# Rehashing Kernel Evaluation in High Dimensions

**Paris Siminelakis** [*1] **Kexin Rong** [*1] **Peter Bailis** [1] **Moses Charikar** [1] **Philip Levis** [1]

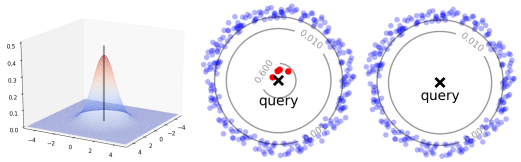
## Abstract

Kernel methods are effective but do not scale well to large scale data, especially in high dimensions where the geometric data structures used to accelerate kernel evaluation suffer from the curse of dimensionality. Recent theoretical advances have proposed fast kernel evaluation algorithms leveraging hashing techniques with worst-case asymptotic improvements. However, these advances are largely confined to the theoretical realm due to concerns such as super-linear preprocessing time and diminishing gains in non-worst case datasets. In this paper, we close the gap between theory and practice by addressing these challenges via *provable and practical* procedures for adaptive sample size selection, preprocessing time reduction, and refined variance bounds that quantify the data-dependent performance of random sampling and hashing-based kernel evaluation methods. Our experiments show that these new tools offer up to $10\times$ improvement in evaluation time on a range of synthetic and real-world datasets.

## 1. Introduction

Kernel methods are a class of non-parametric methods used for a variety of tasks including density estimation, regression, clustering and distribution testing (Muandet et al., 2017). They have a long and influential history in statistical learning (Schölkopf et al., 2002) and scientific computing (Buhmann, 2003). However, the scalability challenges during both training and inference limit their applicability to large scale high-dimensional datasets: a larger training set improves accuracy but incurs a quadratic increase in overall evaluation time.

In this paper, we focus on the problem of *fast approximate evaluation* of the Kernel Density Estimate. Given a dataset $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, a kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, and a vector of (non-negative) weights $u \in \mathbb{R}^n$, the weighted *Kernel Density Estimate* (KDE) at $q \in \mathbb{R}^d$ is given by $\mathrm{KDE}_P^u(q) := \sum_{i=1}^n u_i k(q, x_i)$. Our goal is to, after some preprocessing, efficiently estimate the KDE at a query point with $(1 \pm \epsilon)$ multiplicative accuracy under a small failure probability $\delta$. This problem is well-studied (Greengard & Rokhlin, 1987; Gray & Moore, 2003; March et al., 2015), but in high dimensions only recently novel importance sampling algorithms, called *Hashing-Based-Estimators (HBE)*, demonstrate provable improvements over random sampling (RS) under worst-case assumptions (Charikar & Siminelakis, 2017).




(a) kernel (b) difficult case (c) simple case

*Figure 1.* (a) For radially decreasing kernels (e.g. Gaussian), points close to the query have higher kernel values than those that are far. (b) Random sampling does not perform well when a small number of close points contribute significantly to the query density. (c) Random sampling performs well when most points have similar kernel values (distance from query).

HBE at its core uses a hash function $h$ (randomized space partition) to construct for any $q$ an unbiased estimator of $\mu := \mathrm{KDE}_P^u(q)$[1]. (Charikar & Siminelakis, 2017) showed that given a hash function with evaluation time $T$ and collision probability $\mathbb{P}[h(x) = h(y)] = \Theta(\sqrt{k(x, y)})$, one can get a $(1 \pm \epsilon)$ approximation to $\mu \geq \tau$ in time $\tilde{O}(T/\epsilon^2\sqrt{\mu})$ and space $\tilde{O}(n/\epsilon^2\sqrt{\tau})$, improving over random sampling that requires time $O(1/\epsilon^2\mu)$ in the worst case.

HBE improves upon RS by being able to better sample points with larger weights (kernel values). In particular, RS does not perform well (Figure 1(b)) when a significant portion of the density comes from a few points with large weights (close to the query). HBE samples uniformly from a set of biased samples (hash buckets where the query is mapped to) that has a higher probability of containing high-

[*]Equal contribution [1]Stanford University, Stanford, California, US. Correspondence to: Paris Siminelakis <psimin@stanford.edu>, Kexin Rong <krong@stanford.edu>.

*Proceedings of the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

[1]The value $\mu \in (0, 1]$ can be seen as a *margin* for the decision surface $\sum_{i=1}^n u_i k(q, x_i) \geq 0$.

weight points. For radially decreasing kernels (Figure 1(a)), the biased sample can be produced via Locality Sensitive Hashing (LSH) (Indyk & Motwani, 1998). To obtain $m$ such biased samples for estimating the query density, the scheme requires building $m$ independent hash tables on the entire dataset. The runtime mentioned above is achieved by setting $m = O(1/\epsilon^2\sqrt{\mu})$. In practice, one cares about values of $\mu \geq 1/\sqrt{n}$, which is a lower bound on the order of statistical fluctuations when $P$ consists of $n$ i.i.d samples from some smooth distribution (Sriperumbudur et al., 2016).

**Limitations.** Despite progress on the worst case query time, the idea of using hashing for kernel evaluation, introduced independently in (Spring & Shrivastava, 2017) and (Charikar & Siminelakis, 2017), has largely been confined to the theoretical realm due to a few practical concerns. First, straightforward implementations of the proposed approach require a large amount (e.g. $\tilde{O}(n^{\frac{5}{4}})$ for $\tau = 1/\sqrt{n}$) of preprocessing time and memory to create the requisite number of hash tables. Second, RS can outperform HBE on certain datasets (Figure 1(c)), a fact that is not captured by the worst-case theoretical bounds. Third, to implement this approach (Charikar & Siminelakis, 2017) use an adaptive procedure to estimate the sufficient sample size $m$. For this procedure to work, the estimators need to satisfy some uniform polynomial decay of the variance as a function of $\mu$, that for the popular Gaussian kernel, only the hashing scheme of (Andoni & Indyk, 2006) (with an significant $\exp(O(\log^{\frac{2}{3}} n))$ runtime and memory overhead per hash table) was shown to satisfy. These issues call the applicability of HBE into question and to the best of our knowledge, the method has not been shown before to empirically improve upon competing methods.

### 1.1. Contributions

In this paper, we close the gap between theory and practice by making the following contributions:

1. **Overhead reduction via sketching.** HBE requires super-linear preprocessing time and memory, a result of having to hash the entire dataset once for each sample. To reduce this overhead, we develop new theoretical results on sketching the KDE by a weighted subset of points using hashing and non-uniform sampling (Section 6). Our *Hashing-Based-Sketch (HBS)* is able to better sample sparse regions of space, while still having variance close to that of a random sketch, leading to better performance on low-density points. Applying HBE on top of the sketch results in considerable gains while maintaining accuracy.

2. **Data-dependent diagnostics.** Despite worst-case theoretical guarantees, RS outperforms HBE on certain datasets in practice. To quantify this phenomenon, we introduce an inequality that bounds the variance of unbiased estimators that include HBE and RS (Section 4). We propose a diagnostic procedure utilizing the variance bounds that, for a given dataset, chooses at runtime with minimal overhead the better of the two approaches, and does so *without invoking HBE*; our evaluation shows that the procedure is both accurate and efficient. The new variance bound also allows us to simplify and recover theoretical results of (Charikar & Siminelakis, 2017).

3. **A practical algorithm for the Gaussian kernel.** We design a simplified version of the adaptive procedure of (Charikar & Siminelakis, 2017), that estimates the sufficient sample size $m$ in parallel with estimating the density $\mu$, and provide an improved analysis that results in an order of magnitude speedup in the query time (Section 5). More importantly, our new analysis shows that slightly weaker conditions (non-uniform but bounded polynomial decay) on the variance of the estimators are sufficient for the procedure to work. By proving that HBE based on the hashing scheme of (Datar et al., 2004) satisfy the condition, we give the first *practical algorithm* that provably improves upon RS for the Gaussian kernel in high dimensions. Previously, it was not known how to use this hashing scheme within an adaptive procedure for this purpose.

We perform extensive experimental evaluations of our methods on a variety of synthetic benchmarks as well as real-world datasets. Together, our evaluation against other state-of-the-art competing methods on kernel evaluation shows:

- HBE outperforms all competing methods for synthetic "worst-case" instances with multi-scale structure and dimensions $d \geq 10$, as well as for "structured" instances with a moderate number of clusters (Section 7.1).
- For real-world datasets, HBE is always competitive with alternative methods and is faster for many datasets by up to $\times 10$ over the next best method (Section 7.2).

In summary, our theoretical and experimental results constitute an important step toward making Hashing-Based-Estimators practical and in turn improving the scalability of kernel evaluation in large high-dimensional data sets.

## 2. Related Work

**Fast Multipole Methods and Dual Tree Algorithms.** Historically, the problem of KDE evaluation was first studied in low dimensions ($d \leq 3$) in the scientific computing literature, resulting in the Fast Multipole Method (FMM) (Greengard & Rokhlin, 1987). This influential line of work is based on a hierarchical spatial decomposition and runs in $\tilde{O}(2^d)$ time per evaluation point. Motivated by applications in statistical learning (Gray & Moore, 2001), the problem was revisited in 2000's and analogs of the Fast Multipole

Method (Gray & Moore, 2003), known as Dual Tree Algorithms (Lee et al., 2006), were proposed that aimed to capture latent underlying low-dimensional structure (Ram et al., 2009). Unfortunately, when such low-dimensional structure is absent, these methods, in general, have near-linear $O(n^{1-o(1)})$ runtime. Thus, under general assumptions, the only method that was known (Lee & Gray, 2009) to provably accelerate kernel evaluation in high dimensions was simple uniform random sampling (RS).

**Hashing-Based-Estimators.** The framework of HBE has recently been extended to apply to more general kernels (Charikar & Siminelakis, 2018) and hashing-based ideas have been used to design faster algorithms for "smooth" kernels (Backurs et al., 2018). Besides kernel evaluation, researchers have also applied hashing techniques to get practical improvements in outlier detection (Luo & Shrivastava, 2018a), gradient estimation (Chen et al., 2018), non-parametric clustering (Luo & Shrivastava, 2018b) and range-counting (Wu et al., 2018).

**Approximate Skeletonization.** ASKIT (March et al., 2015) represents a different line of work aimed at addressing the deficiencies of FMM and Dual-Tree methods. ASKIT also produces a hierarchical partition of points, but then uses linear algebraic techniques to approximate the contribution of points to each part via a combination of uniform samples and nearest neighbors. This makes the method robust to the dimension and mostly dependent on the number of points.

**Core-sets.** The problem of sketching the KDE *for all points* has been studied under the name of Core-sets or $\epsilon$-samples (Phillips, 2013). The methods are very effective in low dimensions (Zheng et al., 2013) $d \leq 3$, but become impractical to implement in higher dimensions for large data-sets due to their computational requirements $\Omega(n^2)$ (e.g. (Chen et al., 2010)). The recent paper (Phillips & Tai, 2018) presents an up-to-date summary.

## 3. Preliminaries

In this section, we present basic definitions and give a self contained presentation of Hashing-Based-Estimators that summarizes the parts of (Charikar & Siminelakis, 2017) upon which we build. All material beyond Section 3 is new.

**Notation.** For a set $S \subset [n]$ and numbers $u_1, \ldots, u_n$, let $u_S := \sum_{i \in S} u_i$. Let $\Delta_n := \{u \in \mathbb{R}^n_+ : \|u\|_1 = 1\}$ denote the $n$-dimensional simplex. Throughout the paper we assume that we want to approximate $\mathrm{KDE}^u_P$ with $u \in \Delta_n$. We can handle the general case $u \in \mathbb{R}^d$ by treating the positive and negative part of $u$ separately.

**Kernels.** All our theoretical results, unless explicitly stated, apply to general non-negative kernels. For concreteness and in our experiments, we focus on the Gaussian $\exp(-\|x-y\|^2/\sigma^2)$ and Laplace kernels $\exp(-\|x-y\|/\sigma)$ (Belkin et al., 2018), and typically suppress the dependence on the bandwidth $\sigma > 0$ (equivalently, we can rescale our points).

### 3.1. Multiplicative Approximation & Relative Variance

**Definition 1.** *A random variable $\hat{Z}$ is called an $(\epsilon, \delta)$-approximation to $\mu$ if $\mathbb{P}[|\hat{Z} - \mu| \geq \epsilon\mu] \leq \delta$.*

Given access to an unbiased estimator $\mathbb{E}[Z] = \mu$, our goal is to output an $(\epsilon, \delta)$-approximation to $\mu$. Towards that end the main quantity to control is the *relative variance*.

**Definition 2.** *For a non-negative random variable $Z$ we define the* relative variance *as* $\mathrm{RelVar}[Z] := \frac{\mathrm{Var}[Z]}{\mathbb{E}[Z]^2} \leq \frac{\mathbb{E}[Z^2]}{\mathbb{E}[Z]^2}$.

The relative variance captures the fluctuations of the random variable at the scale of the expectation. This is made precise in the following lemma that combines Chebyshev's and Paley-Zygmund inequality.

**Lemma 1.** *For a non-negative random variable $Z$ and parameters $t > 0$, $\theta \in [0, 1]$, we have:*

$$\mathbb{P}[Z \geq (t+1) \cdot \mathbb{E}[Z]] \leq \frac{1}{t^2} \cdot \mathrm{RelVar}[Z], \tag{1}$$

$$\mathbb{P}[Z > (1-\theta)\mathbb{E}[Z]] \geq \frac{1}{1 + \frac{1}{\theta^2} \cdot \mathrm{RelVar}[Z]}. \tag{2}$$

As $\mathrm{RelVar}[Z]$ decreases, the upper bound in (1) decreases while the lower bound in (2) increases, increasing our overall confidence of $Z$ being an accurate estimate of the mean. Thus, if one can construct a random variable $Z$ with $\mathbb{E}[Z] = \mu$ and small relative variance $\mathrm{RelVar}[Z] = O(\epsilon^2)$, Lemma 1 shows that $Z$ is an $(\epsilon, O(1))$-approximation to $\mu$. In fact, by Chernoff bounds, one can use the median-trick to boost the probability of success (Supplementary material).

### 3.2. Hashing-Based-Estimators

HBE uses hashing to create a data-structure that, after some preprocessing, can produce unbiased estimators for the KDE at query time (Figure 2) with low variance. Let $\mathcal{H}$ be a set of functions and $\nu$ a distribution over $\mathcal{H}$. We denote by $h \sim \mathcal{H}_\nu$ a random function $h \in \mathcal{H}$ sampled from $\nu$ and refer to $\mathcal{H}_\nu$ as a hashing scheme.

**Definition 3.** *Given a hashing scheme $\mathcal{H}_\nu$, we define the collision probability between two elements $x, y \in \mathbb{R}^d$ as $p(x,y) := \mathbb{P}_{h \sim \mathcal{H}_\nu}[h(x) = h(y)]$.*

**Pre-processing:** given dataset $P$ and hashing scheme $\mathcal{H}_\nu$:

- Sample $m$ hash functions $h_t \overset{i.i.d.}{\sim} \mathcal{H}_\nu$ for $t \in [m]$.
- Create hash tables $H_t := h_t(P)$ for $t \in [m]$ by evaluating the hash functions on $P$.

