Hashing-Based-Estimators for Kernel Density in High Dimensions

Moses Charikar **Paris Siminelakis**

Stanford University

FOCS 2017 @ Berkeley, CA

Oct 17, 2017

Importance Sampling for Kernel Density in High Dimensions

Moses Charikar **Paris Siminelakis**

Stanford University

FOCS 2017 @ Berkeley, CA

Oct 17, 2017

Importance Sampling for Kernel Density in High Dimensions

Moses Charikar **Paris Siminelakis**

Stanford University

FOCS 2017 @ Berkeley, CA

Oct 17, 2017

Importance Sampling for Approximating Structured Sums in High Dimensions

Moses Charikar **Paris Siminelakis**

Stanford University

FOCS 2017 @ Berkeley, CA

Oct 17, 2017

Importance Sampling for Approximating Structured Sums in High Dimensions

Moses Charikar **Paris Siminelakis**

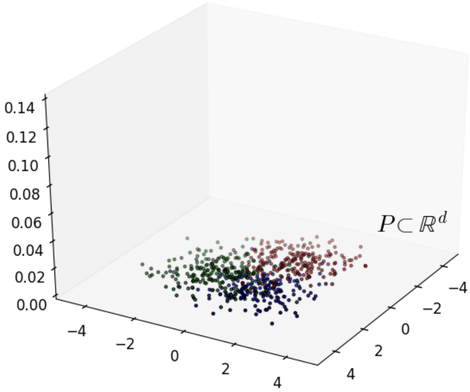
Stanford University

FOCS 2017 @ Berkeley, CA

Oct 17, 2017

Density Estimation

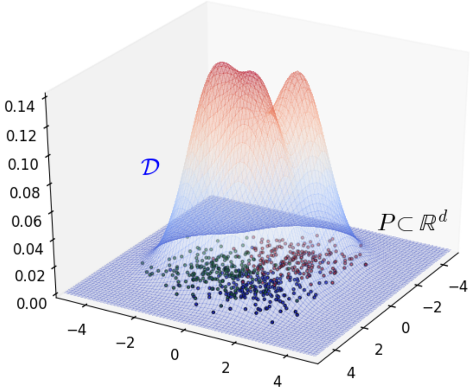
Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?



Non-parametric

Density Estimation

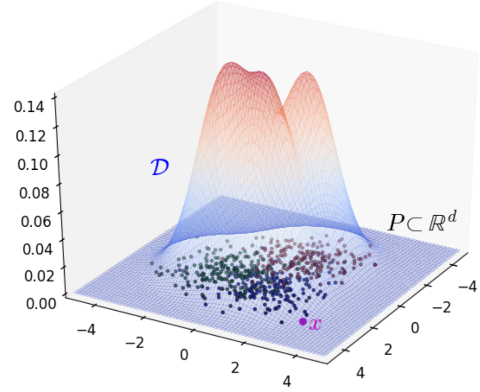
Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?



Non-parametric

Density Estimation

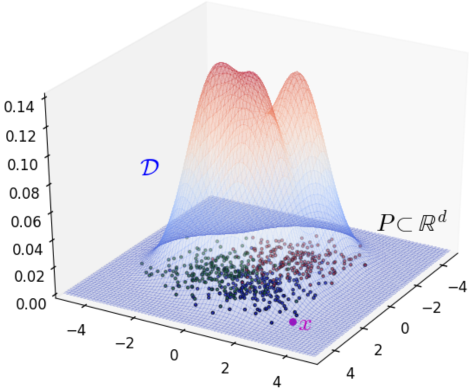
Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?



Non-parametric

Density Estimation

Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?

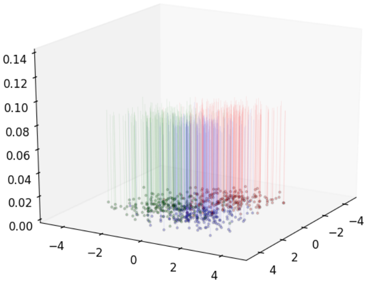


Non-parametric

Kernel Density Estimation

Kernel Density Estimate

$$K_\sigma(x, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$$

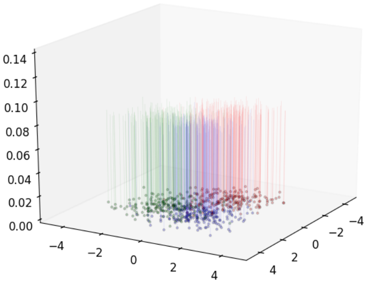


dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

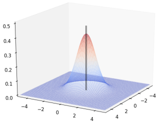
$$\mathrm{KDE}_{\mathbf{P}}(\mathbf{x}) := \frac{1}{|\mathbf{P}|} \sum_{\mathbf{y} \in \mathbf{P}} \mathbf{K}_\sigma(\mathbf{x}, \mathbf{y})$$

Kernel Density Estimate

$$K_\sigma(x, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$$



$\overset{*K_\sigma}{\Longrightarrow}$

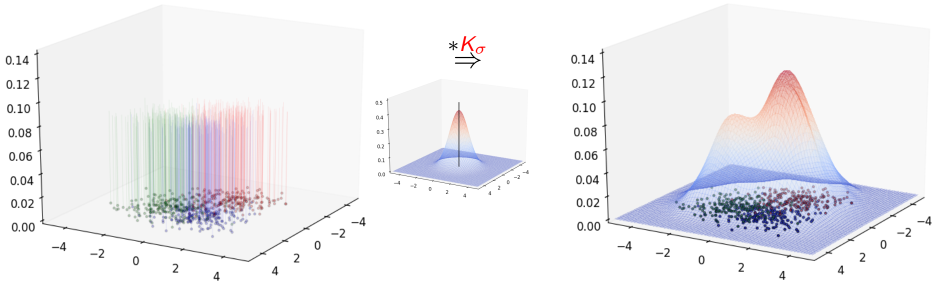


dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

$$\mathrm{KDE}_{\mathbf{P}}(\mathbf{x}) := \frac{1}{|\mathbf{P}|} \sum_{\mathbf{y} \in \mathbf{P}} K_\sigma(\mathbf{x}, \mathbf{y})$$

