Integrating Functions via Distance Sensitive Hashing

Paris Siminelakis Moses Charikar

Stanford University

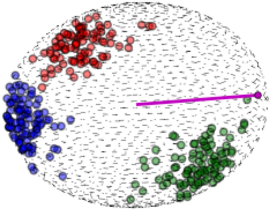
ML Lunch @ Stanford, CA

April 18th, 2018

Problem and Motivation

Problem

$\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_n\} \subset \mathcal{S}^{d-1}$, $\phi : [-1, 1] \to \mathbb{R}$, query $y \in \mathcal{S}^{d-1}$

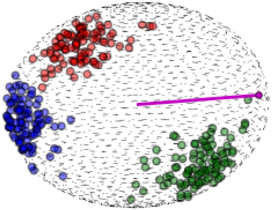


$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

Partition Function Estimation

Problem

$\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_n\} \subset \mathcal{S}^{d-1}$, $\phi : [-1, 1] \to \mathbb{R}$, query $y \in \mathcal{S}^{d-1}$

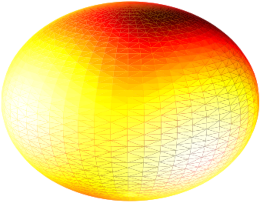


$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

Partition Function Estimation

Problem

$\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_n\} \subset \mathcal{S}^{d-1}$, $\phi : [-1, 1] \rightarrow \mathbb{R}$, query $y \in \mathcal{S}^{d-1}$

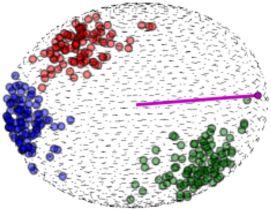


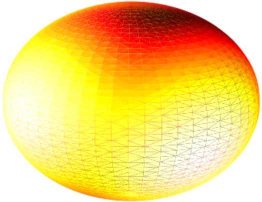
$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

Partition Function Estimation

Problem

$\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_n\} \subset \mathcal{S}^{d-1}$, $\phi : [-1, 1] \to \mathbb{R}$, query $y \in \mathcal{S}^{d-1}$



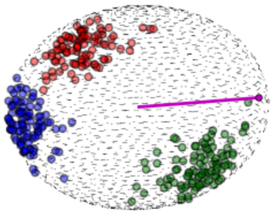


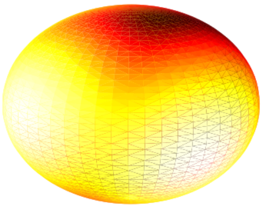
$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

Partition Function Estimation

Problem

$\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_n\} \subset \mathcal{S}^{d-1}$, $\phi : [-1, 1] \to \mathbb{R}$, query $y \in \mathcal{S}^{d-1}$



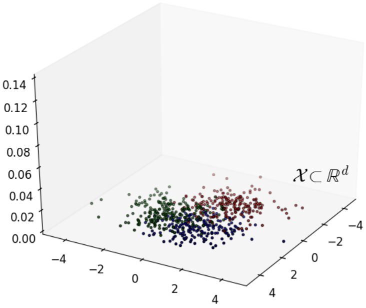


$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

Partition Function Estimation

Kernel Density Estimation

$\mathcal{X} = \{x_1, \ldots, x_n\} \subset r\mathcal{S}^{d-1}$, distribution $\mathcal{D}$, prob. of $y \in \mathcal{S}^{d-1}$?



$$\mathrm{KDE}_{\mathcal{X}}(y) = \frac{1}{n}\sum_{i=1}^{n} K(x, y)$$

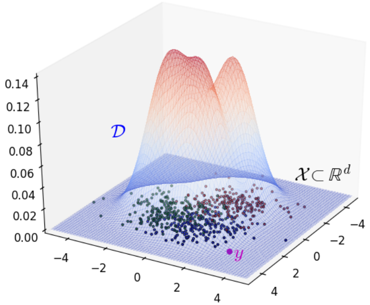
$$K(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}} = e^{\frac{2r^2}{\sigma^2}(\langle x, y\rangle - 1)}$$

outlier detection, clustering, ...

Problem: Data structure that answers queries in sub-linear time?