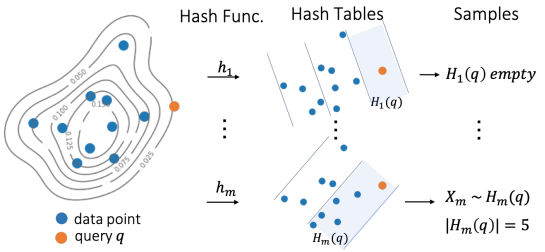



*Figure 2.* Given a dataset, the HBE approach samples a number of hash functions and populates a separate hash table for each hash function. At query time, for each hash table, we sample a point at random from the hash bucket that the query maps to.

The $m$ hash tables allow us to produce at most $m$ independent samples to estimate the KDE for each query.

**Query process:** query $q \in \mathbb{R}^d$, hash table index $t \in [m]$

- let $H_t(q) := \{i \in [n] : h_t(x_i) = h_t(q)\}$ denote the hash bucket (potentially empty) that $q$ maps to.
- If $H_t(q)$ is not empty return $Z_{h_t} := \frac{k(X_t,q)}{p(X_t,q)} u_{H_t(q)}$ where $X_t$ is sampled with probability proportional to $u_i$ from $H_t(q)$, otherwise return 0.

By averaging many samples produced by the hash tables we get accurate estimates. The salient properties of the estimators $Z_{h_t}$ are captured in the following lemma.

**Lemma 2.** *Assuming that* $\forall i \in [n], p(x_i, q) > 0$ *then*

$$\mathbb{E}[Z_h] = \sum_{i=1}^{n} u_i k(x, x_i), \tag{3}$$

$$\mathbb{E}[Z_h^2] = \sum_{i,j=1}^{n} k^2(q, x_i) \frac{u_i \mathbb{P}[i, j \in H(q)] u_j}{p^2(q, x_i)}. \tag{4}$$

As we see in (4), the second moment depends on the hashing scheme through the ratio $\frac{\mathbb{P}[i,j \in H(q)]}{p^2(q,x_i)}$. Thus, we hope to reduce the variance by selecting the hashing scheme appropriately. The difficulty lies in that the variance depends on the positions of points in the whole dataset. (Charikar & Siminelakis, 2017) introduced a property of HBE that allowed them to obtain provable bounds on the variance.

**Definition 4.** *Given a kernel $k$, a hashing scheme is called $(\beta, M)$-scale-free for $\beta \in [0, 1]$ and $M \geq 1$ if:* $\frac{1}{M} \cdot k(x, y)^\beta \leq p(x, y) \leq M \cdot k(x, y)^\beta$.

Thus, one needs to design hashing scheme that "adapts" to the kernel function. Next, we present a specific family of hashing schemes that can be used for kernel evaluation under the Exponential and Gaussian kernel.

### 3.3. HBE via Euclidean LSH

In the context of solving Approximate Nearest Neighbor search in Euclidean space, Datar et al. (Datar et al., 2004) introduced the following hashing scheme, *Euclidean LSH* (eLSH), that uses two parameters $w > 0$ (width) and $\kappa \geq 1$ (power). First, hash functions in the form of $h_i(x) := \lceil \frac{g_i^\top x}{w} + b_i \rceil$ map a $d$ dimensional vector $x$ into a set of integers ("buckets") by applying a random projection ($g_i \overset{i.i.d.}{\sim} \mathcal{N}(0, I_d)$), a random shift ($b_i \overset{i.i.d.}{\sim} U[0, 1]$)) and quantizing the result (width $w > 0$). A concatenation (power) of $\kappa$ such functions gives the final hash function.

They showed that the collision probability for two points $x, y \in \mathbb{R}^d$ at distance $\|x - y\| = c \cdot w$ equals $p_1^\kappa(c)$ where

$$p_1(c) := 1 - 2\Phi(c^{-1}) - \sqrt{\frac{2}{\pi}} c \left(1 - \exp\left\{-\frac{c^{-2}}{2}\right\}\right). \tag{5}$$

In order to evaluate the hashing scheme on a dataset $P$, one needs space and pre-processing time $O(d\kappa \cdot n)$. By picking $w$ large enough, i.e. for small $c$, one can show the following bounds on the collision probability.

**Lemma 3** ((Charikar & Siminelakis, 2017))**.** *For $\delta \leq \frac{1}{2}$ and $c \leq \min\{\delta, \frac{1}{\sqrt{\log(\frac{1}{\delta})}}\}$: $e^{-\sqrt{\frac{2}{\pi}}\delta} \leq \frac{p_1(c)}{e^{-\sqrt{\frac{2}{\pi}}c}} \leq e^{\sqrt{\frac{2}{\pi}}\delta^3}$.*

Taking appropriate powers $\kappa$, one can then use Euclidean LSH to create collision probabilities appropriate for the Exponential and Gaussian kernel.

**Theorem 1** ((Charikar & Siminelakis, 2017))**.** *Let $\mathcal{H}_1(w, \kappa)$ denote the eLSH hashing scheme with width $w$ and power $\kappa \geq 1$. Let $R = \max_{x,y \in P \cup \{q\}} \{\|x - y\|\}$. Then for the*

- ***Laplace kernel,*** *setting $\kappa_e = \lceil \sqrt{2\pi} \log(\frac{1}{\tau}) R \rceil$ and $w_e = \sqrt{\frac{2}{\pi}} \frac{1}{\beta} \kappa_e$ results in $(\beta, \sqrt{e})$-scale free hashing scheme for $k(x, y) = e^{-\|x-y\|_2}$.*
- ***Gaussian kernel,*** *setting $r_t := \frac{1}{2}\sqrt{t \log(1 + \gamma)}$, $\kappa_t := 3\lceil r_t R \rceil^2$, $w_t = \sqrt{\frac{2}{\pi}} \frac{\kappa_t}{r_t}$ results in an estimator for $k(x, y) = e^{-\|x-y\|^2}$ with relative variance*

$$\mathrm{RelVar}[Z_t] \leq V_t(\mu) := 4e^{\frac{3}{2}} \frac{1}{\mu} e^{r_t^2 - r_t \sqrt{\log(\frac{1}{\mu})}}. \tag{6}$$

## 4. Refined Variance bounds and Diagnostics

To understand how many samples are needed to estimate the density through RS or HBE, we need bounds for expressions of the type (4). In particular, for a sequence of numbers $w_1, \ldots, w_n \in [0, 1]$, e.g. $w_i = k(q, x_i)$, and $u \in \Delta_n$ such that $\mu = \sum_{i \in [n]} u_i w_i$, we want to bound:

$$\sup_{u \in \Delta_n, u^\top w = \mu} \sum_{i,j \in [n]} w_i^2 \left(u_i V_{ij} u_j\right) \tag{7}$$

where $V \in \mathbb{R}_+^{n\times n}$ is a non-negative matrix. Our bound will be decomposed into the contribution $\mu_\ell$ from subset of indices $S_\ell \subseteq [n]$ where the weights (kernel values) have bounded range. Specifically, let $\mu \le \lambda \le L \le 1$ and define: $S_1 = \{i \in [n] : L \le w_i \le 1\}, S_2 = \{i \in [n] \setminus S_1 : \lambda \le w_i \le L\}, S_3 = \{i \in [n] \setminus (S_2 \cup S_1) : \mu \le w_i \le \lambda\}, S_4 = \{i \in [n] : w_i < \mu\}$ as well as $\mu_\ell = \sum_{i\in S_\ell} u_i w_i \le \mu$ for $\ell \in [4]$. The intuition behind the definition of the sets is that for radial decreasing kernels, they correspond to spherical annuli around the query.

**Lemma 4.** *For non-negative weights $w_1, \ldots, w_n$, vector $u \in \Delta_n$ and sets $S_1, \ldots, S_4 \subseteq [n]$ as above it holds*

$$\sum_{i,j\in[n]} w_i^2\{u_i V_{ij} u_j\} \le \sum_{\ell\in[3],\ell'\in[3]} \sup_{\substack{i\in S_\ell,\\ j\in S_{\ell'}}} \left\{\frac{V_{ij} w_i}{w_j}\right\} \mu_\ell \mu_{\ell'} + u_{S_4} \sum_{\ell\in[3]} \sup_{\substack{i\in S_\ell,\\ j\in S_4}} \left\{V_{ij}\frac{w_i}{\mu}\right\} \mu_\ell \mu + \sup_{i\in S_4, j\in[n]} \{V_{ij} w_i\} \cdot \mu_4 \quad (8)$$

*where $u_S := \sum_{j\in S} u_j \le 1$.*

This simple lemma allows us to get strong bounds for scale-free hashing schemes simplifying results of (Charikar & Siminelakis, 2017) and extended them to $\beta < 1/2$.

**Theorem 2.** *Let $\mathcal{H}_\nu$ be $(\beta, M)$-scale free hashing scheme and $Z_h$ the corresponding HBE. Then* $\mathrm{RelVar}[Z_h] \le V_{\beta,M}(\mu) := 3M^3/\mu^{\max\{\beta,1-\beta\}}$.

*Proof.* Letting $w_i = k(q, x_i)$, we start from (4) and use the fact that $\mathbb{P}[i, j \in H(q)] \le \min\{p(x_i, q), p(x_j, q)\}$ and that the hashing scheme is $(\beta, M)$-scale free, to arrive at $\mathbb{E}[Z_h^2] \le \sum_{i,j\in[n]} w_i^2\{u_i V_{ij} u_j\}$ with $V_{ij} = M^3 \frac{\min\{w_i^\beta, w_j^\beta\}}{w_i^{2\beta}}$. Let $S_1, \ldots, S_4$ be sets defined for $\ell = L = \mu$. Then $S_2 = S_3 = \emptyset$. By Lemma 4 we get

$$\begin{aligned}
\mathbb{E}[Z_h^2] &\le M^3\Big(\sup_{i,j\in S_1}\{\frac{w_i^{1-2\beta}}{w_j^{1-\beta}}\}\mu_1^2 + \sup_{i\in S_4}\{w_i^{1-\beta}\}\mu_4 \\
&\qquad + u_{S_4}\sup_{i\in S_1, j\in S_4}\{w_i^{1-2\beta} w_j^\beta\}\mu_1\Big) \\
&\le M^3\Big(\frac{1}{\mu^{1-\beta}}\mu_1^2 + u_{S_4}\sup_{i\in S_1}\{\frac{w_i^{1-2\beta}}{\mu^{1-\beta}}\}\mu\mu_1 + \mu^{1-\beta}\mu_4\Big) \\
&\le M^3\Big(\frac{1}{\mu^{1-\beta}} + \frac{1}{\mu^{\max\{\beta,1-\beta\}}} + \frac{1}{\mu^\beta}\Big)\mu^2.
\end{aligned}$$

Noting that $\mathbb{E}[Z_h] = \mu$ and $\mu \le 1$ concludes the proof. □

**Remark 1.** *It is easy to see that random sampling is a (trivial) $(0, 1)$-scale free hashing scheme, hence the relative variance is bounded by $\frac{3}{\mu}$.*

**Remark 2.** *For any $(\frac{1}{2}, M)$-scale free hashing scheme the relative variance is bounded by $\frac{3M^3}{\sqrt{\mu}}$.*

### 4.1. Diagnostics for unbiased estimators

The inequality provides means to quantify the dataset dependent performance of RS and HBE: by setting the coefficients $V_{ij}$ appropriately, Lemma 4 bounds the variance of RS ($V_{ij} = 1$) and HBE ($V_{ij} = \frac{\min\{p(q,x_i), p(q,x_j)\}}{p(q,x_i)^2}$). However, evaluating the bound over the whole dataset is no cheaper than evaluating the methods directly.

Instead, we evaluate the upper bound on a "representative sample" $\tilde{S}_0$ in place of $[n]$ by using the sets $\tilde{S}_\ell = \tilde{S}_0 \cap S_\ell$ for $\ell \in [4]$. Given a set $\tilde{S}_0$ define the estimator $\tilde{\mu}_\ell = \sum_{j\in\tilde{S}_\ell} u_j w_j$. For $\epsilon \in (0, 1)$, let:

$$\lambda_\epsilon := \arg\max_{\tilde{\mu}_0 \le \lambda \le 1} \left\{\tilde{\mu}_3 \le \frac{1}{2}(\epsilon\tilde{\mu}_0 - \tilde{\mu}_4)\right\}, \quad (9)$$

$$L_\epsilon := \arg\min_{\tilde{\mu}_0 \le L \le 1} \left\{\tilde{\mu}_1 \le \frac{1}{2}(\epsilon\tilde{\mu}_0 - \tilde{\mu}_4)\right\}. \quad (10)$$

Since Lemma 4 holds for all $\mu \le \lambda \le L \le 1$, $L_\epsilon, \lambda_\epsilon$ complete the definition of four sets on $\tilde{S}_0$, which we use to evaluate the upper bound. Finally, we produce a representative sample $\tilde{S}_0$ by running our adaptive procedure (Section 5) with Random Sampling. The procedure returns a (random) set $\tilde{S}_0$ such that $\tilde{\mu}_0$ is an $(\epsilon, O(1))$-approximation to $\mu$ for any given query.

**Algorithm 1** Data-dependent Diagnostic

**Input:** set $P$, threshold $\tau \in (0, 1)$, accuracy $\epsilon$, $T \ge 1$, collision probability $p(x, y)$ of the hashing scheme $\mathcal{H}_\nu$.
**for** $t = 1, \ldots, T$ **do**
  $q \leftarrow \mathrm{Random}(P)$ ▷ For each random query
  $(\tilde{S}_0, \tilde{\mu}_0) \leftarrow \mathrm{AMR}(\mathcal{Z}_{RS}(q), \epsilon, \tau)$ ▷ Algorithm 2
  Set $\lambda_\epsilon, L_\epsilon$ using (9) and (10)
  $V_{\mathrm{RS}} \leftarrow$ r.h.s of (8) for $\tilde{S}_0$ and $V_{ij} = 1$.
  $V_{\mathrm{H}} \leftarrow$ r.h.s of (8) for $\tilde{S}_0$, $V_{ij} = \frac{\min\{p(q,x_i), p(q,x_j)\}}{p(q,x_i)^2}$.
  $\mathrm{rV}_{\mathrm{RS}}(t) \leftarrow V_{\mathrm{RS}}/\max\{\tilde{\mu}_0, \tau\}^2$
  $\mathrm{rV}_{\mathrm{HBE}}(t) \leftarrow V_{\mathrm{H}}/\max\{\tilde{\mu}_0, \tau\}^2$.
**Output:** $(\mathrm{mean}_T(r V_{\mathrm{RS}}), \mathrm{mean}_T(r V_{\mathrm{HBE}}))$

## 5. Adaptive estimation of the mean

Towards our goal of producing an $(\epsilon, O(1))$-approximation to $\mu$ for a given query $q$, the arguments in Section 3.1 reduce this task to constructing an unbiased estimator $Z$ of low-relative variance $\mathrm{RelVar}[Z] \le \frac{\epsilon^2}{6}$. Given a basic estimator $Z_0$ and bound $V(\mu)$ on its relative variance, one can always create such an estimator by taking the mean of $O(V(\mu)/\epsilon^2)$ independent samples. The question that remains in order to implement this approach is *how to deal with the fact that $\mu$ and thus $V(\mu)$ is unknown?*

We handle this by providing an improvement to the adaptive estimation procedure of (Charikar & Siminelakis, 2017).