Kernel Density Estimate

$$K_\sigma(x, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$$

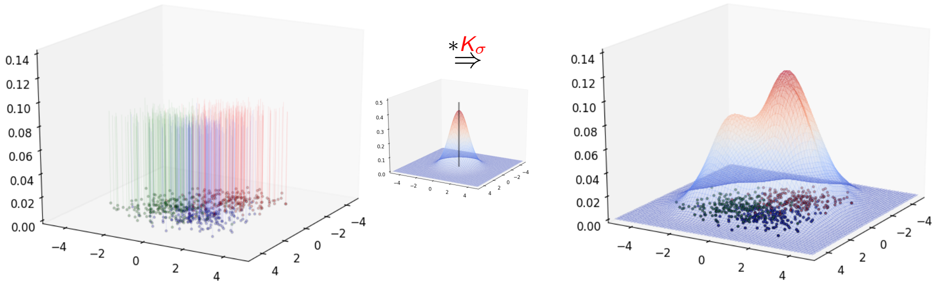


dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$, query $\mathbf{x}$

$$\mathrm{KDE}_{\mathbf{P}}(\mathbf{x}) := \frac{1}{|\mathbf{P}|}\sum_{\mathbf{y}\in\mathbf{P}} K_\sigma(\mathbf{x}, \mathbf{y})$$

Kernel Density Estimate

$$K_\sigma(x, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$$

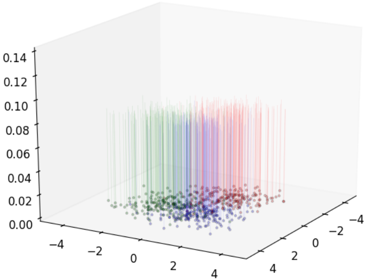


dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

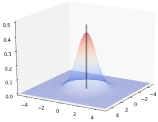
$$\mathrm{KDE}_{\mathbf{P}}(\mathbf{x}) := \frac{1}{|\mathbf{P}|} \sum_{\mathbf{y} \in \mathbf{P}} \mathbf{K}_\sigma(\mathbf{x}, \mathbf{y})$$

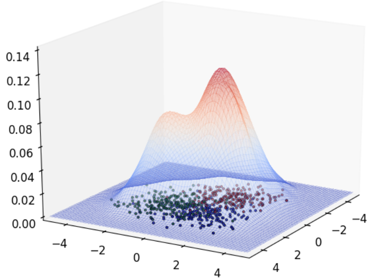
Kernel Density Estimate

$$K_\sigma(x, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$$



$\overset{*K_\sigma}{\Longrightarrow}$



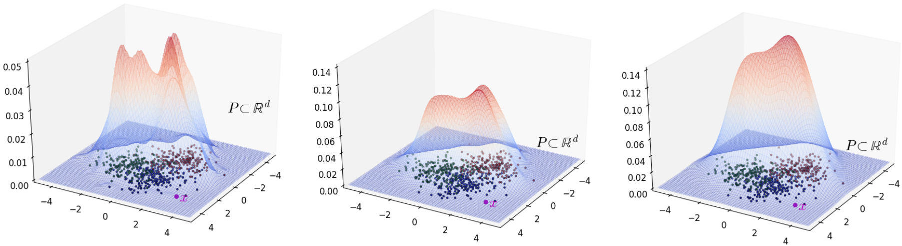


dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

$$\mathrm{KDE}_{\mathbf{P}}(\mathbf{x}) := \frac{1}{|\mathbf{P}|} \sum_{\mathbf{y} \in \mathbf{P}} \mathbf{K}_\sigma(\mathbf{x}, \mathbf{y})$$

Kernel Density Estimate

$$K_\sigma(x, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right)$$



$\sigma_1 = \frac{1}{2} \cdot \mathrm{std}$ $\sigma_2 = \frac{3}{4} \cdot \mathrm{std}$ $\sigma_3 = \mathrm{std}$

dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

$$\mathrm{KDE}_{\mathbf{P}}(\mathbf{x}) := \frac{1}{|\mathbf{P}|} \sum_{\mathbf{y} \in \mathbf{P}} \mathbf{K}_\sigma(\mathbf{x}, \mathbf{y})$$

Applications of KDE

$$\mathrm{KDE}_P^w(\mathbf{x}) := \sum_{y \in P} w_y \cdot K_\sigma(\mathbf{x}, y)$$

Numerous applications in Machine Learning and Statistics:

1. Mode Estimation
2. Outlier Detection
3. Local Regression
4. Density based Clustering/Classification
5. Kernel Methods: k-PCA,k-ridge regression, RKHS
6. Topological Data analysis.

How fast can we **approximate** $\mathrm{KDE}_P(\mathbf{x})$?

Applications of KDE

$$\mathrm{KDE}_P^w(\mathbf{x}) := \sum_{y \in P} w_y \cdot K_\sigma(\mathbf{x}, y)$$

Numerous applications in Machine Learning and Statistics:

1. Mode Estimation
2. Outlier Detection
3. Local Regression
4. Density based Clustering/Classification
5. Kernel Methods: k-PCA,k-ridge regression, RKHS
6. Topological Data analysis.

How fast can we **approximate** $\mathrm{KDE}_P(\mathbf{x})$?

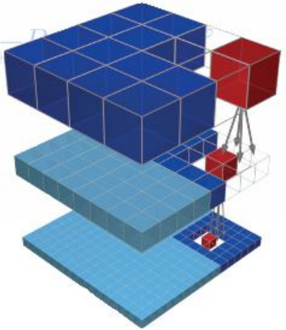
(μ, ϵ, δ)-KDE Problem

Given $\mathbf{P} \subset \mathbb{R}^d$ and a level $\mu \in [\tau, 1]$, design a data-structure that for any query $\mathbf{x} \in \mathbb{R}^d$ answers correctly w.p at least $1 - \delta$ whether

$$\mathrm{KDE}_P(\mathbf{x}) \leq (1 - \epsilon) \cdot \mu \qquad \text{or} \qquad \mathrm{KDE}_P(\mathbf{x}) \geq (1 + \epsilon) \cdot \mu$$

Low Dimensions

Fast Multipole Methods [Barnes-Hut'1985][Greengard-Röglin'87]



Credit: InSiDE ScaFaCoS

- Hierarchical Space Partitions
- **WSPD** [Callaghan, Kosaraju'95]
- Series Expansions of Kernels
- Fast Gauss Transform