Kernel Density Estimation

$\mathcal{X} = \{x_1, \ldots, x_n\} \subset r\mathcal{S}^{d-1}$, distribution $\mathcal{D}$, prob. of $y \in \mathcal{S}^{d-1}$?



$$\mathrm{KDE}_{\mathcal{X}}(y) = \frac{1}{n}\sum_{i=1}^{n} K(x, y)$$

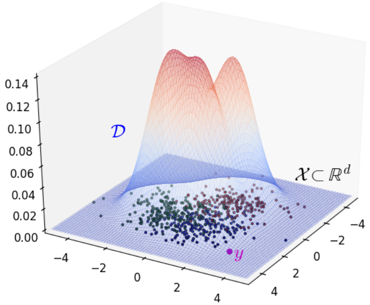
$$K(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}} = e^{\frac{2r^2}{\sigma^2}(\langle x, y\rangle - 1)}$$

outlier detection, clustering, ...

Problem: Data structure that answers queries in sub-linear time?

Kernel Density Estimation

$\mathcal{X} = \{x_1, \ldots, x_n\} \subset r\mathcal{S}^{d-1}$, distribution $\mathcal{D}$, prob. of $y \in \mathcal{S}^{d-1}$?



$$\mathrm{KDE}_{\mathcal{X}}(y) = \frac{1}{n} \sum_{i=1}^{n} K(x, y)$$

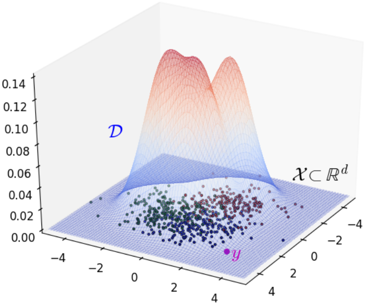
$$K(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}} = e^{\frac{2r^2}{\sigma^2}(\langle x, y\rangle - 1)}$$

outlier detection, clustering, ...

Problem: Data structure that answers queries in sub-linear time?

Kernel Density Estimation

$\mathcal{X} = \{x_1, \dots, x_n\} \subset r\mathcal{S}^{d-1}$, distribution $\mathcal{D}$, prob. of $y \in \mathcal{S}^{d-1}$?



$$\mathrm{KDE}_{\mathcal{X}}(y) = \frac{1}{n}\sum_{i=1}^{n} K(x, y)$$

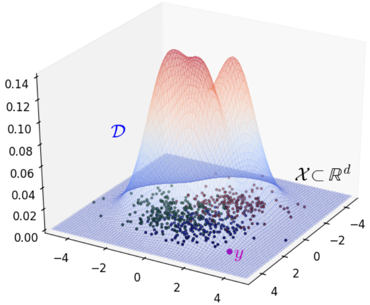
$$K(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}} = e^{\frac{2r^2}{\sigma^2}(\langle x, y\rangle - 1)}$$

outlier detection, clustering, ...

Problem: Data structure that answers queries in sub-linear time?

Kernel Density Estimation

$\mathcal{X} = \{x_1, \ldots, x_n\} \subset r\mathcal{S}^{d-1}$, distribution $\mathcal{D}$, prob. of $y \in \mathcal{S}^{d-1}$?



$$\mathrm{KDE}_{\mathcal{X}}(y) = \frac{1}{n}\sum_{i=1}^{n} K(x, y)$$

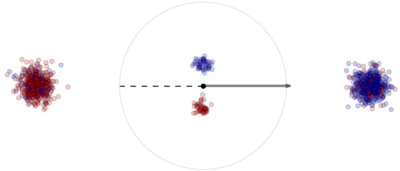
$$K(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}} = e^{\frac{2r^2}{\sigma^2}(\langle x, y\rangle - 1)}$$

outlier detection, clustering, ...

Problem: Data structure that answers queries in sub-linear time?

Empirical Gradient Estimation

$$\{(x_i, s_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{\pm 1\}, \quad \mathcal{L}(\cdot) = \sum_{i=1}^n \ell(\langle s_i x_i, \cdot \rangle), \; \nabla_y \mathcal{L}(y)?$$



Logistic $\ell(\rho) = 1/(1 + exp(-\rho))$

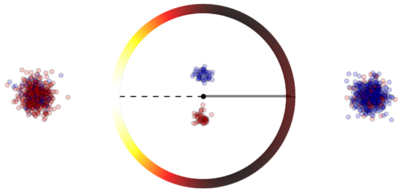
$$\|\nabla_y \ell(\underbrace{\langle s_i x_i, y \rangle}_{\rho_i(y)})\| = \|x_i\| e^{-\log(1+e^{\rho_i})}$$

Discriminative sampling for SG

Problem: Data structure that gets lower-variance SG?