The procedure makes a guess of the density and uses the guess to set the number of samples. Subsequently, the procedure performs a consistency check to evaluate whether the guess was correct up to a small constant and if not, revises the guess and increases the sample size. The procedure applies *generically* to the following class of estimators (whose variance varies polynomially with $\mu$) that we introduce.

### 5.1. $(\alpha, \beta, \gamma)$-regular estimators

**Definition 5.** *For $\alpha, \beta \in (0, 2]$ and $\gamma \geq 1$, an estimator $\mathcal{Z}$ is $(\alpha, \beta, \gamma)$-regular if for some constant $C \geq 1$ and all $t \in [T]$, there exist a random variable $Z_t$ and a function $V_t : (0, 1] \to \mathbb{R}_{++}$ such that*

*(A)* $\mathbb{E}[Z_t] = \mu$ *and* $\mathrm{RelVar}[Z_t] \leq V_t(\mu)$, $\forall \mu \in (0, 1]$,

*(B)* $1 \leq \frac{V_t(y)}{V_t(x)} \leq \left(\frac{x}{y}\right)^{2-\alpha}$ *for all* $x \geq y > 0$,

*(C)* $V_t(\mu_{t+1}) \leq C\mu_t^{-\beta}$ *with* $\mu_t := (1 + \gamma)^{-t}$.

*We write $Z_t \sim \mathcal{Z}(t, \gamma)$ to denote a sample from such an estimator at level $t \in [T]$.*

At a high level, regular estimators generalize properties of scale-free estimators relevant to adaptively estimating the mean. Property $(B)$ affects the success probability whereas $(C)$ affects the running time and space requirements. This level of generality will be necessary to analyze the hashing scheme designed for the Gaussian kernel.

**Regular Estimators via HBE.** For any $t \in [T]$, given a collection of hash tables $\{H_t^{(i)}\}_{i \in [m_t]}$ created by evaluating $m_t$ i.i.d hash functions with collision probability $p_t(x, y)$ on a set $P$ and a fixed kernel $k(x, y)$, we will denote by

$$\mathcal{Z}(t, \gamma) \leftarrow \mathrm{HBE}_{k,p_t}(\{H_t^{(i)}\}_{i \in [m_t]}) \tag{11}$$

a data structure at level $t$, that for any query $q$ is able to produce up to $m_t$ i.i.d unbiased random variables $Z_t^{(i)}(q)$ for the density $\mu$ according to Section 3.2. The union of such data structures for $t \in [T]$ will be denoted by $\mathcal{Z}$.

Before proceeding with the description of the estimation procedure we show that $(\beta, M)$-scale free HBE (that include random sampling) satisfy the above definition.

**Theorem 3.** *Given a $(\beta, M)$-scale free hashing scheme with $\beta \geq \frac{1}{2}$, the corresponding estimator is $(2 - \beta, \beta, \gamma)$-regular with constant $C = 3M^3(1 + \gamma)^\beta$.*

The proof follows easily by Theorem 2 by using the same estimator for all $t \in [T]$ and $V_t(\mu) = V(\mu) = \frac{3M^3}{\mu^\beta}$.

### 5.2. Adaptive Mean Relaxation

For regular estimators we propose the following procedure (Algorithm 2). In the supplementary material we analyze the probability that this procedure successfully estimates the mean as well as the number of samples it uses and obtain the following result.

**Algorithm 2** Adaptive Mean Relaxation (AMR)

**Input:** $(a, \beta, \gamma)$-regular estimator $\mathcal{Z}$, accuracy $\epsilon \in (0, 1)$, threshold $\tau \in (0, 1)$.
$T \leftarrow \lceil \log_{1+\gamma}(\frac{1}{\epsilon\tau}) \rceil + 1$
**for** $t = 1, \ldots, T$ **do** ▷ For each level
  $\mu_t \leftarrow (1 + \gamma)^{-t}$ ▷ Current guess of mean
  $m_t \leftarrow \lceil \frac{6}{\epsilon^2} V_t(\mu_{t+1}) \rceil$ ▷ sufficient samples in level $t$
  $Z_t^{(i)} \sim \mathcal{Z}(t, \gamma)$ i.i.d samples for $i \in [m_t]$.
  $\bar{Z}_t \leftarrow \mathrm{mean}\{Z_t^{(1)}, \ldots, Z_t^{(m_t)}\}$
  **if** $\bar{Z}_t \geq \mu_t$ **then** ▷ consistency check
    **return** $\bar{Z}_t$
**return** 0 ▷ In this case $\mu \leq \epsilon\tau$

**Theorem 4.** *Given an $(a, \beta, \gamma)$-regular estimator $\mathcal{Z}$, the AMR procedure outputs a number $\hat{Z}$ such that*

$$\mathbb{P}[|\hat{Z} - \mu| \leq \epsilon \cdot \max\{\mu, \tau\}] \geq \frac{2}{3} - O_{\gamma,\alpha}(\epsilon^2)$$

*and with the same probability uses $O_\gamma(\frac{1}{\epsilon^2}\frac{1}{\mu^\beta})$ samples.*

Using the bounds on the relative variance in Section 4, we get that any $(\beta \geq 1/2, M)$-scale free estimator can be used to estimate the density $\mu$ using $O(1/\epsilon^2\mu^\beta)$ samples.

### 5.3. Regular estimator for Gaussian Kernel

We show next show a construction of a HBE for Gaussian kernel (Algorithm 3) that results in a *regular estimator*.

**Algorithm 3** Gaussian HBE (Gauss-HBE)

**Input:** $\gamma \geq 1$, $\tau \in (0, 1)$, $\epsilon \in (0, 1)$, dataset $P$.
$T \leftarrow \lceil \log_{1+\gamma}(\frac{1}{\epsilon\tau}) \rceil + 1$, $R \leftarrow \sqrt{\log(1/\epsilon\tau)}$
**for** $t = 1, \ldots, T$ **do**
  $m_t \leftarrow \lceil \frac{6}{\epsilon^2} V_t((1 + \gamma)^{-(t+1)}) \rceil$ ▷ see (6)
  **for** $i = 1, \ldots, m_t$ **do**
    $H_t^{(i)} \leftarrow \mathrm{eLSH}(w_t, \kappa_t, P)$ ▷ see Theorem 3
    $p_t(x, y) := p_1^{\kappa_t}(\|x - y\|/w_t)$ ▷ see (5)
    $k(x, y) := e^{-\|x-y\|^2}$
  $\mathcal{Z}_{\mathrm{Gauss}}(t, \gamma) \leftarrow \mathrm{HBE}_{k,p_t}(\{H_t^{(i)}\}_{i \in [m_t]})$
**Output:** $\mathcal{Z}_{\mathrm{Gauss}}$

**Theorem 5.** *$\mathcal{Z}_{\mathrm{Gauss}}$ is $(1, \frac{3}{4}, \gamma)$-regular and takes preprocessing time/space bounded by $O_{d,\kappa_T,\gamma}(\epsilon^{-3+\frac{1}{4}}\tau^{-\frac{3}{4}} \cdot n)$.*

The proof (given in the supplementary material) is based on using (6) to show that Definition 5 is satisfied with appropriate selection of constants. We also note that (6) can be derived using Lemma 4.

## 6. Sketching the Kernel Density Estimate

In this section, we introduce an approach based on hashing to create a "sketch" of KDE that we can evaluate using HBE or other methods.

**Definition 6.** *Let $(u, P)$ be a set of weights and points. We call $(w, S)$ an $(\epsilon, \delta, \tau)$-sketch iff for any $q \in \mathbb{R}^d$:*

$$\mathbb{E}[|\mathrm{KDE}_S^w(q) - \mathrm{KDE}_P^u(q)|^2] \leq \epsilon^2 \tau \delta \cdot \mathrm{KDE}_P^u(q). \quad (12)$$

Let $\mu = \mathrm{KDE}_P^u(q)$, using Chebyshev's inequality it is immediate that any $(\epsilon, \delta, \tau)$-sketch satisfies for any $q \in \mathbb{R}^d$: $\mathbb{P}[|\mathrm{KDE}_S^w(q) - \mu| \geq \epsilon \max\{\tau, \mu\}] \leq \delta$.

**Remark 3.** *It is easy to see that one can construct such a sketch by random sampling $m \geq \frac{1}{\epsilon^2 \delta} \frac{1}{\tau}$ points.*

**Hashing-Based-Sketch (HBS).** The scheme we propose samples a random point by first sampling a hash bucket $H_i$ with probability $\propto u_{H_i}^\gamma$ and then sampling a point $j$ from the bucket with probability $\propto u_j$. The weights are chosen such that the sample is an unbiased estimate of the density. This scheme interpolates between uniform over buckets and uniform over the whole data set as we vary $\gamma \in [0, 1]$. We pick $\gamma^*$ so that the variance is controlled and with the additional property that any non-trivial bucket $u_{H_i} \geq \tau$ will have a representative in the sketch with some probability. This last property is useful in estimating low-density points (e.g. for outlier detection (Gan & Bailis, 2017)). For any hash table $H$ and a vector $u \in \Delta_n$ (simplex), let $B = B(H)$ denote the number of buckets and $u_{\max} = u_{\max}(H) := \max\{u_{H_i} : i \in [B]\}$ the maximum weight of any hash bucket of $H$. The precise definition of our Hashing-Based-Sketch is given in Algorithm 4.

**Theorem 6.** *Let $H$ be the hash function sampled by the HBS procedure. For $\epsilon > 0$ and $\delta \in [e^{-\frac{6}{\epsilon^2}\frac{u_{\max}}{n\tau}}, e^{-\frac{6}{\epsilon^2}})$, let:*

$$\gamma^* = \left\{1 - \frac{\log(\frac{\epsilon^2}{6}\log(1/\delta))}{\log(\frac{u_{\max}}{\tau})}\right\}^{\mathbb{I}[B \leq (\frac{1}{2})^{\frac{1}{6}}\frac{1}{\tau}]}, \quad (13)$$

$$m = \frac{6}{\epsilon^2}\frac{1}{\tau}(B u_{\max})^{1-\gamma^*} < \frac{\log(\frac{1}{\delta})}{\tau}. \quad (14)$$

*Then $(S_m, w)$ is an $(\epsilon, \frac{1}{6}, \tau)$-sketch and if $B \leq (\frac{1}{2})^{\frac{1}{6}}\frac{1}{\tau}$ any hash bucket with weight at least $\tau$ will have non empty intersection with $S_m$ with probability at least $1 - \delta$.*

## 7. Experiments

In this section, we evaluate the performance of hashing-based methods on kernel evaluation on real and synthetic datasets, as well as test the diagnostic procedure's ability to predict dataset-dependent estimator performance [2].

[2]Source code available at: http://github.com/kexinrong/rehashing

**Algorithm 4** Hashing-Based-Sketch (HBS)

**Input:** set $P$, sketch size $m$, hashing scheme $\mathcal{H}_\nu$, threshold $\tau \in (0, 1)$, $u \in \Delta_n$
Sample $h \sim \mathcal{H}_\nu$ and create hash table $H = h(P)$.
Set $\gamma$ according to (13)
$S_m \leftarrow \emptyset$, $w \leftarrow 0 \cdot \mathbf{1}_m$, $B \leftarrow B(H)$
**for** $j = 1, \ldots, m$ **do**
  Sample hash bucket $H_i$ with probability $\propto u_{H_i}^\gamma$
  Sample a point $X_j$ from $H_i$ with probability $\propto u_j$
  $S_m \leftarrow S_m \cup \{X_j\}$
  $w_j(\gamma, m) \leftarrow \frac{u_{H_i}}{m} \frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_i}^\gamma}$
**Output:** $(S_m, w)$

### 7.1. Experimental Setup

**Baselines and Evaluation Metrics.** As baselines for density estimation (using $u = \frac{1}{n}\mathbf{1}$), we compare against ASKIT (March et al., 2016), a tree-based method FigTree (Morariu et al., 2009), and random sampling (RS). We used the open sourced libraries for FigTree and ASKIT; for comparison, all implementations are in C++ and we report results on a single core. We tune each set of parameters via binary search to guarantee an average relative error of at most 0.1. By default, the reported query time and relative error are averaged over 10K random queries that we evaluate exactly as ground truth. The majority of query densities in evaluation are larger than $\tau = 10^{-4}$ (roughly $\frac{1}{10\sqrt{n}}$).

**Synthetic Benchmarks.** To evaluate algorithms under generic scenarios, we need benchmarks with diverse structure. The variance bound suggests that having a large number of points far from the query is hard for HBE, whereas having a few points close to the query is hard for RS. In addition, the performance of space-partitioning methods depends mostly on how clustered vs spread-out the datasets are, with the latter being harder. A fair benchmark should therefore include all above regimes.

We propose a procedure that generates a $d$-dimensional dataset where for $D$ random directions, clusters of points are placed on $s$ different distance scales, such that *i)* each distance scale contributes equally to the kernel density at the origin, *ii)* the outermost scale has $n$ points per direction, *iii)* the kernel density around the origin is approximately $\mu$. By picking $D \gg n$ the instances become more random like, while for $D \ll n$ they become more clustered. Using this procedure, we create two families of instances:

- **"worst-case":** we take the union of two datasets generated for $D = 10, n = 50\text{K}$ and $D = 5\text{K}$, $n = 100$ while keeping $s = 4$, $\mu = 10^{-3}$ fixed and varying $d$.
- **$D$-structured:** we set $N = 500\text{K}$, $s = 4$, $\mu = 10^{-3}$, $d = 100$ and vary $D$ while keeping $nD = N$.