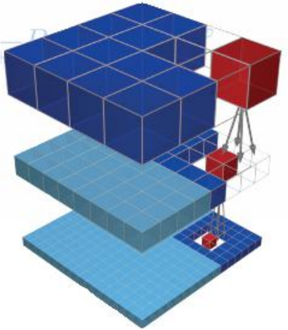
$$O(\log^d(n)) - \text{time}$$

- Core-sets [Phillips'11,+'17]

Curse of Dimensionality $d = \Omega(\frac{\log n}{\log \log n})$

Low Dimensions

Fast Multipole Methods [Barnes-Hut'1985][Greengard-Röglin'87]



Credit: InSiDE ScaFaCoS

- Hierarchical Space Partitions
- **WSPD** [Callaghan, Kosaraju'95]
- Series Expansions of Kernels
- Fast Gauss Transform

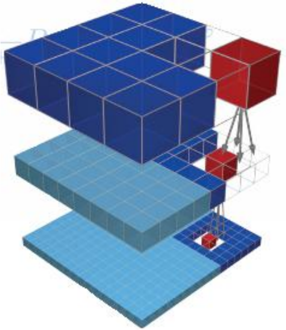
$$O(\log^{d}(n)) - \text{time}$$

- Core-sets [Phillips'11,+'17]

Curse of Dimensionality $d = \Omega(\frac{\log n}{\log \log n})$

Low Dimensions

Fast Multipole Methods [Barnes-Hut'1985][Greengard-Röglin'87]



Credit: InSiDE ScaFaCoS

- Hierarchical Space Partitions
- **WSPD** [Callaghan, Kosaraju'95]
- Series Expansions of Kernels
- Fast Gauss Transform

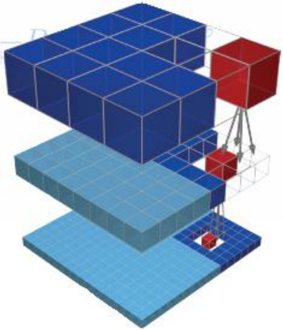
$$O(\log^d(n)) - \text{time}$$

- Core-sets [Phillips'11,+'17]

Curse of Dimensionality $d = \Omega(\frac{\log n}{\log \log n})$

Low Dimensions

Fast Multipole Methods [Barnes-Hut'1985][Greengard-Röglin'87]



Credit: *InSiDE ScaFaCoS*

- Hierarchical Space Partitions
- **WSPD** [Callaghan, Kosaraju'95]
- Series Expansions of Kernels
- Fast Gauss Transform

$$O(\log^{d}(n)) - \text{time}$$

- Core-sets [Phillips'11,+'17]

Curse of Dimensionality $d = \Omega(\frac{\log n}{\log \log n})$

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|}\sum_{y \in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|}\sum_{y \in P} K^2(x, y) \leq \frac{1}{|P|}\sum_{y \in P} K(x, y) = (\mathbb{E}[Z_{RS}])^2 \frac{1}{\mu}$$

Median-of-Means technique finishes the proof □

Random Sampling was state of the art!

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|} \sum_{y \in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu} \frac{1}{\epsilon^2} \log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|} \sum_{y \in P} K^2(x, y) \leq \frac{1}{|P|} \sum_{y \in P} K(x, y) = (\mathbb{E}[Z_{RS}])^2 \frac{1}{\mu}$$

Median-of-Means technique finishes the proof □

Random Sampling was state of the art!

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|}\sum_{y \in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|}\sum_{y \in P} K^2(x, y) \leq \frac{1}{|P|}\sum_{y \in P} K(x, y) = (\mathbb{E}[Z_{RS}])^2\frac{1}{\mu}$$

Median-of-Means technique finishes the proof □

Random Sampling was state of the art!

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|}\sum_{y\in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|}\sum_{y\in P} K^2(x,y) \le \frac{1}{|P|}\sum_{y\in P} K(x,y) = (\mathbb{E}[Z_{RS}])^2 \frac{1}{\mu}$$

Median-of-Means technique finishes the proof □

Random Sampling was state of the art!

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|}\sum_{y \in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|}\sum_{y \in P} K^2(x, y) \le \frac{1}{|P|}\sum_{y \in P} K(x, y) = (\mathbb{E}[Z_{RS}])^2 \frac{1}{\mu}$$

Median-of-Means technique finishes the proof □

Random Sampling was state of the art!

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|}\sum_{y\in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|}\sum_{y\in P} K^2(x,y) \le \frac{1}{|P|}\sum_{y\in P} K(x,y) = (\mathbb{E}[Z_{RS}])^2\frac{1}{\mu}$$

Median-of-Means technique finishes the proof $\square$

Random Sampling was state of the art!

High Dimensions

$$\mu = \mathrm{KDE}_P(\mathbf{x}) = \frac{1}{|P|}\sum_{y\in P} K(\mathbf{x}, y)$$

Proposition

Random Sampling solves (μ, ϵ, δ)-KDE problem in $O(\frac{1}{\mu}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

Proof: Variance calculation

$$\mathbb{E}[Z_{RS}^2] = \frac{1}{|P|}\sum_{y\in P} K^2(x, y) \leq \frac{1}{|P|}\sum_{y\in P} K(x, y) = (\mathbb{E}[Z_{RS}])^2\frac{1}{\mu}$$

Median-of-Means technique finishes the proof $\square$

Random Sampling was state of the art!

Lower bounds (μ, ϵ, δ)-KDE Problem

Theorem 1 [Charikar, S'17]

Any data structure in the cell probe model with m cells, wordsize $w \leq \frac{1}{\mu}$, that is correct with probability $> \frac{1}{2}$ using a single probe satisfies: $m \cdot w = \Omega(\frac{1}{\mu})$.

- Lower bound against adaptive coresets $\rightarrow f(S, x)$
- For 1-probe random sampling is optimal.
- Holds only for the **Gaussian kernel**.
- Reduce hard instances for c-ANN with $c = O(\frac{\log(1/\mu)}{\log(1/\epsilon)})$ and $d = \Omega(\log^3(n))$.

Main Result

Theorem 2 [Charikar, S.'17]

There exists a data-structure based on hashing that requires space $\tilde{O}_R(n\frac{1}{\sqrt{\tau}}\frac{1}{\epsilon^2})$ that solves the (ϵ, μ, δ)-KDE Problem for any $\mu \in [\tau, 1]$ using $\tilde{O}_R(\frac{1}{\sqrt{\mu}}\frac{1}{\epsilon^2})$ time, where $R = \mathrm{diam}(P \cup \{x\})$

Gaussian	Exponential	Generalized t-Student
$e^{-\|x-y\|^2}$	$e^{-\|x-y\|}$	$\frac{1}{1+\|x-y\|^t}$

- $\sqrt{\mu}$-improvement over Random Sampling.
- Adaptively estimate μ.