Empirical Gradient Estimation

$$\{(x_i, s_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{\pm 1\}, \quad \mathcal{L}(\cdot) = \sum_{i=1}^n \ell(\langle s_i x_i, \cdot \rangle), \; \nabla_y \mathcal{L}(y)?$$



Logistic $\ell(\rho) = 1/(1 + exp(-\rho))$

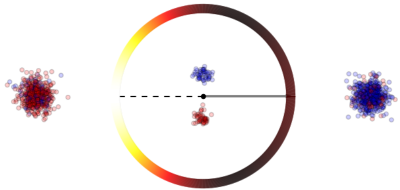
$$\|\nabla_y \ell(\underbrace{\langle s_i x_i, y \rangle}_{\rho_i(y)})\| = \|x_i\| e^{-\log(1+e^{\rho_i})}$$

Discriminative sampling for SG

Problem: Data structure that gets lower-variance SG?

Empirical Gradient Estimation

$$\{(x_i, s_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{\pm 1\}, \quad \mathcal{L}(\cdot) = \sum_{i=1}^n \ell(\langle s_i x_i, \cdot \rangle), \ \nabla_y \mathcal{L}(y)?$$



Logistic $\ell(\rho) = 1/(1 + exp(-\rho))$

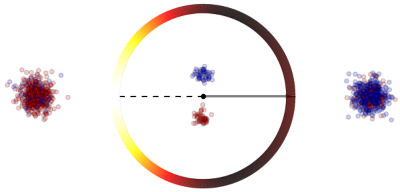
$$\|\nabla_y \ell(\underbrace{\langle s_i x_i, y \rangle}_{\rho_i(y)})\| = \|x_i\| e^{-\log(1+e^{\rho_i})}$$

Discriminative sampling for SG

Problem: Data structure that gets lower-variance SG?

Empirical Gradient Estimation

$$\{(x_i, s_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{\pm 1\}, \quad \mathcal{L}(\cdot) = \sum_{i=1}^n \ell(\langle s_i x_i, \cdot \rangle), \ \nabla_y \mathcal{L}(y)?$$



Logistic $\ell(\rho) = 1/(1 + exp(-\rho))$

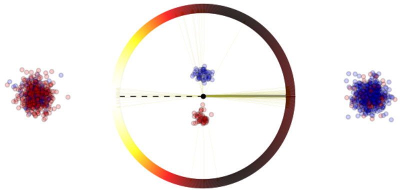
$$\|\nabla_y \ell(\underbrace{\langle s_i x_i, y \rangle}_{\rho_i(y)})\| = \|x_i\| e^{-\log(1+e^{\rho_i})}$$

Discriminative sampling for SG

Problem: Data structure that gets lower-variance SG?

Empirical Gradient Estimation

$$\{(x_i, s_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{\pm 1\}, \quad \mathcal{L}(\cdot) = \sum_{i=1}^n \ell(\langle s_i x_i, \cdot \rangle), \ \nabla_y \mathcal{L}(y)?$$



Logistic $\ell(\rho) = 1/(1 + exp(-\rho))$

$$\|\nabla_y \ell(\underbrace{\langle s_i x_i, y \rangle}_{\rho_i(y)})\| = \|x_i\| e^{-\log(1+e^{\rho_i})}$$

Discriminative sampling for SG

Problem: Data structure that gets lower-variance SG?

Applications of "Partition Function Estimation"

$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

- Kernel Density Estimation
- Robust Optimization: let $\beta = c \log n$

$$\frac{1}{\beta} \log \left(\frac{1}{n} \sum_{i=1}^{n} e^{\beta \phi(\langle x_i, y \rangle)} \right) \approx \max_{i \in [n]} \{ \phi(\langle x_i, y \rangle) \}$$

- Variance reduction for **Stochastic Gradients**

Applications of "Partition Function Estimation"

$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y\rangle)}$$

- Kernel Density Estimation
- Robust Optimization: let $\beta = c \log n$

$$\frac{1}{\beta}\log\left(\frac{1}{n}\sum_{i=1}^{n} e^{\beta\phi(\langle x_i, y\rangle)}\right) \approx \max_{i\in[n]}\{\phi(\langle x_i, y\rangle)\}$$