*Table 1.* Comparison of preprocess time and average query time on real world datasets. Bold numbers correspond to the best result. The sources of the datasets are: MSD (Bertin-Mahieux et al., 2011), GloVe (Pennington et al., 2014), SVHN (Netzer et al., 2011), TMY3 (Hendron & Engebrecht, 2010), covtype (Blackard & Dean, 1999), TIMIT (Garofolo, 1993).

| | | | | Preprocess Time (sec) | | | | Average Query Time (ms) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Description | n | d | HBE | FigTree | ASKIT | RS | HBE | FigTree | ASKIT | RS |
| TMY3 | energy | $1.8M$ | 8 | 15 | 28 | 838 | 0 | **0.8** | 3.0 | 28.4 | 55.0 |
| census | census | $2.5M$ | 68 | 66 | 4841 | 1456 | 0 | **2.8** | 1039.9 | 103.7 | 27.5 |
| covertype | forest | $581K$ | 54 | 22 | 31 | 132 | 0 | **1.9** | 23.0 | 47.0 | 149.4 |
| TIMIT | speech | $1M$ | 440 | 298 | 1579 | 439 | 0 | **24.3** | 1531.8 | 169.8 | 32.8 |
| ALOI | image | $108K$ | 128 | 24 | 70 | 14 | 0 | 6.3 | 53.5 | **5.4** | 21.2 |
| SVHN | image | $630K$ | 3072 | 2184 | $> 10^5$ | 877 | 0 | 67.9 | $> 10^4$ | **43.0** | 370.1 |
| MSD | song | $463K$ | 90 | 55 | 2609 | 107 | 0 | 5.2 | 326.9 | 8.1 | **2.1** |
| GloVe | embedding | $400K$ | 100 | 98 | 5603 | 76 | 0 | 19.0 | 656.5 | 86.1 | **5.0** |

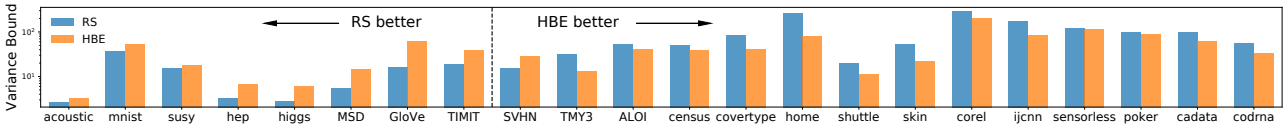



*Figure 3.* Predicted relative variance upper bound on real datasets. RS exhibits lower variance for datasets on the left. Overall, the diagnostic procedure correctly predicts the performance all but one dataset (SVHN).

In the supplementary material we include a precise description of the procedure as well as example scatter plots.

### 7.2. Results

**Synthetic datasets.** We evaluate the four algorithms for kernel density estimation on worst-case instances with $d \in [10, 500]$ and on $D$-structured instances with $D \in [1, 100K]$. For "worst-case" instances, while all methods experience increased query time with increased dimension, HBE consistently outperforms other methods for dimensions ranging from 10 to 500 (Figure 4 left). For the $D$-structured instances (Figure 4 right), FigTree and RS dominate the two extremes where the dataset is highly "clustered" $D \ll n$ or "scattered" on a single scale $D \gg n$, while HBE achieves the best performance for datasets in between. ASKIT's runtime is relatively unaffected by the number of clusters but is outperformed by other methods.

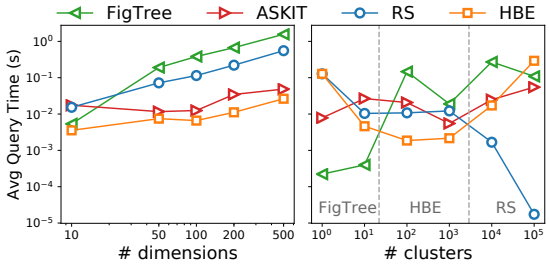



*Figure 4.* Synthetic Experiment.

**Real-world datasets.** We repeat the above experiments on eight large real-world datasets from various domains. We z-normalize each dataset dimension, and tune bandwidth based on Scott's rule (Scott, 2015). We exclude a small percent of queries whose density is below $\tau$. Table 1 reports the preprocessing time as well as average query time for each method. Overall, HBE achieves the best average query time on four datasets; we focus on query time since it dominates the runtime given enough queries. With sketching, HBE also exhibits comparable preprocessing overhead to FigTree and ASKIT. While the performances of FigTree and ASKIT degrades with the increased dimension and dataset size respectively, HBE remains competitive across the board.

**Diagnosis.** To assess the accuracy of the diagnostic procedure, we compare the predicted variance of RS and HBE on an extended set of datasets. RS exhibits lower variances (smaller error with the same number of samples) for datasets on the left. Overall, our proposed diagnosis correctly identified the better estimator for all but one dataset. Notice that although RS has better sampling efficiency on TIMIT, HBE ends up having better query time since the frequent cache misses induced by the large dataset dimension offset the additional computation cost of HBE. For all datasets in Table 1, the diagnosis costs between 8% to 59% of the setup time of HBE, indicating that running the diagnosis is cheaper than running even a single query with the HBE.

**Supplementary material.** We include an experimental comparison between HBS and competing methods (Chen et al., 2010; Cortes & Scott, 2017), where it is consistently the best method for low density queries while having similar performance for random queries. We also show how to use the diagnostic procedure to produce dataset visualizations.

## Acknowledgments

We thank Leslie Greengard for valuable conversations and encouragement in earlier stages of this work. We also thank Edward Gan, Daniel Kang, Sahaana Suri, and Kai-Sheng Tai for feedback on an early draft of this paper. The first author has been partially supported by an Onassis Foundation Scholarship. This research was supported in part by affiliate members and other supporters of the Stanford DAWN project—Ant Financial, Facebook, Google, Intel, Microsoft, NEC, SAP, Teradata, and VMware – as well as Toyota Research Institute, Keysight Technologies, Northrop Grumman, and Hitachi. We also acknowledge the support of the Intel/NSF CPS Security under grant No. 1505728, the Stanford Secure Internet of Things Project and its supporters (VMware, Ford, NXP, Google), and the Stanford System X Alliance.

## References

Andoni, A. and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pp. 459–468. IEEE, 2006.

Backurs, A., Charikar, M., Indyk, P., and Siminelakis, P. Efficient Density Evaluation for Smooth Kernels. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 615–626. IEEE, 2018.

Belkin, M., Ma, S., and Mandal, S. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, vol.24:131–151, 1999.

Buhmann, M. D. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.

Charikar, M. and Siminelakis, P. Hashing-Based-Estimators for kernel density in high dimensions. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pp. 1032–1043. IEEE, 2017.

Charikar, M. and Siminelakis, P. Multi-Resolution Hashing for Fast Pairwise Summations. *arXiv preprint arXiv:1807.07635*, 2018.

Chen, B., Xu, Y., and Shrivastava, A. Lsh-sampling breaks the computational chicken-and-egg loop in adaptive stochastic gradient estimation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018. URL https://openreview.net/forum?id=Hk8zZjRLf.

Chen, Y., Welling, M., and Smola, A. Super-samples from Kernel Herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pp. 109–116, Arlington, Virginia, United States, 2010. AUAI Press. ISBN 978-0-9749039-6-5.

Cortes, E. C. and Scott, C. Sparse Approximation of a Kernel Mean. *Trans. Sig. Proc.*, 65(5):1310–1323, March 2017. ISSN 1053-587X.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 253–262. ACM, 2004.

Gan, E. and Bailis, P. Scalable kernel density classification via threshold-based pruning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 945–959. ACM, 2017.

Garofolo, J. S. TIMIT acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993.

Gray, A. G. and Moore, A. W. N-body'problems in statistical learning. In *Advances in neural information processing systems*, pp. 521–527, 2001.

Gray, A. G. and Moore, A. W. Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 203–211. SIAM, 2003.

Greengard, L. and Rokhlin, V. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2): 325–348, 1987.

Hendron, R. and Engebrecht, C. Building america research benchmark definition: Updated december 2009, 2010.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613. ACM, 1998.

Lee, D. and Gray, A. G. Fast high-dimensional kernel summations using the monte carlo multipole method. In *Advances in Neural Information Processing Systems*, pp. 929–936, 2009.

Lee, D., Moore, A. W., and Gray, A. G. Dual-tree fast gauss transforms. In *Advances in Neural Information Processing Systems*, pp. 747–754, 2006.

Luo, C. and Shrivastava, A. Arrays of (locality-sensitive) Count Estimators (ACE): Anomaly Detection on the Edge. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 1439–1448. International World Wide Web Conferences Steering Committee, 2018a.

Luo, C. and Shrivastava, A. Scaling-up Split-Merge MCMC with Locality Sensitive Sampling (LSS). *arXiv preprint arXiv:1802.07444*, 2018b.

March, W., Xiao, B., and Biros, G. ASKIT: Approximate Skeletonization Kernel-Independent Treecode in High Dimensions. *SIAM Journal on Scientific Computing*, 37 (2):A1089–A1110, 2015.

March, W., Xiao, B., Yu, C., and Biros, G. ASKIT: An Efficient, Parallel Library for High-Dimensional Kernel Summations. *SIAM Journal on Scientific Computing*, 38 (5):S720–S749, 2016.

Morariu, V. I., Srinivasan, B. V., Raykar, V. C., Duraiswami, R., and Davis, L. S. Automatic online tuning for fast Gaussian summation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1113–1120. Curran Associates, Inc., 2009.

Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NeurIPS workshop on deep learning and unsupervised feature learning*, 2011.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

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

Phillips, J. M. and Tai, W. M. Near-optimal coresets of kernel density estimates. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 99. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Ram, P., Lee, D., March, W., and Gray, A. G. Linear-time algorithms for pairwise statistical problems. In *Advances in Neural Information Processing Systems*, pp. 1527–1535, 2009.

Schölkopf, B., Smola, A. J., Bach, F., et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Scott, D. W. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

Spring, R. and Shrivastava, A. A New Unbiased and Efficient Class of LSH-Based Samplers and Estimators for Partition Function Computation in Log-Linear Models. *arXiv preprint arXiv:1703.05160*, 2017.

Sriperumbudur, B. et al. On the optimal estimation of probability measures in weak and strong topologies. *Bernoulli*, 22(3):1839–1893, 2016.

Wu, X., Charikar, M., and Natchu, V. Local density estimation in high dimensions. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5296–5305. PMLR, 10–15 Jul 2018.

Zheng, Y., Jestes, J., Phillips, J. M., and Li, F. Quality and efficiency for kernel density estimates in large data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 433–444. ACM, 2013.

# Supplementary material for: Rehashing Kernel Evaluation in High Dimensions

**Paris Siminelakis** [*1] **Kexin Rong** [*1] **Peter Bailis** [1] **Moses Charikar** [1] **Philip Levis** [1]

**Outline** In the first two sections, we give proofs for all our formal results while restating them for convenience. In Section 2, we give the precise description of our Hashing-Based-Sketch and its theoretical analysis. In Section 3, we present our diagnostic and visualization procedures in more detail. In Section 4, we explain our design decisions and the procedure for generating a fair synthetic benchmark. In Section 5, we provide additional setup details and results for the experimental evaluation.

## 1. Proofs

### 1.1. Basic inequalities

We first state without proof some well known inequalities that we will use in the proofs.

**Lemma 1** (Chebyshev's and Paley-Zygmund inequalities)**.** *For a non-negative random variable $Z$ and parameters $t > 0$, $\theta \in [0,1]$, we have*

$$\mathbb{P}[Z \geq (t+1)\cdot \mathbb{E}[Z]] \leq \frac{1}{t^2}\cdot \mathrm{RelVar}[Z], \tag{1}$$

$$\mathbb{P}[Z > (1-\theta)\mathbb{E}[Z]] \geq \frac{1}{1+\frac{1}{\theta^2}\cdot \mathrm{RelVar}[Z]}. \tag{2}$$

**Theorem 1** (Chernoff bounds)**.** *Let $X = \sum_{i=1}^n X_i$, where $X_i = 1$ with probability $p_i$ and $X_i = 0$ with probability $1-p_i$, and all $X_i$ are independent. Let $v = \mathbb{E}[X] = \sum_{i=1}^n p_i$. Then for $\delta > 0$*

$$\mathbb{P}[X \geq (1+\delta)v] \leq e^{-\frac{\delta^2}{2+\delta}v}, \tag{3}$$

$$\mathbb{P}[X \leq (1-\delta)v] \leq e^{-\frac{1}{2}\delta^2 v}. \tag{4}$$

### 1.2. Median-trick to boost success probability

The median-trick is based on concentration of sums of independent binary random variables. If we define binary random variables appropriately we can obtain bounds for the concentration of the median of i.i.d. random variables around their expectation.

**Lemma 2.** *Let $Z_1, \ldots, Z_L$ be $L \geq 1$ i.i.d. copies of a non-negative random variable with $\mathrm{RelVar}[Z] \leq \frac{\epsilon^2}{6}$ then:*

$$\mathbb{P}\left[\mathrm{median}\{Z_1,\ldots,Z_L\} \geq (1+\epsilon)\mathbb{E}[Z]\right] \leq e^{-\frac{L}{6}},$$
$$\mathbb{P}\left[\mathrm{median}\{Z_1,\ldots,Z_L\} \leq (1-\epsilon)\mathbb{E}[Z]\right] \leq e^{-\frac{L}{4}}.$$

*Proof of Lemma 2.* Let

$$X_i = \mathbb{I}[Z_i \geq (1+\epsilon)\mathbb{E}[Z]],$$
$$Y_i = \mathbb{I}[Z_i \leq (1-\epsilon)\mathbb{E}[Z]].$$

By Lemma 1, we have that

$$a_i = \mathbb{E}[X_i] \leq \frac{1}{\epsilon^2}\frac{\epsilon^2}{6} \leq \frac{1}{6}, \quad b_i = \mathbb{E}[Y_i] \leq \frac{1}{7}.$$

We get the following upper bounds

$$\mathbb{P}[\mathrm{median}\{Z_1,\ldots,Z_L\} \geq (1+\epsilon)\mathbb{E}[Z]] \leq \mathbb{P}[\sum_{i=1}^L X_i \geq \frac{L}{2}].$$

$$\mathbb{P}[\mathrm{median}\{Z_1,\ldots,Z_L\} \geq (1+\epsilon)\mathbb{E}[Z]] \leq \mathbb{P}[\sum_{i=1}^L Y_i \geq \frac{L}{2}],$$

that along with Chernoff bounds will give us our result. We only show the first inequality as the second one follows similarly. Let $A = \sum_{i=1}^L a_i \leq L/6$, the first event is bounded by $\exp(-\frac{(\frac{L/2}{A}-1)^2}{2+(\frac{L/2}{A}-1)}A) \leq \exp(-L/6)$. □

### 1.3. Moments of Hashing-Based-Estimators

**Lemma 3.** *Assuming that $\forall i \in [n], p(x_i, q) > 0$ then*

$$\mathbb{E}[Z_h] = \sum_{i=1}^n u_i k(x, x_i), \tag{5}$$

$$\mathbb{E}[Z_h^2] = \sum_{i,j=1}^n k^2(q, x_i)\frac{u_i \mathbb{P}[i,j \in H(q)] u_j}{p^2(q,x_i)}. \tag{6}$$

[*]Equal contribution [1]Stanford University, Stanford, California, US. Correspondence to: Paris Siminelakis <psimin@stanford.edu>, Kexin Rong <krong@stanford.edu>.