Main Result

Theorem 2 [Charikar, S.'17]

There exists a data-structure based on hashing that requires space $\tilde{O}_R(n\frac{1}{\sqrt{\tau}}\frac{1}{\epsilon^2})$ that solves the (ϵ, μ, δ)-KDE Problem for any $\mu \in [\tau, 1]$ using $\tilde{O}_R(\frac{1}{\sqrt{\mu}}\frac{1}{\epsilon^2})$ time, where $R = \mathrm{diam}(P \cup \{x\})$

Gaussian	Exponential	Generalized t-Student
$e^{-\|x-y\|^2}$	$e^{-\|x-y\|}$	$\frac{1}{1+\|x-y\|^t}$

- $\sqrt{\mu}$-improvement over Random Sampling.
- Adaptively estimate μ.

Main Result

Theorem 2 [Charikar, S.'17]

There exists a data-structure based on hashing that requires space $\tilde{O}_R(n\frac{1}{\sqrt{\tau}}\frac{1}{\epsilon^2})$ that solves the (ϵ, μ, δ)-KDE Problem for any $\mu \in [\tau, 1]$ using $\tilde{O}_R(\frac{1}{\sqrt{\mu}}\frac{1}{\epsilon^2})$ time, where $R = \mathrm{diam}(P \cup \{x\})$

Gaussian	Exponential	Generalized t-Student
$e^{-\|x-y\|^2}$	$e^{-\|x-y\|}$	$\frac{1}{1+\|x-y\|^t}$

- $\sqrt{\mu}$-improvement over Random Sampling.
- Adaptively estimate μ.

Main Result

Theorem 2 [Charikar, S.'17]

There exists a data-structure based on hashing that requires space $\tilde{O}_R(n\frac{1}{\sqrt{\tau}}\frac{1}{\epsilon^2})$ that solves the (ϵ, μ, δ)-KDE Problem for any $\mu \in [\tau, 1]$ using $\tilde{O}_R(\frac{1}{\sqrt{\mu}}\frac{1}{\epsilon^2})$ time, where $R = \mathrm{diam}(P \cup \{x\})$

Gaussian	Exponential	Generalized t-Student
$e^{-\|x-y\|^2}$	$e^{-\|x-y\|}$	$\frac{1}{1+\|x-y\|^t}$

- $\sqrt{\mu}$-improvement over Random Sampling.
- Adaptively estimate μ.

Upper bound

1. Unbiased Estimator $\Rightarrow$ **Importance sampling**
2. Assuming μ is known $\Rightarrow$ Bound variance (Hölder-type ineq.)
3. Take enough samples to lower variance $\Rightarrow$ Median-of-means
4. Deal with μ unknown $\Rightarrow$ Adaptive mean relaxation (general)

Upper bound

1. Unbiased Estimator $\Rightarrow$ **Importance sampling**
2. Assuming μ is known $\Rightarrow$ Bound variance (Hölder-type ineq.)
3. Take enough samples to lower variance $\Rightarrow$ Median-of-means
4. Deal with μ unknown $\Rightarrow$ Adaptive mean relaxation (general)

Upper bound

1. Unbiased Estimator $\Rightarrow$ **Importance sampling**
2. Assuming μ is known $\Rightarrow$ Bound variance (Hölder-type ineq.)
3. Take enough samples to lower variance $\Rightarrow$ Median-of-means
4. Deal with μ unknown $\Rightarrow$ Adaptive mean relaxation (general)

Upper bound

1. Unbiased Estimator $\Rightarrow$ **Importance sampling**
2. Assuming μ is known $\Rightarrow$ Bound variance (Hölder-type ineq.)
3. Take enough samples to lower variance $\Rightarrow$ Median-of-means
4. Deal with μ unknown $\Rightarrow$ Adaptive mean relaxation (general)

Importance Sampling

Simplified view

For each $x_i \in P$ and query $\mathbf{x}$ let $w_i := K(\mathbf{x}, x_i)$.

Approximate $\mathrm{KDE}_P(\mathbf{x}) \Leftrightarrow$ Approximate $\sum_{i=1}^{n} w_i$

- Random sampling samples each point with prob. $\frac{1}{|P|}$
- **Issue:** if small number of weights have large contribution.

Build a **better** sampler!

Simplified view

For each $x_i \in P$ and query $\mathbf{x}$ let $w_i := K(\mathbf{x}, x_i)$.

Approximate $\mathrm{KDE}_P(\mathbf{x}) \Leftrightarrow$ Approximate $\sum_{i=1}^{n} w_i$

- Random sampling samples each point with prob. $\frac{1}{|P|}$
- **Issue:** if small number of weights have large contribution.

Build a **better** sampler!

Simplified view

For each $x_i \in P$ and query $\mathbf{x}$ let $w_i := K(\mathbf{x}, x_i)$.

$$\text{Approximate } \mathrm{KDE}_P(\mathbf{x}) \Leftrightarrow \text{Approximate } \sum_{i=1}^{n} w_i$$

- Random sampling samples each point with prob. $\frac{1}{|P|}$
- **Issue:** if small number of weights have large contribution.

Build a **better** sampler!

Simplified view

For each $x_i \in P$ and query $\mathbf{x}$ let $w_i := K(\mathbf{x}, x_i)$.

Approximate $\mathrm{KDE}_P(\mathbf{x}) \Leftrightarrow$ Approximate $\sum_{i=1}^{n} w_i$

- Random sampling samples each point with prob. $\frac{1}{|P|}$
- **Issue:** if small number of weights have large contribution.

Build a **better** sampler!

Simplified view

For each $x_i \in P$ and query $\mathbf{x}$ let $w_i := K(\mathbf{x}, x_i)$.

$$\mathrm{KDE}_P(\mathbf{x}) \Leftrightarrow \text{Approximate } \sum_{i=1}^{n} w_i$$

- Random sampling samples each point with prob. $\frac{1}{|P|}$
- **Issue:** if small number of weights have large contribution.

Build a **better** sampler!