- Variance reduction for **Stochastic Gradients**

Applications of "Partition Function Estimation"

$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

- Kernel Density Estimation
- Robust Optimization: let $\beta = c \log n$

$$\frac{1}{\beta} \log \left(\frac{1}{n} \sum_{i=1}^{n} e^{\beta \phi(\langle x_i, y \rangle)} \right) \approx \max_{i \in [n]} \{ \phi(\langle x_i, y \rangle) \}$$

- Variance reduction for **Stochastic Gradients**

Applications of "Partition Function Estimation"

$$Z(y) = \sum_{i=1}^{n} e^{\phi(\langle x_i, y \rangle)}$$

- Kernel Density Estimation
- Robust Optimization: let $\beta = c \log n$

$$\frac{1}{\beta} \log \left(\frac{1}{n} \sum_{i=1}^{n} e^{\beta \phi(\langle x_i, y \rangle)} \right) \approx \max_{i \in [n]} \{ \phi(\langle x_i, y \rangle) \}$$

- Variance reduction for **Stochastic Gradients**

Modelling Binary data with Exponential Families

$\mathcal{X} = \{-1, +1\}^d$, parameter vector $y \in r\mathcal{S}^{d-1}$, density on $\mathcal{X}$

$$p_y(x) = \frac{1}{Z(y)} e^{\langle x, y \rangle} = \frac{1}{Z(y)} e^{r\sqrt{d}\left\langle \frac{x}{\|x\|}, \frac{y}{\|y\|} \right\rangle}$$

with $Z(y) = \sum_{x \in \mathcal{X}} e^{\langle x, y \rangle}$ being instrumental for

1. **Sampling:** given y sample $x \sim p_y$
2. **Maximum likelihood:** estimate gradient
3. **Hypothesis testing:** $z_1, \ldots, z_m \sim p_y$, $y = y_1$ or $y = y_2$?

$Z(y)$ requires time 2^d compute exactly

Modelling Binary data with Exponential Families

$\mathcal{X} = \{-1, +1\}^d$, parameter vector $y \in r\mathcal{S}^{d-1}$, density on $\mathcal{X}$

$$p_y(x) = \frac{1}{Z(y)} e^{\langle x, y \rangle} = \frac{1}{Z(y)} e^{r\sqrt{d}\left\langle \frac{x}{\|x\|}, \frac{y}{\|y\|} \right\rangle}$$

with $Z(y) = \sum_{x \in \mathcal{X}} e^{\langle x, y \rangle}$ being instrumental for

1. **Sampling:** given y sample $x \sim p_y$
2. **Maximum likelihood:** estimate gradient
3. **Hypothesis testing:** $z_1, \ldots, z_m \sim p_y$, $y = y_1$ or $y = y_2$?

$Z(y)$ requires time 2^d compute exactly

Modelling Binary data with Exponential Families

$\mathcal{X} = \{-1, +1\}^d$, parameter vector $y \in r\mathcal{S}^{d-1}$, density on $\mathcal{X}$

$$p_y(x) = \frac{1}{Z(y)} e^{\langle x, y \rangle} = \frac{1}{Z(y)} e^{r\sqrt{d} \left\langle \frac{x}{\|x\|}, \frac{y}{\|y\|} \right\rangle}$$

with $Z(y) = \sum_{x \in \mathcal{X}} e^{\langle x, y \rangle}$ being instrumental for

1. **Sampling:** given y sample $x \sim p_y$
2. **Maximum likelihood:** estimate gradient
3. **Hypothesis testing:** $z_1, \ldots, z_m \sim p_y$, $y = y_1$ or $y = y_2$?

$Z(y)$ requires time 2^d compute exactly

Modelling Binary data with Exponential Families

$\mathcal{X} = \{-1, +1\}^d$, parameter vector $y \in r\mathcal{S}^{d-1}$, density on $\mathcal{X}$

$$p_y(x) = \frac{1}{Z(y)} e^{\langle x, y \rangle} = \frac{1}{Z(y)} e^{r\sqrt{d}\left\langle \frac{x}{\|x\|}, \frac{y}{\|y\|} \right\rangle}$$

with $Z(y) = \sum_{x \in \mathcal{X}} e^{\langle x, y \rangle}$ being instrumental for

1. **Sampling:** given y sample $x \sim p_y$
2. **Maximum likelihood:** estimate gradient
3. **Hypothesis testing:** $z_1, \ldots, z_m \sim p_y$, $y = y_1$ or $y = y_2$?