*Proceedings of the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

*Proof of Lemma 3.* We start with the expectation:

$$\begin{aligned}\mathbb{E}_{h,X}[\frac{k(q,X)}{p(q,X)}u_{H(q)}] &= \mathbb{E}_h[\mathbb{E}_X[\frac{k(q,X)}{p(q,X)}]u_{H(q)}] \\ &= \mathbb{E}_h[\sum_{i\in H(q)}\frac{u_i}{u_{H(q)}}\frac{k(q,x_i)}{p(q,x_i)}u_{H(q)}] \\ &= \sum_{i=1}^{n} u_i\mathbb{E}[\mathbb{I}[h(x_i)=h(q)]]\frac{k(x_i,q)}{p(x_i,q)} \\ &= \sum_{i=1}^{n} u_i k(x_i,q)\end{aligned}$$

We proceed with the second moment:

$$\begin{aligned}\mathbb{E}_{h,X}[\frac{k^2(q,X)}{p^2(q,X)}u^2_{H(q)}] &= \mathbb{E}_h[\mathbb{E}_X[\frac{k^2(q,X)}{p^2(q,X)}]u^2_{H(q)}] \\ &= \mathbb{E}_h[\sum_{i\in H(q)}\frac{u_i}{u_{H(q)}}\frac{k^2(q,x_i)}{p^2(q,x_i)}u^2_{H(q)}] \\ &= \mathbb{E}_h[\sum_{i\in H(q)} u_i\frac{k^2(q,x_i)}{p^2(q,x_i)}u_{H(q)}] \\ &= \mathbb{E}_h[\sum_{i,j\in H(q)} u_iu_j\frac{k^2(q,x_i)}{p^2(q,x_i)}] \\ &= \sum_{i,j=1}^{n} k^2(x_i,q)\frac{u_i\mathbb{P}[i,j\in H(q)]u_j}{p^2(x_i,q)}\end{aligned}$$

□

### 1.4. Refined Variance bound

Here, we derive our new inequality bounding the variance of HBE and RS. Let $\mu \le \lambda \le L \le 1$ and define:

$$S_1 = \{i\in[n] : L \le w_i \le 1\} \tag{7}$$
$$S_2 = \{i\in[n]\setminus S_1 : \lambda \le w_i \le L\} \tag{8}$$
$$S_3 = \{i\in[n]\setminus(S_2\cup S_1) : \mu \le w_i \le \lambda\} \tag{9}$$
$$S_4 = \{i\in[n] : w_i < \mu\} \tag{10}$$

as well as $\mu_\ell = \sum_{i\in S_\ell} u_i w_i \le \mu$. The intuition behind the definition of the sets is that for radial decreasing kernels they correspond to spherical annuli around the query (Figure 1).

**Lemma 4.** *For non-negative weights $w_1,\dots,w_n$, vector $u\in\Delta_n$ and sets $S_1,\dots,S_4\subseteq[n]$ as above it holds*

$$\begin{aligned}\sum_{i,j\in[n]} w_i^2\{u_iV_{ij}u_j\} \le &\sum_{\ell\in[3],\ell'\in[3]}\sup_{\substack{i\in S_\ell,\\ j\in S_{\ell'}}}\left\{\frac{V_{ij}w_i}{w_j}\right\}\mu_\ell\mu_{\ell'} \\ &+ u_{S_4}\sum_{\ell\in[3]}\sup_{\substack{i\in S_\ell,\\ j\in S_4}}\left\{V_{ij}\frac{w_i}{\mu}\right\}\mu_\ell\mu \\ &+ \sup_{i\in S_4,j\in[n]}\{V_{ij}w_i\}\cdot\mu_4 \end{aligned}\tag{11}$$

*where $u_S := \sum_{j\in S} u_j \le 1$.*

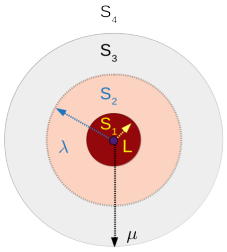



*Figure 1.* Depiction of the sets that appear in Lemma 4

*Proof of Lemma 4.* First we observe that $S_1\uplus S_2\uplus S_3\uplus S_4 = [n]$ forms a partition:

$$\begin{aligned}\sum_{i,j\in[n]} u_iu_jV_{ij}w_i^2 = &\sum_{\ell,\ell'\in[3]}\sum_{i\in S_\ell,j\in S_{\ell'}} u_iu_jV_{ij}w_i^2 \\ &+ \sum_{\ell\in[3]}\sum_{i\in S_\ell,j\in S_4} u_iu_jV_{ij}w_i^2 \\ &+ \sum_{i\in S_4,j\in[n]} u_iu_jV_{ij}.w_i^2\end{aligned}\tag{12}$$

For the first three sets we have some bounds on the ration $\frac{w_i}{w_j}$ whereas for the last set we have a bound on the $w_i$. We utilize these by:

$$\sum_{\substack{i\in S_\ell,\\ j\in S_{\ell'}}}\frac{V_{ij}w_i}{w_j}u_iw_iu_jw_j \le \sup_{\substack{i\in S_\ell,\\ j\in S_{\ell'}}}\{\frac{V_{ij}w_i}{w_j}\}\sum_{i\in S_\ell} w_iu_i\sum_{j\in S_{\ell'}} w_ju_j,$$

$$\sum_{\substack{i\in S_\ell,\\ j\in S_4}}\frac{V_{ij}w_i}{\mu}w_iu_iu_j\mu \le u_{S_4}\sup_{\substack{i\in S_\ell,\\ j\in S_4}}\left\{\frac{V_{ij}w_i}{\mu}\right\}\mu\sum_{i\in S_\ell} w_iu_i,$$

$$\sum_{\substack{i\in S_4,\\ j\in[n]}}\{V_{ij}w_i\}u_iu_jw_i \le \sup_{i\in S_4,j\in[n]}\{V_{ij}w_i\}\|u\|_1\sum_{j\in S_4} w_iu_i.$$

Identifying $\mu_i$ in the above expressions and substituting the bounds in (12) completes the proof. □

### 1.5. Adaptive procedure

**Theorem 2.** *Given an $(a,\beta,\gamma)$-regular estimator $\mathcal{Z}$, the AMR procedure outputs a number $\hat{Z}$ such that*

$$\mathbb{P}[|\hat{Z}-\mu| \le \epsilon\cdot\max\{\mu,\tau\}] \ge \frac{2}{3} - O_{\gamma,\alpha}(\epsilon^2)$$

*and with the same probability uses $O_\gamma(\frac{1}{\epsilon^2}\frac{1}{\mu^\beta})$ samples.*

*Proof of Theorem 2.* Recall that $\mu_t = (1+\gamma)^{-t}$ and let $t_0 := t_0(\mu)\in\mathbb{Z}$ such that:

$$\mu_{t_0+1} \le \mu \le \mu_{t_0} \tag{13}$$

We consider two cases $t_0 < T$ or $t_0 \ge T$.

**Case I** ($t_0 < T$)**.** In this case, we want to show that our algorithm with constant probability does not terminate before $t_0$ and not after $t_0 + 1$.

Let $\bar{Z}_t$ be the mean of $m_t$ i.i.d. samples $Z_t^{(i)} \sim \mathcal{Z}(t, \gamma)$ with mean $\mathbb{E}[Z_t^{(i)}] = \mu$ and $\mathrm{RelVar}[Z_t^{(i)}] \le V_t(\mu)$. Then,

$$\mathrm{RelVar}[\bar{Z}_t] \le \frac{\epsilon^2}{6} \frac{V_t(\mu)}{V_t(\mu_{t+1})}. \tag{14}$$

Let $A_0$ be the event that the algorithm terminates before $t_0$.

$$\begin{aligned}
\mathbb{P}[A_0] &= \mathbb{P}[\exists t < t_0, \bar{Z}_t \ge \mu_t] && (15)\\
&\le \sum_{t<t_0} \mathbb{P}[\bar{Z}_t \ge \left(\frac{\mu_t}{\mu}\right)\mu] && (16)\\
&\le \frac{\epsilon^2}{6} \sum_{t=1}^{t_0-1} \frac{\mu^2}{(\mu_t-\mu)^2} \frac{V_t(\mu)}{V_t(\mu_{t+1})} && (17)\\
&\le \frac{\epsilon^2}{6} \sum_{t=1}^{t_0-1} \frac{\mu^2}{(\mu_t-\mu)^2} (\frac{\mu_{t+1}}{\mu})^{2-\alpha}. && (18)
\end{aligned}$$

where in (16) we use union bound, in (17) we use the first part of Lemma 1 and in (18) property (B) of a regular estimator. In the next three inequalities we use (13), $t \le t_0 - 1$ and $\sum_{t=0}^{s} x^s \le (1-x)^{-1}$ for $x < 1$.

$$\begin{aligned}
\mathbb{P}[A_0] &\le \frac{\epsilon^2}{6} \sum_{t=1}^{t_0-1} \frac{1}{(1-\frac{\mu_{t_0}}{\mu_t})^2} \frac{\mu_{t+1}^2}{\mu_t^2} (\frac{\mu_{t_0}}{\mu_{t+1}})^{\alpha} && (19)\\
&\le \frac{\epsilon^2}{6} \frac{1}{\gamma^2} \mu_{t_0}^{\alpha} \sum_{t=1}^{t_0-1} (1+\gamma)^{-\alpha(t_0-t-1)} && (20)\\
&\le \frac{\epsilon^2}{6} \frac{1}{\gamma^2} \mu_{t_0}^{\alpha} \frac{1}{1-(1+\gamma)^{-\alpha}}. && (21)
\end{aligned}$$

Furthermore, let $A_1$ be the event that the algorithm terminates after $t > t_0 + 1$.

$$\begin{aligned}
\mathbb{P}[A_1] &= \mathbb{P}[\forall t \le t_0 + 1, \bar{Z}_t < \mu_t] && (22)\\
&\le \mathbb{P}[\bar{Z}_{t_0+1} < \mu_{t_0+1}] && (23)\\
&= 1 - \mathbb{P}[\bar{Z}_{t_0+1} \ge \mu_{t_0+1}]. && (24)
\end{aligned}$$

Using the second part of Lemma 1 (Paley-Zygmund)

$$\begin{aligned}
\mathbb{P}[\bar{Z}_{t_0+1} \ge \mu_{t_0+1}] &\ge \frac{1}{1 + \frac{(\gamma+1)^2}{\gamma^2} \frac{\epsilon^2}{6} \frac{V_{t_0+1}(\mu)}{V_{t_0+1}(\mu_{t_0+1})}} && (25)\\
&\ge \left(1 + \frac{(\gamma+1)^2}{\gamma^2} \frac{\epsilon^2}{6}\right)^{-1}. && (26)
\end{aligned}$$

Therefore, $\mathbb{P}[A_1] \le 1 - \left(1 + \frac{(\gamma+1)^2}{\gamma^2} \frac{\epsilon^2}{6}\right)^{-1} \le \frac{(\gamma+1)^2}{\gamma^2} \frac{\epsilon^2}{6}$. Finally, let $t^*$ be the (random) level where the algorithm terminates and $A_2$ be the event that $|\bar{Z}_{t^*} - \mu| > \epsilon\mu$. If any of the three events happen we say that the procedure fails. We can bound the failure probability by:

$$\begin{aligned}
\mathbb{P}[F] &= \mathbb{P}[A_0 \vee A_1 \vee A_2]\\
&= \mathbb{P}[A_0 \vee A_1 \vee A_2 \wedge A_0] + \mathbb{P}[(A_0 \vee A_1 \vee A_2) \wedge A_0^c]\\
&\le \mathbb{P}[A_0] + \mathbb{P}[A_1 \wedge A_0^c] + \mathbb{P}[A_2 \wedge A_0^c]. && (27)
\end{aligned}$$

To bound the last term we use:

$$\begin{aligned}
\mathbb{P}[A_2 \wedge A_0^c] &= \mathbb{P}[A_2 \wedge A_0^c \wedge A_1] + \mathbb{P}[A_2 \wedge A_0^c \wedge A_1^c]\\
&\le \mathbb{P}[A_1] + \mathbb{P}[A_2 \wedge A_0^c \wedge A_1^c].
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{P}[A_2 \wedge A_0^c \wedge A_1^c] &= \sum_{t \in \{t_0, t_0+1\}} \mathbb{P}[|\bar{Z}_t - \mu| > \epsilon\mu \wedge t^* = t]\\
&\le \sum_{t \in \{t_0, t_0+1\}} \mathbb{P}[|\bar{Z}_t - \mu| > \epsilon\mu]\\
&\le \frac{1}{\epsilon^2} \sum_{t \in \{t_0, t_0+1\}} \mathrm{RelVar}[\bar{Z}_t]\\
&\le \frac{1}{\epsilon^2} \sum_{t \in \{t_0, t_0+1\}} \frac{\epsilon^2}{6} \frac{V_t(\mu)}{V_t(\mu_{t+1})}.
\end{aligned}$$

By definition $\mu \ge \mu_{t+1}$ for all $t \ge t_0$, thus by (B) and (28):

$$\mathbb{P}[A_2 \wedge A_0^c \wedge A_1^c] \le \frac{2}{6} = \frac{1}{3}. \tag{28}$$

Hence, the overall probability failure is bounded by:

$$\begin{aligned}
\mathbb{P}[F] &\le \mathbb{P}[A_0] + 2\mathbb{P}[A_1] + \mathbb{P}[A_2 \wedge A_0^c \wedge A_1^c]\\
&\le \frac{\epsilon^2}{6} \frac{1}{\gamma^2} \mu_{t_0}^{\alpha} \frac{1}{1-(1+\gamma)^{-\alpha}} + 2\frac{(\gamma+1)^2}{\gamma^2} \frac{\epsilon^2}{6} + \frac{1}{3}.
\end{aligned}$$

When the algorithm succeeds the total number of samples is bounded by

$$\begin{aligned}
\sum_{t=1}^{t_0+1} \lceil \frac{6}{\epsilon^2} V_t(\mu_{t+1}) \rceil &\le (t_0+1) + \frac{6C}{\epsilon^2} \sum_{t=1}^{t_0+1} (1+\gamma)^{\beta(t+1)}\\
&\le (t_0+1) + \frac{6C}{\epsilon^2} (1+\gamma)^{2\beta} \frac{(1+\gamma)^{\beta t_0}}{\gamma}\\
&\le (t_0+1) + \frac{6C}{\epsilon^2} \frac{(1+\gamma)^{\beta}}{\gamma} \frac{1}{\mu^{\beta}}.
\end{aligned}$$

**Case II** ($t_0 \ge T$)**.** In this case $\mu \le \mu_T \le \frac{1}{1+\gamma}\epsilon\tau$. By the same arguments as in the case $t_0 < T$ we get that the probability terminates before $t < t_0$ is at most $\frac{\epsilon^2}{6\gamma^2} \mu_{t_0}^{\alpha} \frac{1}{1-(1+\gamma)^{-\alpha}}$. If the condition $\bar{Z}_T \ge \mu_T$ is satisfied then:

$$\mathbb{P}[|\bar{Z}_T - \mu| > \epsilon\mu] \le \frac{1}{\epsilon^2} \mathrm{RelVar}[\bar{Z}_T] \le \frac{1}{6} \tag{29}$$

If $\bar{Z}_T < \mu_T$ then:

$$|0 - \mu| \le \mu \le \mu_T \le \frac{1}{1+\gamma}\epsilon\tau \le \epsilon \max\{\mu, \tau\} \tag{30}$$

**Conclusion.** Thus, overall if $\hat{Z}$ is the output of AMR:

$$\mathbb{P}[|\hat{Z}-\mu|>\epsilon\max\{\mu,\tau\}] \le \frac{\epsilon^2}{6}\frac{1}{\gamma^2}\mu_{t_0}^{\alpha}\frac{1}{1-(1+\gamma)^{-\alpha}} + 2\frac{(\gamma+1)^2}{\gamma^2}\frac{\epsilon^2}{6}+\frac{1}{3}$$