Importance Sampling (IS)

- **Black box** Q returns index i with probability q_i.
- Unbiased estimator

$$Z_I = \frac{w_I}{q_I}, \qquad I \sim \mathbf{Q}$$

- Variance $\sum_{i=1}^{n} \frac{w_i^2}{q_i}$ minimized for $q_i \propto w_i = K_\sigma(\mathbf{x}, x_i)$.

How to efficiently get such sampling probabilities $q(\mathbf{x}, y)$ for every query $\mathbf{x} \in \mathbb{R}^d$?

Importance Sampling (IS)

- **Black box** Q returns index i with probability q_i.
- Unbiased estimator

$$Z_I = \frac{w_I}{q_I}, \qquad I \sim \mathbf{Q}$$

- Variance $\sum_{i=1}^{n} \frac{w_i^2}{q_i}$ minimized for $q_i \propto w_i = K_\sigma(\mathbf{x}, x_i)$.

How to efficiently get such sampling probabilities $q(\mathbf{x}, y)$ for every query $\mathbf{x} \in \mathbb{R}^d$?

Importance Sampling (IS)

- **Black box** Q returns index i with probability q_i.
- Unbiased estimator

$$Z_I = \frac{w_I}{q_I}, \qquad I \sim \mathbf{Q}$$

- Variance $\sum_{i=1}^{n} \frac{w_i^2}{q_i}$ minimized for $q_i \propto w_i = K_\sigma(\mathbf{x}, x_i)$.

How to efficiently get such sampling probabilities $q(\mathbf{x}, y)$ for every query $\mathbf{x} \in \mathbb{R}^d$?

Importance Sampling (IS)

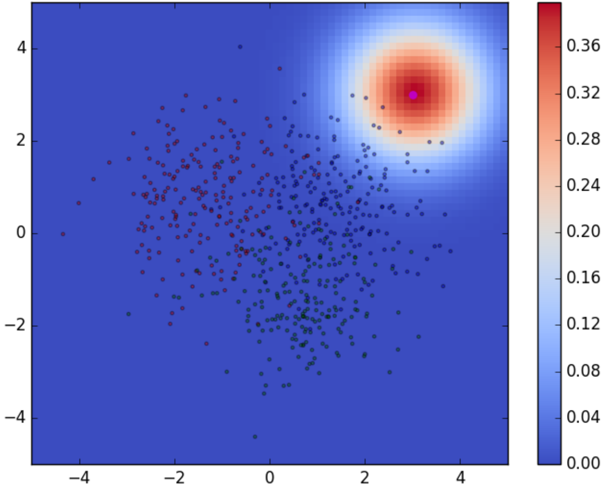
- **Black box** Q returns index i with probability q_i.
- Unbiased estimator

$$Z_I = \frac{w_I}{q_I}, \qquad I \sim \mathbf{Q}$$

- Variance $\sum_{i=1}^{n} \frac{w_i^2}{q_i}$ minimized for $q_i \propto w_i = K_\sigma(\mathbf{x}, x_i)$.

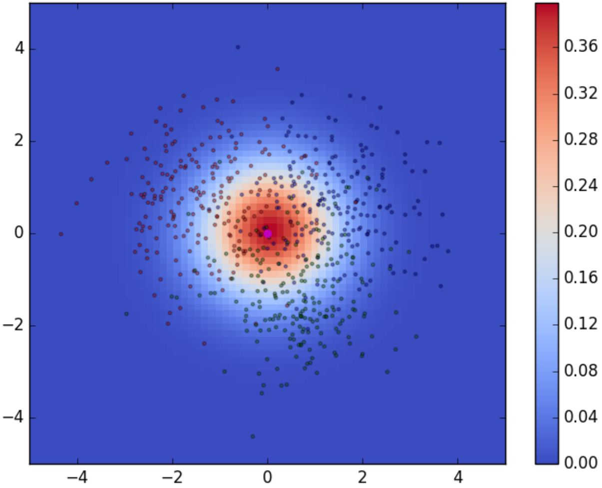
How to efficiently get such sampling probabilities $q(\mathbf{x}, y)$ for every query $\mathbf{x} \in \mathbb{R}^d$?

Adaptive Sampling Probabilities



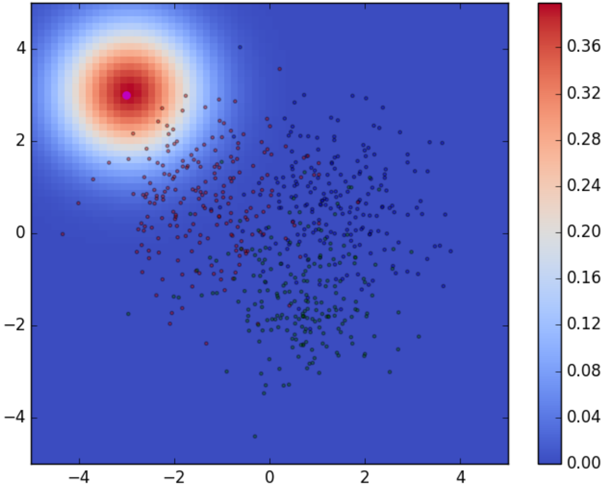
Locality Sensitive Hashing [IM'98][DIIM'04][AI'06]!

Adaptive Sampling Probabilities



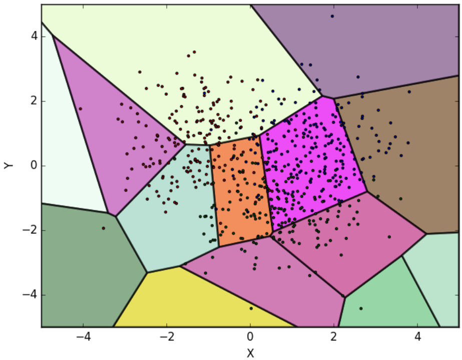
Locality Sensitive Hashing [IM'98][DIIM'04][AI'06]!

Adaptive Sampling Probabilities



Locality Sensitive Hashing [IM'98][DIIM'04][AI'06]!

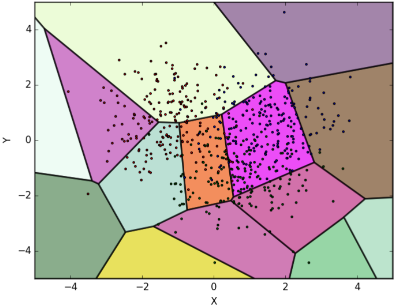
Adaptive Sampling Probabilities



Locality Sensitive Hashing [IM'98][DIIM'04][AI'06]!

