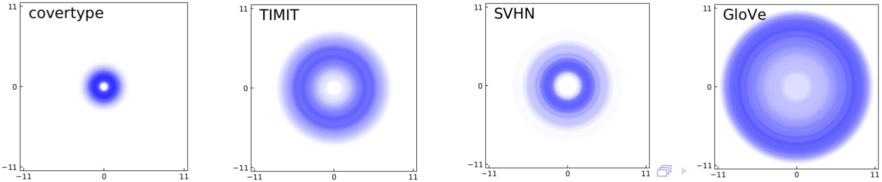




Comparison on Real-world Datasets

HBE is consistently best or second-best method





Benchmark Instances

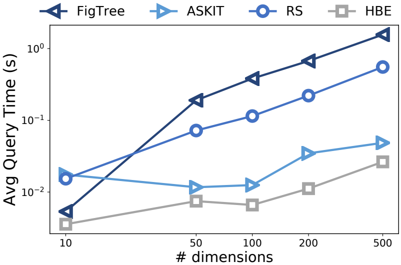
Synthetic Benchmarks:

1. Worst-case: no *single* **geometric aspect** can be exploited!
2. D-clusters: gauge impact of different geometric aspects.

Worst-case Instances

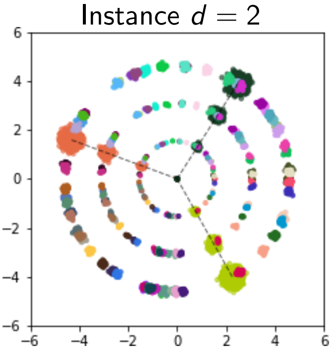
Union of highly-clustered with uncorrelated points

(fixed $\mu = 10^{-3}$, dimension $d \in [10, 500]$, 100K queries)



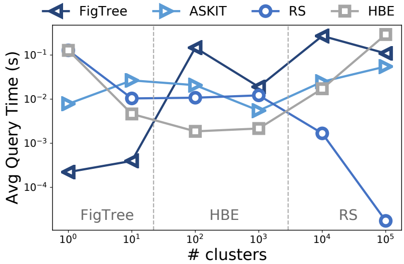
"Worst-case" data sets

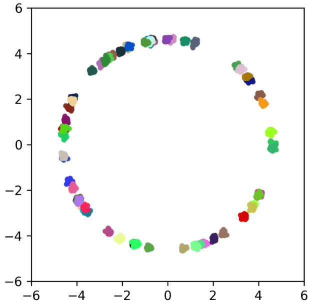
- HBE best
- ASKIT second best



Instances with D clusters

Fix $N = n \cdot D = 500K$, vary $D \in [1, 10^5]$





D-structured datasets:

- $D \ll \sqrt{N}$: space partitions
- $D \sim N^{1-\delta}$: Random Samp.
- $1 \ll D \ll N$: HBE

Conclusion

Rehashing Kernel Evaluation in High Dimensions

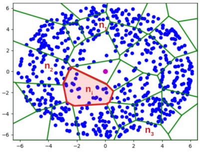
Hashing-Based-Estimators:

1. made practical + often state-of-the-art + worst-case guarant.
2. data-dependent diagnostics: when to use & how to tune

"Rehashing" methodology

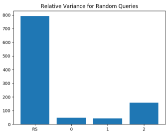
Open Source Implementation and Experiments

(https://github.com/kexinrong/rehashing)



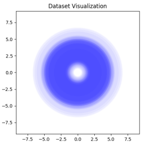
Sketch

→



Diagnostics

→



Visualization

→ Config file (deployment)

Thank you!

psimin@stanford.edu