As we see in the above expression the failure probability is dominated by the $\frac{1}{3}$ term. For example for $\gamma=1,\epsilon=0.2$, $\alpha=1$ we have that the extra term is less than $0.0667$. □

### 1.6. Regular estimator for Gaussian Kernel

**Theorem 3.** *$\mathcal{Z}_{\text{Gauss}}$ is $(1,\frac{3}{4},\gamma)$-regular and takes preprocessing time/space bounded by $O_{d,\kappa_T,\gamma}(\epsilon^{-3+\frac{1}{4}}\tau^{-\frac{3}{4}}\cdot n)$.*

*Proof of Theorem 3.* By Lemma 3 and Theorem 1 (Section 2.3 in main paper), (A) holds with $V_t(\mu) := \frac{4e^{\frac{3}{2}}}{\mu}e^{r_t^2-r_t\sqrt{\log(\frac{1}{\mu})}}$. Moreover, since $\forall x\ge y>0$

$$\frac{V_t(y)}{V_t(x)} = \frac{x}{y}e^{-r_t(\sqrt{\log(\frac{1}{y})}-\sqrt{\log(\frac{1}{x})})} \le \left(\frac{x}{y}\right)^{2-1} \tag{31}$$

and $V_t'(x) = -\frac{4e^{\frac{3}{2}}}{x}e^{r_t^2-r_t\sqrt{\log(\frac{1}{\mu})}}(\frac{1}{x}+\frac{r_t}{2\sqrt{\log(\frac{1}{x})}}) < 0$, property (B) holds with $\alpha=1$. Finally,

$$V_t(\mu_{t+1}) = 4e^{\frac{3}{2}}e^{\{\frac{1}{4}-\frac{1}{2}\sqrt{\frac{t+1}{t}}+(1+\frac{1}{t})\}t\log(1+\gamma)} \tag{32}$$

$$= 4e^{\frac{3}{2}}\left(\frac{1}{\mu_t}\right)^{\frac{1}{4}-\frac{1}{2}\sqrt{\frac{t+1}{t}}+(1+\frac{1}{t})} \tag{33}$$

$$\le 4e^{\frac{3}{2}}(1+\gamma)^{1-\frac{1}{\sqrt{2}}}\cdot\left(\frac{1}{\mu_t}\right)^{\frac{3}{4}}, \tag{34}$$

and consequently (C) holds with $\beta=\frac{3}{4}$. Finally, the estimator uses at most $O(\frac{1}{\epsilon^2}V_T(\mu_{T+1}))$ hash tables each taking preprocessing time/space $O_{d,q_T,\gamma}(n)$ space. □

## 2. Sketching

For any hash table $H$ and a vector $u\in\Delta_n$ (simplex), let $B=B(H)$ denote the number of buckets and $u_{\max}=u_{\max}(H):=\max\{u_{H_i}:i\in[B]\}$ the maximum weight of any hash bucket of $H$. The precise definition of our Hashing-Based-Sketch is given below in Algorithm 1.

For a fixed $H$, we can obtain the following bounds on the first two moments of our sketch $(S_m,w)$.

**Lemma 5** (Moments)**.** *For the sketch $(S_m,w)$ produced by the HBS procedure it holds that*

$$\mathbb{E}[\text{KDE}_{S_m}^w|H] = \text{KDE}_P^u(q),$$

$$\text{Var}[(\text{KDE}_{S_m}^w)^2|H] \le \frac{1}{m}(Bu_{\max})^{1-\gamma^*}\sum_{i=1}^n k^2(x_i,q)u_i.$$

**Algorithm 1** Hashing-Based-Sketch (HBS)

**Input:** set $P$, size $m$, hashing scheme $\mathcal{H}_\nu$, threshold $\tau\in(0,1)$, $u\in\Delta_n$
Sample $h\sim\mathcal{H}_\nu$ and create hash tabel $H=h(P)$.
Set $\gamma$ according to (35)
$S_m\leftarrow\emptyset$, $w\leftarrow 0\cdot\mathbf{1}_m$, $B\leftarrow B(H)$
**for** $j=1,\dots,m$ **do**
  Sample hash bucket $H_i$ with probability $\propto u_{H_i}^\gamma$
  Sample a point $X_j$ from $H_i$ with probability $\propto u_j$
  $S_m\leftarrow S_m\cup\{X_j\}$
  $w_j(\gamma,m)\leftarrow\frac{u_{H_i}}{m}\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_i}^\gamma}$
**Output:** $(S_m,w)$

The above analysis shows that the sketch is always unbiased and that the variance depends on the hash function $H$ only through $(Bu_{\max})^{1-\gamma^*}\ge 1$. We postpone the proof of this lemma after showing how it implies the following theorem.

**Theorem 4.** *Let $H$ be the hash function sampled by the HBS procedure. For $\epsilon>0$ and $\delta\in[e^{-\frac{6}{\epsilon^2}\frac{u_{\max}}{n\tau}},e^{-\frac{6}{\epsilon^2}})$, let:*

$$\gamma^* = \left\{1-\frac{\log(\frac{\epsilon^2}{6}\log(1/\delta))}{\log(\frac{u_{\max}}{\tau})}\right\}^{\mathbb{I}[B\le(\frac{1}{2})^{\frac{1}{6}}\frac{1}{\tau}]}, \tag{35}$$

$$m = \frac{6}{\epsilon^2}\frac{1}{\tau}(Bu_{\max})^{1-\gamma^*} < \frac{\log(\frac{1}{\delta})}{\tau}. \tag{36}$$

*Then $(S_m,w)$ is an $(\epsilon,\frac{1}{6},\tau)$-sketch and if $B\le(\frac{1}{2})^{\frac{1}{6}}\frac{1}{\tau}$ any hash bucket with weight at least $\tau$ will have non empty intersection with $S_m$ with probability at least $1-\delta$.*

*Proof of Theorem 4.* Given a hash bucket with weight at least $\tau$, the probability that we sample a point from that bucket is at least:

$$\rho \ge \frac{\tau^\gamma}{B^{1-\gamma}} = \tau\frac{1}{(B\tau)^{1-\gamma}} \tag{37}$$

The probability that we see no point after $m$ independent samples is less than $(1-\rho)^m \le e^{-m\frac{\tau^\gamma}{B^{1-\gamma}}}$ For $m\ge\frac{\log(1/\delta)}{\tau}(B\tau)^{1-\gamma}$ this probability is at most $\delta$. On the other hand by Lemma 5 if $m\ge\frac{6}{\epsilon^2}\frac{1}{\tau}(Bu_{\max})^{1-\gamma}$ we have that $\text{Var}[\text{KDE}_{S_m}^w]\le\frac{\epsilon^2}{6}\mu\tau$. The case $B>2^{-\frac{1}{6}}\frac{1}{\tau}$ is trivial as $\gamma^*=1$. For $B\le 2^{-\frac{1}{6}}\frac{1}{\tau}\Rightarrow u_{\max}\ge\frac{1}{B}\ge\tau 2^{\frac{1}{6}}$. We set $\gamma$ to make the two lower bounds on $m$ equal,

$$\frac{6}{\epsilon^2}\frac{1}{\tau}(Bu_{\max})^{1-\gamma} = \frac{\log(1/\delta)}{\tau}(B\tau)^{1-\gamma} \tag{38}$$

$$\Leftrightarrow \left(\frac{u_{\max}}{\tau}\right)^{1-\gamma} = \frac{\epsilon^2\log(1/\delta)}{6} \tag{39}$$

$$\Leftrightarrow \gamma = 1-\frac{\log(\frac{\epsilon^2}{6}\log(1/\delta))}{\log(\frac{u_{\max}}{\tau})}. \tag{40}$$

This is strictly less than one for $\log(1/\delta)\frac{\epsilon^2}{6} > 1 \Rightarrow \delta < e^{-\frac{6}{\epsilon^2}}$, and more than zero for $\delta \geq e^{-\frac{6}{\epsilon^2}\frac{u_{\max}}{\tau}}$. Since $u_{\max} \geq \tau 2^{1/6}$ the two inequalities are consistent. Furthermore,

$$
\begin{aligned}
m &= \frac{6}{\tau\epsilon^2} \cdot (Bu_{\max})^{1-\gamma^*} && (41)\\
&= \frac{6}{\tau\epsilon^2} \cdot (Bu_{\max})^{\log(\frac{\epsilon^2\log(1/\delta)}{6})\frac{1}{\log(\frac{u_{\max}}{\tau})}} && (42)\\
&= \frac{6}{\tau\epsilon^2} \cdot e^{\log(\log(1/\delta)\frac{\epsilon^2}{6})\frac{\log(Bu_{\max})}{\log(\frac{u_{\max}}{\tau})}} && (43)\\
&\leq \frac{6}{\tau\epsilon^2} \cdot \left(\log(1/\delta)\frac{\epsilon^2}{6}\right)^{(1-\frac{1}{6}\frac{\log 2}{\log(\frac{u_{\max}}{\tau})})} && (44)\\
&< \frac{\log(1/\delta)}{\tau}. && (45)
\end{aligned}
$$

□

**Remark 1.** *Observe that $\frac{\log\frac{1}{\delta}}{\tau}$ is the number of samples that random sampling would require in order to have the same property for any bucket with $u_{H_i} \geq \tau$. When $\gamma^* < 1$, our scheme always uses less samples by a factor of $\left(\log(1/\delta)\frac{\epsilon^2}{6}\right)^{\frac{\log(B\tau)}{\log(\frac{u_{\max}}{\tau})}} < 1$.*

Thus, our sketch will have similar variance with random sampling in dense regions of the space but will have better performance for relatively "sparse" regions.

### 2.1. Proof of Lemma 5

*Proof of Lemma 5.* Let $I$ be the random hash bucket and $X_I$ the corresponding random point, then for a single point:

$$
\begin{aligned}
\mathbb{E}[\mathrm{KDE}^{w_1}_{\{X_I\}}] &= \mathbb{E}_I[\mathbb{E}_{X_I}[\frac{u_{H_I}}{m}\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_I}^\gamma}k(X_I,q)]]\\
&= \mathbb{E}_I[\sum_{j\in H_I}\frac{u_{H_I}}{m}\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_I}^\gamma}k(x_j,q)\frac{u_j}{u_{H_I}}]\\
&= \frac{1}{m}\mathbb{E}_I[\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_I}^\gamma}\sum_{j\in H_I}k(x_j,q)u_j]\\
&= \frac{1}{m}\sum_{i\in[B]}\sum_{j\in H_i}k(x_j,q)u_j\\
&= \frac{1}{m}\mathrm{KDF}^u_P(q).
\end{aligned}
$$

The first part follows by linearity of expectation. Similarly,

$$
\mathbb{E}[(\mathrm{KDF}^w_{S_m})^2] \leq \sum_{j=1}^m \mathbb{E}[(\mathrm{KDF}^{w_j}_{\{x_j\}})^2] + (\mathrm{KDF}^u_P(q))^2.
$$

By linearity we only have to bound the first term

$$
\begin{aligned}
\mathbb{E}[(\mathrm{KDE}^{w_1}_{\{X_I\}})^2] &= \mathbb{E}_I[\mathbb{E}_{X_I}[(\frac{u_{H_I}}{m}\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_I}^\gamma}k(X_I,q))^2]]\\
&= \mathbb{E}_I[\sum_{j\in H_I}(\frac{u_{H_I}}{m}\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{u_{H_I}^\gamma}k(x_j,q))^2\frac{u_j}{u_{H_I}}]\\
&= \mathbb{E}_I[(\frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{mu_{H_I}^\gamma})^2 u_{H_I}\sum_{j\in H_I}k^2(x_j,q)u_j]\\
&= \frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{m^2}\sum_{i\in[B]}u_{H_I}^{1-\gamma}\sum_{j\in H_I}k^2(x_j,q)u_j\\
&\leq \frac{\sum_{i'=1}^B u_{H_{i'}}^\gamma}{m^2}u_{\max}^{1-\gamma}\sum_{i\in[B]}\sum_{j\in H_I}k^2(x_j,q)u_j\\
&\leq \frac{(Bu_{\max})^{1-\gamma}}{m^2}\sum_{j=1}^n k^2(x_j,q)u_j.
\end{aligned}
$$

The last inequality follows by applying Hölder's inequality with $p = \frac{1}{\gamma}$ and $q = \frac{1}{1-\gamma}$, and due to $u \in \Delta_n$. □

## 3. Diagnostic and Visualization procedures

In this section, we show how our refined variance bounds along with the adaptive procedure lead to a diagnostic procedure estimating the variance of RS and HBE, as well as to a visualization procedure that gives intuition about the "local structure" of the queries in a given dataset.

### 3.1. Diagnostic procedure

In order to go beyond worst-case bounds (typically expressed as a function of the query density $\mu$) and provide dataset specific bounds on the variance of different methods (RS and HBE) we use Lemma 4. By setting the coefficients $V_{ij}$ appropriately, we can bound the variance of RS ($V_{ij} = 1$) and HBE ($V_{ij} = \frac{\min\{p(q,x_i),p(q,x_j)\}}{p(q,x_i)^2}$). Unfortunately, evaluating the bound (11) directly over the whole dataset for a single query is no cheaper than evaluating the methods directly.

At a high level, our diagnostic procedure goes around this by evaluating the upper bound for each query $q$ on 'representative sample" $\tilde{S}_0(q)$ instead of $P$ . By doing this for a number $T$ of random queries picked uniformly from the dataset $P$, we get an estimate of the average relative variance for different methods.

Specifically, given $\tau \in (0,1)$ and $\epsilon \in (0,1)$, for a single query let $\tilde{S}_0$ be the random set produced by the AMR procedure (Algorithm 1, Section 4.2) called with random sampling and define the sets $\tilde{S}_\ell = \tilde{S}_0 \cap S_\ell$ for $\ell \in [4]$ and

their corresponding "densities" $\tilde{\mu}_\ell = \sum_{i\in\tilde{S}_\ell} u_i w_i$. Let

$$\lambda_\epsilon := \arg\max_{\tilde{\mu}_0 \le \lambda \le 1} \left\{ \tilde{\mu}_3 \le \frac{1}{2}(\epsilon\tilde{\mu}_0 - \tilde{\mu}_4) \right\} \tag{46}$$

$$L_\epsilon := \arg\min_{\tilde{\mu}_0 \le L \le 1} \left\{ \tilde{\mu}_1 \le \frac{1}{2}(\epsilon\tilde{\mu}_0 - \tilde{\mu}_4) \right\} \tag{47}$$

be such that $\tilde{\mu}_2 \ge (1-\epsilon)\tilde{\mu}_0$, i.e. most of the mass is captured by the set $\tilde{S}_2$ (that is an spherical annulus for kernels that are decreasing with distance). Since Lemma 4 holds for all $\mu \le \lambda \le L \le 1$, $L_\epsilon, \lambda_\epsilon$ complete the definition of four sets $\tilde{S}_1, \ldots, \tilde{S}_4$ which we use to evaluate (11), and denote by $V_{\text{method}}(q)$ the corresponding bound used with $V_{ij}$ corresponding to a certain estimator, e.g. $\text{method} \in \{\text{RS}, \text{HBE}\}$. Below we give the procedure in pseudo-code.