Locality Sensitive Hashing

- Hash family $\mathcal{H}$, e.g. $h_{\omega,u}(x) = \lceil \frac{\omega^\top x + u}{w} \rceil$
- Distribution ν, e.g. $\omega \sim N(0, I_d), u \sim [0, w]$



- Collision probability

$$p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$$

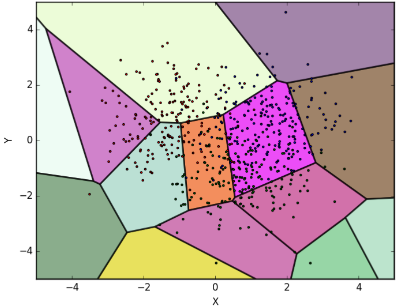
- Monotone function f such

$$p(\mathbf{x}, y) = f(\|x - y\|)$$

LSH as **Importance Sampling**!

Locality Sensitive Hashing

- Hash family $\mathcal{H}$, e.g. $h_{\omega,u}(x) = \lceil \frac{\omega^\top x+u}{w} \rceil$
- Distribution ν, e.g. $\omega \sim N(0, I_d), u \sim [0, w]$



- Collision probability

$$p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$$

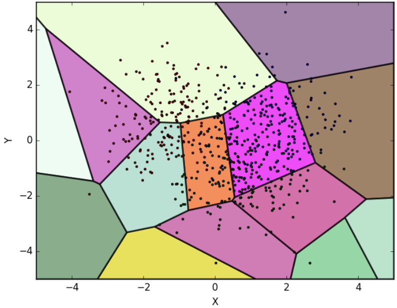
- Monotone function f such

$$p(\mathbf{x}, y) = f(\|x - y\|)$$

LSH as **Importance Sampling**!

Locality Sensitive Hashing

- Hash family $\mathcal{H}$, e.g. $h_{\omega,u}(x) = \lceil \frac{\omega^\top x + u}{w} \rceil$
- Distribution ν, e.g. $\omega \sim N(0, I_d), u \sim [0, w]$



- Collision probability

$$p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$$

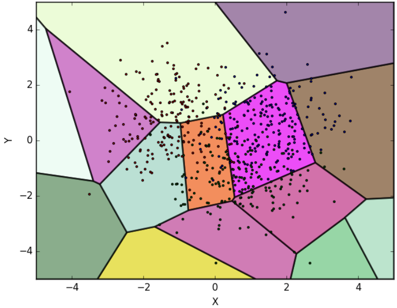
- Monotone function f such

$$p(\mathbf{x}, y) = f(\|x - y\|)$$

LSH as **Importance Sampling**!

Locality Sensitive Hashing

- Hash family $\mathcal{H}$, e.g. $h_{\omega,u}(x) = \lceil \frac{\omega^\top x + u}{w} \rceil$
- Distribution ν, e.g. $\omega \sim N(0, I_d), u \sim [0, w]$



- Collision probability

$$p(\mathbf{x}, y) = \mathbb{P}_{h \sim \mathcal{H}}[h(\mathbf{x}) = h(y)]$$

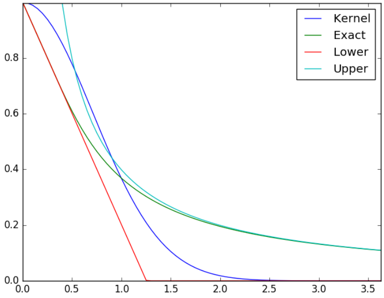
- Monotone function f such

$$p(\mathbf{x}, y) = f(\|x - y\|)$$

LSH as **Importance Sampling**!

Locality Sensitive Hashing

- Hash family $\mathcal{H}$, e.g. $h_{\omega,u}(x) = \lceil \frac{\omega^\top x + u}{w} \rceil$
- Distribution ν, e.g. $\omega \sim N(0, I_d), u \sim [0, w]$



- Collision probability

$$p(\mathbf{x}, y) = \mathbb{P}_{h \sim \mathcal{H}}[h(\mathbf{x}) = h(y)]$$

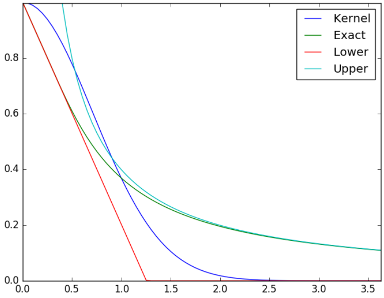
- Monotone function f such

$$p(\mathbf{x}, y) = f(\|x - y\|)$$

LSH as **Importance Sampling**!

Locality Sensitive Hashing

- Hash family $\mathcal{H}$, e.g. $h_{\omega,u}(x) = \lceil \frac{\omega^\top x + u}{w} \rceil$
- Distribution ν, e.g. $\omega \sim N(0, I_d), u \sim [0, w]$



- Collision probability

$$p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$$

- Monotone function f such

$$p(\mathbf{x}, y) = f(\|x - y\|)$$

LSH as **Importance Sampling**!

Hashing-Based-Estimators

Importance Sampling through Hashing

Preprocessing

- Hash functions $\mathcal{H}$ with c.p. $p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$.
- Evaluate $h_1, \ldots, h_m \sim \mathcal{H}$ on $\mathbf{P}$.

Querying

- Conditioning: let $H_1(\mathbf{x}) := \{y \in \mathbf{P} : h_1(y) = h_1(\mathbf{x})\}$.
- Random Sampling: pick a random index I from $H_1(\mathbf{x})$

Unbiased Estimator

$$Z_{h_1}(\mathbf{x}) = \frac{K(\mathbf{x}, x_I)}{\frac{p(\mathbf{x}, x_I)}{|H_1(\mathbf{x})|}}$$

Main **technical contribution** - bounding the variance!

Importance Sampling through Hashing

Preprocessing

- Hash functions $\mathcal{H}$ with c.p. $p(\mathbf{x}, y) = \mathbb{P}_{h \sim \mathcal{H}}[h(\mathbf{x}) = h(y)]$.
- Evaluate $h_1, \ldots, h_m \sim \mathcal{H}$ on $\mathbf{P}$.