**Algorithm 2** Diagnostic

**Input:** set $P$, threshold $\tau \in (\frac{1}{n}, 1)$, accuracy $\epsilon$, $T \ge 1$, collision probability $p(x,y)$ of the hashing scheme $\mathcal{H}_\nu$.
**for** $t = 1, \ldots, T$ **do**
  $q \leftarrow \text{Random}(P)$ ▷ For each random query
  $(\tilde{S}_0, \tilde{\mu}_0) \leftarrow \text{AMR}(\mathcal{Z}_{RS}(q), \epsilon, \tau)$
  Set $\lambda_\epsilon, L_\epsilon$ using (46) and (47)
  Let $V_{\text{RS}}$ be the r.h.s of (11) for $\tilde{S}_0$ and $V_{ij} = 1$.
  Let $V_{\text{HBE}}$ be the r.h.s of (11) for $\tilde{S}_0$ and

$$V_{ij} = \frac{\min\{p(q,x_i), p(q,x_j)\}}{p(q,x_i)^2} \tag{48}$$

  $\text{rV}_{\text{RS}}(t) \leftarrow V_{\text{RS}}/\max\{\hat{\mu}, \tau\}^2$
  $\text{rV}_{\text{HBE}}(t) \leftarrow V_{\text{HBE}}/\max\{\hat{\mu}, \tau\}^2$.
**Output:** $(\text{mean}_T(rV_{\text{RS}}), \text{mean}_T(rV_{\text{HBE}}))$

**Remark 2.** *We only show the procedure for choosing between RS and HBE with a specific hashing scheme. The same procedure can be used to evaluate a multitude of hashing schemes to select the best one for a given dataset.*

### 3.2. Visualization procedure

We can use the information from our diagnostics to visualize what is the data set like by aggregating local information for random queries. For a set $S$, let $r_S = \min_{i\in S} \log(\frac{1}{k(x_i,q)})$ and $R_S = \max_{j\in S} \log(\frac{1}{k(x_j,q)})$. The basis of our visualization is the following fact:

**Lemma 6.** *Let $X$ be a random sample from $S$, then $E[k^2(X,q)] \le \exp(R_S - r_S) \cdot \mu_S^2$.*

*Proof of Lemma 6.* We have that $e^{-R_S} \le k(x_i,q) \le e^{-r_S}$, therefore

$$\mathbb{E}[k^2(X,q)] = \sum_{i\in S} k^2(x_i,q)u_i \tag{49}$$

$$\le e^{-r_S} \sum_{i\in S} k(x_i,q)u_i \frac{\mu_S}{\mu_S} \tag{50}$$

$$\le e^{R_S - r_S}\mu_S^2 \tag{51}$$

where in the last part we used $\mu_S \ge e^{-R_S}$. □

Thus if we plot an annulus of width $w_S = R_S - r_s$ then $e^{w_S}$ is an estimate of the relative variance for RS! The visualization procedure when given a sequence of $T$ pairs of numbers $(\lambda_t, L_t)$ for $t \in [T]$ (produced by the diagnostic procedure) plots overlapping annuli around the origin representing the queries. Since often the ratio $\max_{i,j\in S} \frac{k(x_i,q)}{k(x_j,q)}$ is refered to as the *condition number* of the set $S$, we call our procedure the Log-Condition plot.

**Algorithm 3** Log-Condition Plot

**Input:** $\{(\lambda_t, L_t)\}_{t\in[T]}$.
**for** $t = 1, \ldots, T$ **do** ▷ For each query
  $r_t \leftarrow \log(1/L_t)$,
  $R_t \leftarrow \log(1/\lambda_t)$
  draw 2D-annulus$(r_t, R_t)$
**Output:** figure with overlapping annuli.

**Remark 3.** *In the specific case of the Laplace (exponential) kernel, the radii we are plotting correspond to actual distances.*

## 4. Synthetic benchmarks

In this section, we introduce a general procedure to create tunable synthetic datasets that exhibit different local structure around the query. We then show how to use this procedure as a building block to create two different family of instances with specific characteristics aimed to test kernel density evaluation methods.

### 4.1. $(\mu, D, n, s, d, \sigma)$-Instance

Since the problem of kernel density is query dependent and the kernel typically depends only on the distance, we shall always assume that the query point is at the origin $q = 0 \in \mathbb{R}^d$.

We further assume that the kernel is an invertible function of the distance $K(r) \in [0,1]$ and let $K^{-1}(\mu) \in [0,\infty)$ be the inverse function. For example, the exponential kernel is given by $K(r) = e^{-r}$ and the inverse function is given by $K^{-1}(\mu) = \log(\frac{1}{\mu})$.

The dataset is created with points lying in $D$ different directions and $s$ distance scales (equally spaced between 0 and

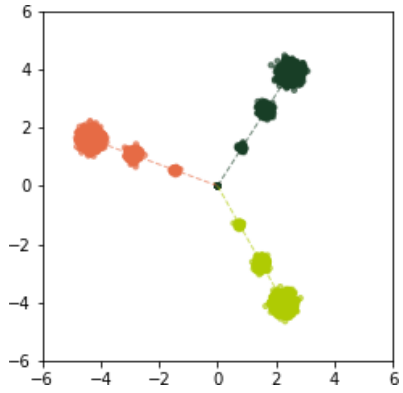


*Figure 2.* ($\mu = 0.01$, $D = 3$, $s = 4$, $d = 2$, $\sigma = 0.05$)-Instance. Each of the $D = 3$ directions is coded with a different color.

$R = K^{-1}(\mu)$) such that the *contribution from each direction and scale* to the kernel density at the origin is *equal*. To achieve this the number of points $n_j$ placed a at the $j$-th distance scale $r_j$ is given by

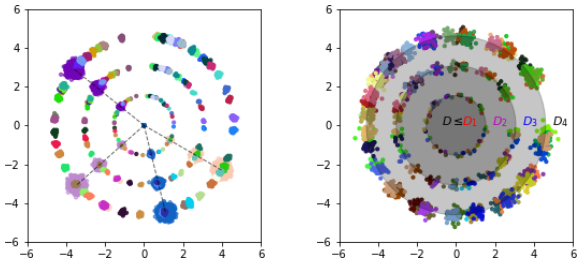
$$n_\ell := \lfloor n \frac{\mu}{K(r_j)} \rfloor. \tag{52}$$

The reasoning behind this design choice is to make sure that we have diversity in the distance scales that matter in the problem, so not to favor a particular class of methods (e.g. random sampling , nearest-neighbor based). Also, placing the points on the same direction makes the instance more difficult for HBE as the variance in (6) increases with the ratio $\frac{\mathbb{P}[h(i)=h(j)=h(q)]}{\mathbb{P}[h(i)=h(q)]^2}$, that expresses how correlated the values $\{h(i), h(j), h(q)\}$ are. We give an example visualization of such data sets in 2 dimensions in Figure 2. The detailed procedure is described below (Algorithm 4).

**Algorithm 4** $(\mu, D, n, s, d, \sigma)$-Instance

**Input:** $\mu \in [\frac{1}{n}, 1]$, $D \geq 1$, $n \geq 1$, $s \geq 2$ , $d \geq 1$, $\sigma \geq 0$, kernel $K$, inverse $K^{-1}$.
$R \leftarrow K^{-1}(\mu)$, $r_0 \leftarrow K^{-1}(1)$, $P \leftarrow \emptyset$.
**for** $j = 0, \ldots, s-1$ **do**
  $r_{j+1} \leftarrow \frac{R-r_0}{s-1} j + r_0$ ▷ distances for each D
  $n_{j+1} \leftarrow \lfloor n \frac{\mu}{K(r_{j+1})} \rfloor$ ▷ points at each distance
**for** $i = 1, \ldots, D$ **do**
  $v_i \leftarrow \frac{g_i}{\|g_i\|}$ with $g_i \sim \mathcal{N}(0, I_d)$. ▷ random direction
  **for** j=1,..., s **do** ▷ For each distance scale
    **for** $\ell = 1, \ldots, n_j$ **do** ▷ generate a "cluster"
      $g_{ij\ell} \sim \mathcal{N}(0, I_d)$
      $x_{ij\ell} \leftarrow s_j v_i + \frac{\sigma}{\sqrt{d}} s_j g_{ij\ell}$
      $P \leftarrow P \cup \{x_{ij\ell}\}$
**Output:** Set of points $P$

**Remark 4.** *If $D \ll n$ this class of instances becomes highly structured with a small number of tightly knit "clusters" (Figure 2). One would expect in this case, space-partioning methods to perform well. At the same time by Lemma 4, this type of instances are the ones that maximize the variance of both HBE $(s \geq 1)$ and RS $(s > 1)$ .*




(a) "worst-case" instance (b) $D$-structured instance

*Figure 3.* The two family of instances for $d = 2$.

**Remark 5.** *On the other hand if $D \gg n$ the instances become spread out (especially in high dimensions). This type of instances are ideal for sampling based methods when $s = 1$, and difficult for space-partitioning methods.*

Based on the above remarks we propose the following subclass of instances.

### 4.2. "Worst-case" instance

In order to create an instance that is *hard for all methods* we take a union of the two extremes $D \ll n$ and $D \gg n$. We call such instances "worst-case" as there does not seem to be a single type of structure that one can exploit, and these type of instances realize the worst-case variance bounds for both HBE and RS. In particular, if we want to generate an instance with $N$ points, we first set $D$ a small constant and $n = \Theta(N)$ and take the union of such a dataset with another using $D = \Omega(N^{1-o(1)})$ and $n = O(N^{o(1)})$. An example of such a dataset is given in 3(a).

### 4.3. $D$-structured instance

"Worst-case" instances are aimed to be difficult for any kernel evaluation method. In order to create instances that have more varied structure, we use our basic method to create a single parameter family of instance by fixing $N, \mu, \sigma, s, d$ and setting $n = \frac{N}{D}$. We call this family of instances as *$D$-structured*. As one increases $D$, two things happen:

- The number of directions (clusters) increases.
- $n = \frac{N}{D}$ decreases and hence certain distance scales disappear. By (52), if $n\mu < K(r_j) \Rightarrow D_j > \frac{N\mu}{K(r_j)}$ then distance scale $j$ will have no points assigned to it.

Hence, for this family when $D \ll \frac{N}{D} \leftrightarrow D \ll \sqrt{N}$ the instances are highly structured and we expect space-

*Table 1.* Preprocessing time ($init$) and total query time ($query$) on 10K random queries for additional datasets. All runtime measurements are reported in seconds.

| Dataset | Time | RS | HBE | ASKIT | FigTree |
|---|---|---|---|---|---|
| higgs | $init$ | 0 | 141 | 25505 | $>$ 1day |
| | $query$ | **6** | 18 | 1966 | $>$ 1day |
| hep | $init$ | 0 | 138 | 23421 | $>$ 20 hours |
| | $query$ | **6** | 11 | 1581 | $>$ 1day |
| susy | $init$ | 0 | 67 | 5326 | 3245 |
| | $query$ | 18 | **12** | $>$ 9756 | 5392 |
| home | $init$ | 0 | 11 | 237 | 7 |
| | $query$ | 2369 | **17** | 376 | 33 |
| mnist | $init$ | 0 | 211 | 14 | 437 |
| | $query$ | **168** | 389 | ? | 1823 |

*Table 2.* Preprocessing time (in seconds) for clustering test.

| $n$ | $D$ | HBE | FigTree | ASKIT |
|---|---|---|---|---|
| $500K$ | 1 | 192 | 2 | 113 |
| $50K$ | 10 | 20 | 3 | 105 |
| $5K$ | 100 | 16 | 16 | 105 |
| 500 | 1000 | 19 | 174 | 104 |
| 50 | 10000 | 39 | 1516 | 102 |
| 5 | 100000 | 334 | 0.3 | 101 |

partitioning methods to perform well. On the other extreme as $D$ increases and different distance scales start to die out (Figure 3(b)) the performance of random sampling keeps improving until there is only one (the outer) distance scale, where random sampling will be extremely efficient. On the other hand HBE's will have roughly similar performance on the two extremes as both correspond to worst-case datasets for scale-free estimators with $\beta = 1/2$, and will show slight improvement in between $1 \ll D \ll n$. This picture is confirmed by our experiments.

## 5. Experiments

### 5.1. Datasets

We provide detailed descriptions of the datasets as well as the bandwidth used for the kernel density evaluation in Table 3. We also include specifications for the additional datasets acquired from LIBSVM (Chang & Lin, 2011) and the UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017) that were used to evaluate the accuracy of the diagnostic procedure.

We selected the top eight datasets in Table 3 for density evaluation in the main paper as they are the largest, most complex datasets in our collection. We provide additional density evaluation results in Table 1 for datasets with comparable sizes or dimensions to the ones reported in the main paper. For higgs and hep, FigTree failed to finish the evaluation within a day. Given the performance of RS on these datasets, we don't expect FigTree to achieve better performance even if the query returns successfully. For mnist, we were not able to get ASKIT to achieve relative error below 1 even after trying parameters that span a few orders of magnitude; this is potentially caused by the high-dimensionality and sparsity of this dataset.

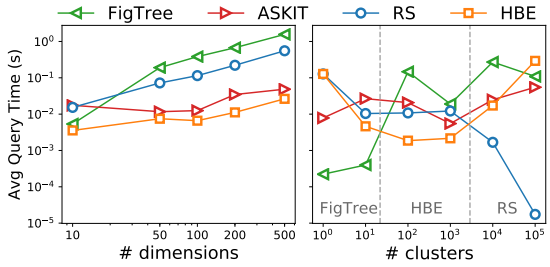



*Figure 4.* Results from the synthetic experiment (repeat of results in the main paper for easy reference).

### 5.2. Synthetic Experiment

For the clustering test, we set $\mu = 0.001, s = 4, d = 100, \sigma = 0.01, N = 500K$. The varying parameters are the number of clusters ($D$) and the number of points per cluster $n$. We report preprocessing time (in seconds) for all methods in Table 2. The ordering of methods according to preprocessing time largely follows the that of query time.

As discussed in Section 4.3, for the $D$-structured instances as the number of points per cluster $n$ decreases, smaller distance scales start to disappear due to (52). Let $D_i$ be the threshold such that for $D > D_i$, there are no-points in scale $i$. The corresponding numbers for our experiment is roughly $D1 = 500$ (at distance 0) , $D2 = 1077$, $D3 = 10K$, $D4 = N = 500K$. In particular, only a single distance scale remains when $D = 100K > D3$, a set up in which RS is orders of magnitude more efficient than alternative methods (Figure 4 right).

### 5.3. Sketching Experiment

In this section, we evaluate the quality of the proposed hashing-based sketch. As baselines, we compare against sparse kernel approximation (SKA) (Cortes & Scott, 2017), kernel herding algorithm (Herding) (Chen et al., 2010) and uniform sampling. To control for the difference in the complexity (Table 4), we compare the approximation error achieved by sketches of the same size ($s$) under the same compute budget ($2n$, where $n$ is dataset size). We describe the detailed setup below, including necessary modifications to meet the computational constraints.