Querying

- Conditioning: let $H_1(\mathbf{x}) := \{y \in \mathbf{P} : h_1(y) = h_1(\mathbf{x})\}$.
- Random Sampling: pick a random index I from $H_1(\mathbf{x})$

Unbiased Estimator

$$Z_{h_1}(\mathbf{x}) = \frac{K(\mathbf{x}, x_I)}{\frac{p(\mathbf{x}, x_I)}{|H_1(\mathbf{x})|}}$$

Main **technical contribution** - bounding the variance!

Importance Sampling through Hashing

Preprocessing

- Hash functions $\mathcal{H}$ with c.p. $p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$.
- Evaluate $h_1, \ldots, h_m \sim \mathcal{H}$ on $\mathbf{P}$.

Querying

- Conditioning: let $H_1(\mathbf{x}) := \{y \in \mathbf{P} : h_1(y) = h_1(\mathbf{x})\}$.
- Random Sampling: pick a random index I from $H_1(\mathbf{x})$

Unbiased Estimator

$$Z_{h_1}(\mathbf{x}) = \frac{K(\mathbf{x}, x_I)}{\frac{p(\mathbf{x}, x_I)}{|H_1(\mathbf{x})|}}$$

Main **technical contribution** - bounding the variance!

Importance Sampling through Hashing

Preprocessing

- Hash functions $\mathcal{H}$ with c.p. $p(\mathbf{x}, y) = \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{x}) = h(y)]$.
- Evaluate $h_1, \ldots, h_m \sim \mathcal{H}$ on $\mathbf{P}$.

Querying

- Conditioning: let $H_1(\mathbf{x}) := \{y \in \mathbf{P} : h_1(y) = h_1(\mathbf{x})\}$.
- Random Sampling: pick a random index I from $H_1(\mathbf{x})$

Unbiased Estimator

$$Z_{h_1}(\mathbf{x}) = \frac{K(\mathbf{x}, x_I)}{\frac{p(\mathbf{x}, x_I)}{|H_1(\mathbf{x})|}}$$

Main **technical contribution** - bounding the variance!

Variance of HBE

$$\mathbb{E}[Z_h^2] = \sum_{i=1}^{n} \frac{w_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| | i \in H(\mathbf{x})]$$

Theorem 3 [Charikar, S.'17]

Worst case datasets for HBE have support on two points.

- Linearity of expectation: $\mathbb{E}[|H(x)||i \in H(x)] = \sum_j \frac{P(i,j \in H(x))}{p_i}$
- Monotonicity: $P(i,j \in H(x)) \leq \min\{p_i, p_j\}$

$$\mathbb{E}[Z_h^2] \leq \sup\left\{ f^\top A f \middle| \|f\|_1 \leq 1, \|f\|_{w,1} \leq \mu \right\}$$
$$\leq 4 \cdot \|\tilde{A}(\mu, \{p_i\}, \{w_i\})\|_{1,\infty}$$

Quantifies **Compatibility** between $\{w_i\}, \{p_i\}$ at level μ

Variance of HBE

$$\mathbb{E}[Z_h^2] = \sum_{i=1}^{n} \frac{w_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| \,|\, i \in H(\mathbf{x})]$$

Theorem 3 [Charikar, S.'17]

Worst case datasets for HBE have support on two points.

- Linearity of expectation: $\mathbb{E}[|H(x)| \,|\, i \in H(x)] = \sum_j \frac{P(i,j \in H(x))}{p_i}$
- Monotonicity: $P(i,j \in H(x)) \leq \min\{p_i, p_j\}$

$$\mathbb{E}[Z_h^2] \leq \sup\left\{ f^\top A f \,\middle|\, \|f\|_1 \leq 1, \|f\|_{w,1} \leq \mu \right\}$$
$$\leq 4 \cdot \|\tilde{A}(\mu, \{p_i\}, \{w_i\})\|_{1,\infty}$$

Quantifies **Compatibility** between $\{w_i\}, \{p_i\}$ at level μ

Variance of HBE

$$\mathbb{E}[Z_h^2] = \sum_{i=1}^{n} \frac{w_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})||i \in H(\mathbf{x})]$$

Theorem 3 [Charikar, S.'17]

Worst case datasets for HBE have support on two points.

- Linearity of expectation: $\mathbb{E}[|H(x)||i \in H(x)] = \sum_j \frac{P(i,j \in H(x))}{p_i}$
- Monotonicity: $P(i,j \in H(x)) \leq \min\{p_i, p_j\}$

$$\mathbb{E}[Z_h^2] \leq \sup\left\{ f^\top A f \,\middle|\, \|f\|_1 \leq 1, \|f\|_{w,1} \leq \mu \right\}$$
$$\leq 4 \cdot \|\tilde{A}(\mu, \{p_i\}, \{w_i\})\|_{1,\infty}$$

Quantifies **Compatibility** between $\{w_i\}, \{p_i\}$ at level μ

Variance of HBE

$$\mathbb{E}[Z_h^2] = \sum_{i=1}^{n} \frac{w_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| | i \in H(\mathbf{x})]$$

Theorem 3 [Charikar, S.'17]

Worst case datasets for HBE have support on two points.

- Linearity of expectation: $\mathbb{E}[|H(x)||i \in H(x)] = \sum_j \frac{P(i,j \in H(x))}{p_i}$
- Monotonicity: $P(i,j \in H(x)) \leq \min\{p_i, p_j\}$

$$\mathbb{E}[Z_h^2] \leq \sup\left\{ f^\top A f \middle| \|f\|_1 \leq 1, \|f\|_{w,1} \leq \mu \right\}$$
$$\leq 4 \cdot \|\tilde{A}(\mu, \{p_i\}, \{w_i\})\|_{1,\infty}$$

Quantifies **Compatibility** between $\{w_i\}, \{p_i\}$ at level μ

Variance of HBE

$$\mathbb{E}[Z_h^2] = \sum_{i=1}^{n} \frac{w_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| | i \in H(\mathbf{x})]$$

Theorem 3 [Charikar, S.'17]

Worst case datasets for HBE have support on two points.