*Table 3.* Specifications of real-world datasets.

| Dataset | N | d | $\sigma$ | Description |
|---|---|---|---|---|
| census | $2.5M$ | 68 | 3.46 | Samples from 1900 US census. |
| TMY3 | $1.8M$ | 8 | 0.43 | Hourly energy load profiles for US references buildings. |
| TIMIT | $1M$ | 440 | 10.97 | Speech data for acoustic-phonetic studies. First 1M data points used. |
| SVHN | $630K$ | 3072 | 28.16 | Google Street View house numbers. Raw pixel values of 32x32 images. |
| covertype | $581K$ | 54 | 2.25 | Cartographic variables for predicting forest cover type. |
| MSD | $463K$ | 90 | 4.92 | Audio features of popular songs. |
| GloVe | $400K$ | 100 | 4.99 | Pre-trained word vectors from Wikipedia 2014 + Giga 5 word. 5. 6B tokens, 400K vocab. |
| ALOI | $108K$ | 128 | 3.89 | Color image collection of 1000 small objects. Each image is represented by a 128 dimensional SIFT feature vector. |
| higgs | $11M$ | 28 | 3.41 | Signatures of Higgs bosons from Monte Carlo simulations. |
| hep | $10.5M$ | 27 | 3.36 | Signatures of high energy physics particles (Monte Carlo simulations). |
| susy | $5M$ | 18 | 2.24 | Signatures of supersymmetric particles (Monte Carlo simulations). |
| home | $969K$ | 10 | 0.53 | Home gas sensor measurements. |
| skin | $245K$ | 3 | 0.24 | Skin Segmentation dataset. |
| ijcnn | $142K$ | 22 | 0.90 | IJCNN 2001 Neural Network Competition. |
| acoustic | $79K$ | 50 | 1.15 | Vehicle classification in distributed sensor networks. |
| mnist | $70K$ | 784 | 11.15 | 28x28 images of handwritten digits. |
| corel | $68K$ | 32 | 1.04 | Image dataset, with color histograms as features. |
| sensorless | $59K$ | 48 | 2.29 | Dataset for sensorless drive diagnosis. |
| codrna | $59K$ | 8 | 1.13 | Detection of non-coding RNAs. |
| shuttle | $44K$ | 9 | 0.62 | Space shuttle flight sensors. |
| poker | $25K$ | 10 | 1.37 | Poker hand dataset. |
| cadata | $21K$ | 8 | 0.62 | California housing prices. |

*Table 4.* Overview of algorithm complexity and parameter choice for the sketching experiment ($n$: dataset size, $s$: sketch size, $T$: number of hash tables, $m$: sample size for herding).

| Algorithm | Complexity | Parameters |
|---|---|---|
| HBS | $O(n'T + s)$ | $n' = \frac{2n}{5}, T = 5$ |
| SKA | $O(n_c s_c + s_c^3)$ | $s_c = n^{\frac{1}{3}}, n_c = n^{\frac{2}{3}}$ |
| Herding | $O(n_h m + n_h s)$ | $m = s, n_h = \frac{n}{s}$ |

**HBS.** For HBS, we used 5 hash tables, each hashing a subset of $\frac{2}{5}n$ points in the dataset. In practice, we found that varying this small constant on the number of hash tables does not have a noticeable impact on the performance.

**SKA.** SKA (Algorithm 5) produces the sketch by greedily finding $s$ points in the dataset that minimizes the maximum distance. The associated weights are given by solving an equation that involves the kernel matrix of the selected points. SKA's complexity $O(ns + s^3)$ is dominated by the matrix inversion procedure used to solve the kernel matrix equation. To ensure that SKA is able to match the sketch size of alternative methods under the compute budget of $2n$, we augment SKA with random samples when necessary:

- If the target sketch size is smaller than $n^{\frac{1}{3}}$ ($s < n^{\frac{1}{3}}$), we use SKA to produce a sketch of size $s$ from a subsample of $n/s$ data points.
- For $s > n^{\frac{1}{3}}$, we use SKA to produce a sketch of size $s_c = n^{\frac{1}{3}}$ from a subsample of $n_c = n^{\frac{2}{3}}$ data points. We match the difference in sketch size by taking an additional $s - s_c$ random samples from the remaining $n - n_c$ data points that were not used for the SKA sketch. The final estimate is a weighted average between the SKA sketch and the uniform sketch: $\frac{1}{s_c}$ for SKA and $(1 - \frac{1}{s_c})$ for uniform, where the weights are determined by the size of the two sketches.

The modification uses SKA as a form of regularization on random samples. Since SKA iteratively selects points that are farthest away from the current set, the resulting sketch is helpful in predicting the "sparser" regions of the space. These sparser regions, in turn, are the ones that survive in the $n^{2/3}$ random sample of the dataset (sub-sampling with probability $n^{-1/3}$), therefore SKA naturally includes points from "sparse" clusters of size $\Omega(n^{1/3})$ in the original set.

**Herding.** The kernel Herding algorithm (Algorithm 6) first estimates the density of the dataset via random sam-

pling; the sketch is then produced by iteratively selecting points with maximum residual density. The algorithm has a complexity of $O(nm + ns)$, where $m$ stands for the sample size used to produce the initial density estimates.

To keep Herding under the same $2n$ compute budget, we downsample the dataset to size $n_h = \frac{n}{s}$, and use $m = s$ samples to estimate the initial density. This means that, the larger the sketch size is, the less accurate the initial density estimate is. As a result, we observe degrading performance at larger sketch sizes $s = \Omega(\sqrt{n})$.

**Algorithm 5** Sparse Kernel Approximation (SKA)

**Input:** set $P$, kernel $K$, size $s$.
$S = \{x_1, \dots, x_s\} \leftarrow$ Greedy-kcenter$(P, s)$
$K \in \mathbb{R}^{s \times s}$ with $K_{ij} \leftarrow k(x_i, x_j)$ for $x_i, x_j \in S$.
$y \in \mathbb{R}^s$ with $y_i \leftarrow \mathrm{KDE}_P^w(x_i)$ for $x_i \in S$.
Let $\hat{w}$ be a solution to $K\hat{w} = y$.
**Output:** $(S, \hat{w})$

**Algorithm 6** Approximate Kernel Herding (AKH)

**Input:** set $P$, kernel $K$, size $s$, samples $m$.
**for** $i = 1, \dots, |P|$ **do**
  $P_i \leftarrow \mathrm{Random}(P, m)$. ▷ random set of $m$ points
  $d_i \leftarrow \mathrm{KDE}_{P_i}(x_i)$ ▷ estimate of the density
$S_0 \leftarrow \emptyset$ ▷ initialization
**for** $t = 1, \dots, s$ **do**
  $j^* \leftarrow \arg\max_{i \in [n]} \{d_i - \mathrm{KDE}_{S_{t-1}}(x_i)\}$ ▷ greedy
  $S_t \leftarrow S_{t-1} \cup \{x_j^*\}$ ▷ add point to the set
**Output:** $(S_s, \frac{1}{s}\mathbf{1}_s))$ ▷ return the sketch

**Results.** Figure 5 reports the relative error on random queries (left), i.e. uniformly random points from the dataset, and low-density queries (right), uniformly random points of the dataset with density around a threshold $\tau$. Uniform, SKA and HBS achieve similar mean error on random queries, while the latter two have improved performance on low-density queries, with HBS performing slightly better than SKA. By design, HBS has similar performance with random sampling on average, but performs better on relatively "sparse" regions due to the theoretical guarantee that buckets with weight at least $\tau$ are sampled with high probability. Combining SKA with random samples initially results in performance degradation but eventually acts as a form of regularization improving upon random. Kernel Herding is competitive only for a small number of points in the sketch.

**Overhead reduction.** In Table 5, we report on the estimated preprocessing runtime reduction enabled by HBS for the density estimation results reported in Table 1 of the main paper. The estimates are calculated by the dividing the number of data points hashed according to the original

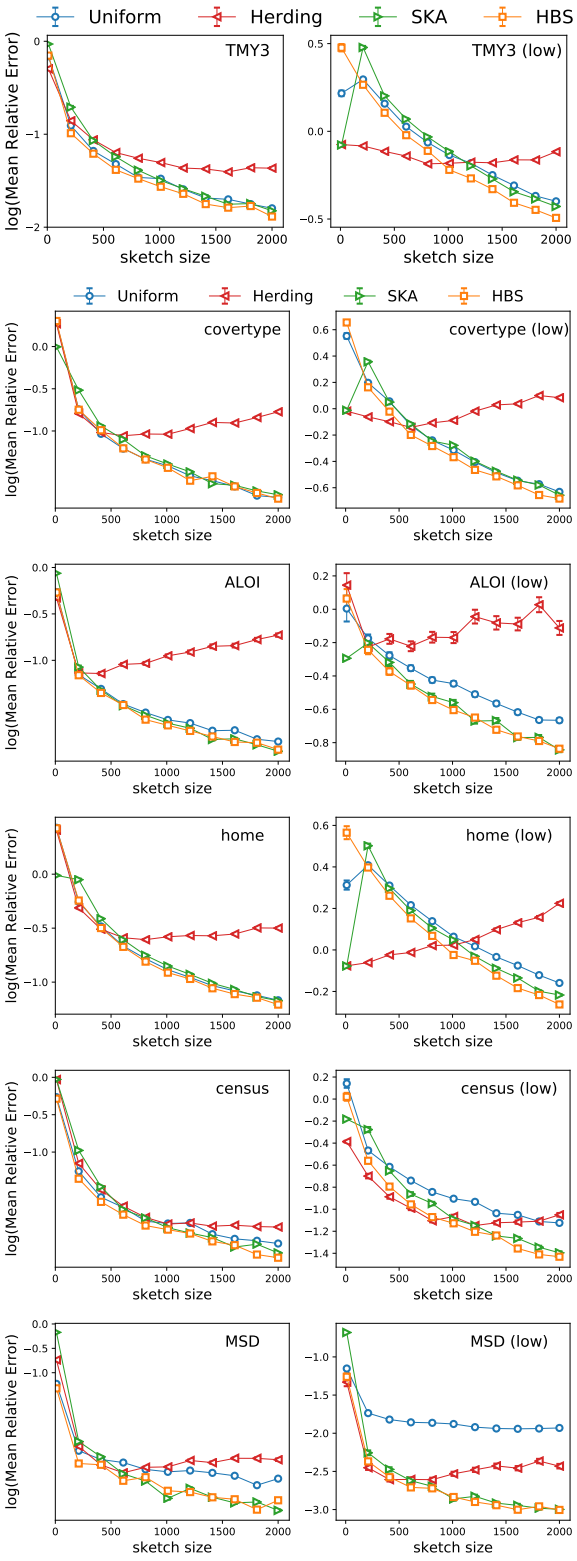



*Figure 5.* Sketching results on random and low-density queries.

*Table 5.* Estimated overhead reduction enabled by HBS.

| | census | TMY3 | TIMIT | SVHN | covertype | MSD | GloVe | ALOI |
|---|---|---|---|---|---|---|---|---|
| Reduction (est.) | 958× | 755× | 821× | 659× | 554× | 607× | 595× | 303× |

HBE procedure by the number of data points hashed after enabling HBS.

### 5.4. Visualizations of real-world data sets

Our Log-Condition plots use circles with radius $r$ to represent points with weights roughly $e^{-r}$ (roughly at distance $\sqrt{r}$ for the Gaussian kernel). The visualizations are generated by plotting overlapping annuli around the origin that represent a random queries from the dataset, such that the width of the annulus roughly corresponds to the log of the relative variance of random sampling.

We observe two distinctive types of visualizations. Datasets like census exhibit dense inner circles, meaning that a small number of points close to the query contribute significantly towards the density. To estimate the density accurately, one must sample from these small clusters, which HBE does better than RS. In contrast, datasets like MSD exhibit more weight on the outer circles, meaning that a large number of "far" points is the main source of density. Random sampling has a good chance of seeing these "far" points, and therefore, tends to perform better on such datasets. The top two plots in Figure 6 amplify these observations on synthetic datasets with highly clustered/scattered structures. RS performs better for all datasets in the right column except for SVHN.

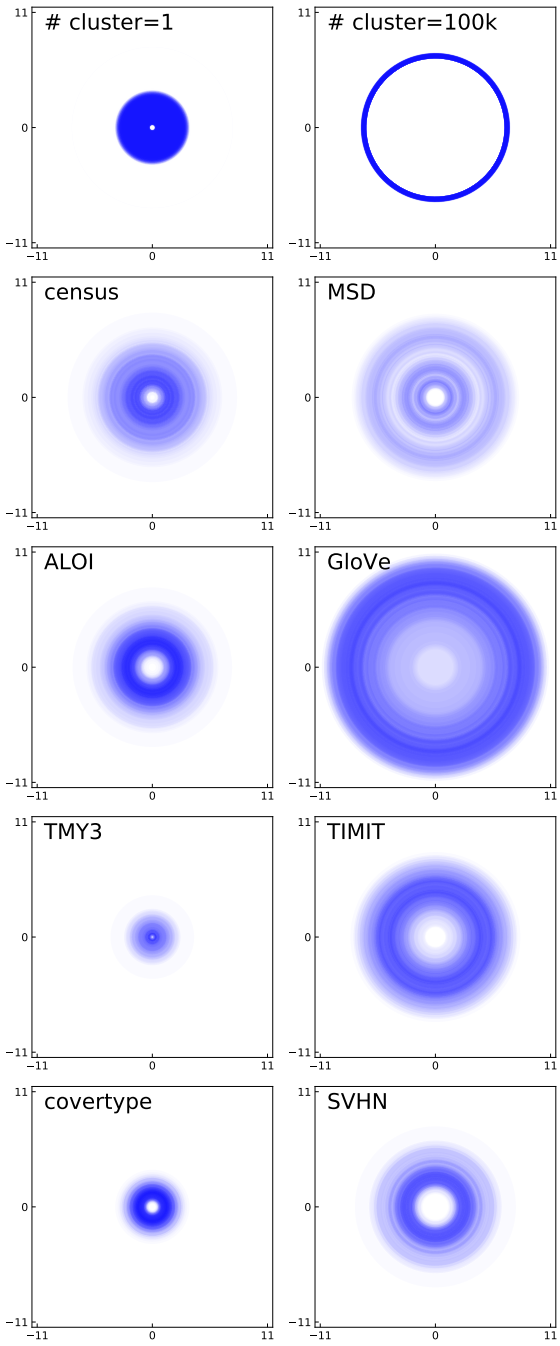



*Figure 6.* Visualizations of datasets. The top row shows two extreme cases of highly clustered (# cluster=1) versus highly scattered (#cluster=100k) datasets. RS performs better for all datasets in the right column expect for SVHN.

## References

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen, Y., Welling, M., and Smola, A. Super-samples from Kernel Herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pp. 109–116, Arlington, Virginia, United States, 2010. AUAI Press. ISBN 978-0-9749039-6-5.

Cortes, E. C. and Scott, C. Sparse Approximation of a Kernel Mean. *Trans. Sig. Proc.*, 65(5):1310–1323, March 2017. ISSN 1053-587X.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.