- Linearity of expectation: $\mathbb{E}[|H(x)||i \in H(x)] = \sum_j \frac{P(i,j \in H(x))}{p_i}$
- Monotonicity: $P(i,j \in H(x)) \leq \min\{p_i, p_j\}$

$$\begin{aligned} \mathbb{E}[Z_h^2] &\leq \sup\left\{ f^\top A f \,\middle|\, \|f\|_1 \leq 1, \|f\|_{w,1} \leq \mu \right\} \\ &\leq 4 \cdot \|\tilde{A}(\mu, \{p_i\}, \{w_i\})\|_{1,\infty} \end{aligned}$$

Quantifies **Compatibility** between $\{w_i\}, \{p_i\}$ at level μ

Variance of HBE

$$\mathbb{E}[Z_h^2] = \sum_{i=1}^{n} \frac{w_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| | i \in H(\mathbf{x})]$$

Theorem 3 [Charikar, S.'17]

Worst case datasets for HBE have support on two points.

- Linearity of expectation: $\mathbb{E}[|H(x)||i \in H(x)] = \sum_j \frac{P(i,j \in H(x))}{p_i}$
- Monotonicity: $P(i,j \in H(x)) \leq \min\{p_i, p_j\}$

$$\mathbb{E}[Z_h^2] \leq \sup\left\{ f^\top A f \,\middle|\, \|f\|_1 \leq 1, \|f\|_{w,1} \leq \mu \right\}$$
$$\leq 4 \cdot \|\tilde{A}(\mu, \{p_i\}, \{w_i\})\|_{1,\infty}$$

Quantifies **Compatibility** between $\{w_i\}, \{p_i\}$ at level μ

Scale-free Estimators

We then study (β, M) **scale-free** estimators

$$M^{-1} \cdot k(x,y)^{\beta} \leq p(x,y) \leq M \cdot k(x,y)^{\beta}$$

Theorem 4 [Charikar, S'17]

For any $\beta \in [\frac{1}{2}, 1]$ the variance of *scale-free* estimators is $\leq \mu^2 (\frac{M^3}{\mu^{1-\beta}})$

$$\mathrm{Var} \leq \mu^2 O(\frac{1}{\mu^{\beta}} + \frac{1}{\mu^{1-\beta}}) \Rightarrow \beta^* = \frac{1}{2}$$

Scale-free Estimators

We then study (β, M) **scale-free** estimators

$$M^{-1} \cdot k(x,y)^{\beta} \leq p(x,y) \leq M \cdot k(x,y)^{\beta}$$

Theorem 4 [Charikar, S'17]

For any $\beta \in [\frac{1}{2}, 1]$ the variance of *scale-free* estimators is $\leq \mu^2(\frac{M^3}{\mu^{1-\beta}})$

$$\mathrm{Var} \leq \mu^2 O(\frac{1}{\mu^{\beta}} + \frac{1}{\mu^{1-\beta}}) \Rightarrow \beta^* = \frac{1}{2}$$

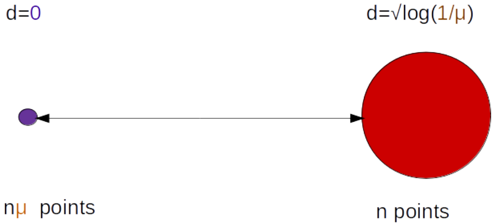
Scale-free Estimators

We then study (β, M) **scale-free** estimators

$$M^{-1} \cdot k(x,y)^{\beta} \leq p(x,y) \leq M \cdot k(x,y)^{\beta}$$

Theorem 4 [Charikar, S'17]

For any $\beta \in [\frac{1}{2}, 1]$ the variance of *scale-free* estimators is $\leq \mu^2(\frac{M^3}{\mu^{1-\beta}})$



$$\mathrm{Var} \leq \mu^2 O(\frac{1}{\mu^{\beta}} + \frac{1}{\mu^{1-\beta}}) \Rightarrow \beta^* = \frac{1}{2}$$

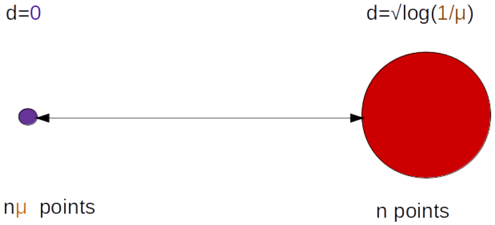
Scale-free Estimators

We then study (β, M) **scale-free** estimators

$$M^{-1} \cdot k(x,y)^{\beta} \leq p(x,y) \leq M \cdot k(x,y)^{\beta}$$

Theorem 4 [Charikar, S'17]

For any $\beta \in [\frac{1}{2}, 1]$ the variance of *scale-free* estimators is $\leq \mu^2(\frac{M^3}{\mu^{1-\beta}})$



$$\mathrm{Var} \leq \mu^2 O(\frac{1}{\mu^{\beta}} + \frac{1}{\mu^{1-\beta}}) \Rightarrow \beta^* = \frac{1}{2}$$

Scale-free Estimators through LSH

Theorem 5 [Charikar, S.'17]

There exist scale-free estimators for the following kernels.

Table : Scale free estimators for KDE using LSH

Kernel	M	LSH
$e^{-\|\mathbf{x}-y\|^2}$	$e^{O(R^{\frac{4}{3}} \log\log n)}$	Ball Carving [AI'06]
$e^{-\|\mathbf{x}-y\|}$	$\sqrt{e}$	Euclidean [Datar et al'04]
$\frac{1}{1+\|\mathbf{x}-y\|_2^p}$	$3^{p/2}$	Euclidean [Datar et al'04]

General framework that applies to other problems!

Scale-free Estimators through LSH

Theorem 5 [Charikar, S.'17]

There exist scale-free estimators for the following kernels.

Table : Scale free estimators for KDE using LSH

Kernel	M	LSH
$e^{-\|\mathbf{x}-y\|^2}$	$e^{O(R^{\frac{4}{3}} \log\log n)}$	Ball Carving [AI'06]
$e^{-\|\mathbf{x}-y\|}$	$\sqrt{e}$	Euclidean [Datar et al'04]
$\frac{1}{1+\|\mathbf{x}-y\|_2^p}$	$3^{p/2}$	Euclidean [Datar et al'04]

General framework that applies to other problems!

Future work

- Partition function approximation with M. Charikar [upcoming]
- **General polynomial kernels** using different techniques with *A. Backurs, M. Charikar, P. Indyk* [upcoming]
- Data-dependent hashing [in progress]

Open problems

- **Open:** Statistical or **Offline** setting
- **Open:** Importance sampling for RFF? [AKMMVZ, ICML'17]
- **Open:** Lower bounds!

Thank You!

psimin@stanford.edu