$Z(y)$ requires time 2^d compute exactly

Previous Work and Main Result

Reducing Space through Random Sampling

$L(\phi)$ Lipschitz const, $\phi_{\max} - \phi_{\min} \leq 2L(\phi)$, equiv. estimate:

$$Z(y) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} e^{\phi(\langle x, y \rangle) - \phi_{\max}} \in [e^{-2L(\phi)}, 1]$$

Random Sampling and Median-of-means:

$$m = \underbrace{\left\lceil \frac{6}{\epsilon^2} \exp(2L(\phi)) \right\rceil}_{\text{mean of } n \text{ samples}} \cdot \underbrace{\left\lceil 9 \log(\frac{1}{\delta}) \right\rceil}_{\text{median of } K \text{ means}}$$

For $Z(y) = \mu \in [\Omega(\frac{1}{n}), 1]$ we require $O(\frac{1}{\epsilon^2} \frac{1}{\mu} \log(\frac{1}{\delta}))$ samples.

$$n = \Theta(e^{2L(\phi)}) \text{ and } \mu \in [n^{-1}, 1].$$

Reducing Space through Random Sampling

$L(\phi)$ Lipschitz const, $\phi_{\max} - \phi_{\min} \leq 2L(\phi)$, equiv. estimate:

$$Z(y) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} e^{\phi(\langle x, y \rangle) - \phi_{\max}} \in [e^{-2L(\phi)}, 1]$$

Random Sampling and Median-of-means:

$$m = \underbrace{\left\lceil \frac{6}{\epsilon^2} \exp(2L(\phi)) \right\rceil}_{\text{mean of } n \text{ samples}} \cdot \underbrace{\left\lceil 9 \log(\frac{1}{\delta}) \right\rceil}_{\text{median of } K \text{ means}}$$

For $Z(y) = \mu \in [\Omega(\frac{1}{n}), 1]$ we require $O(\frac{1}{\epsilon^2} \frac{1}{\mu} \log(\frac{1}{\delta}))$ samples.

$$n = \Theta(e^{2L(\phi)}) \text{ and } \mu \in [n^{-1}, 1].$$

Reducing Space through Random Sampling

$L(\phi)$ Lipschitz const, $\phi_{\max} - \phi_{\min} \leq 2L(\phi)$, equiv. estimate:

$$Z(y) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} e^{\phi(\langle x,y \rangle) - \phi_{\max}} \in [e^{-2L(\phi)}, 1]$$

Random Sampling and Median-of-means:

$$m = \underbrace{\left\lceil \frac{6}{\epsilon^2} \exp(2L(\phi)) \right\rceil}_{\text{mean of } n \text{ samples}} \cdot \underbrace{\left\lceil 9 \log(\frac{1}{\delta}) \right\rceil}_{\text{median of } K \text{ means}}$$

For $Z(y) = \mu \in [\Omega(\frac{1}{n}), 1]$ we require $O(\frac{1}{\epsilon^2} \frac{1}{\mu} \log(\frac{1}{\delta}))$ samples.

$$n = \Theta(e^{2L(\phi)}) \text{ and } \mu \in [n^{-1}, 1].$$

Previous Work

Mussman and Ermon [ICML'16] employed a series of reductions:

$$(1 \pm \epsilon) Z^{-1}(y) \to \log(1+\epsilon) - \mathrm{MIPS} \to (1 + \frac{\epsilon}{r^2}) - \mathrm{ANN}$$

to solve the linear case for $x, y \in r\mathcal{S}^{d-1}$ where $L(\phi) = r^2$.

1. **Empirically outperforms** baseline methods.
2. ANN data structure by Andoni-Razenshteyn [STOC'15] :

$$n^{1-O(\frac{\epsilon}{r^2})} = e^{2L(\phi) - O(\epsilon)}$$

3. This is tight in worst case Andoni et al. [SODA'17]

Mussmann, Chen, Ermon [UAI'17]: lazy evaluation ⇒ speed up.

Previous Work

Mussman and Ermon [ICML'16] employed a series of reductions:

$$(1 \pm \epsilon) Z^{-1}(y) \to \log(1+\epsilon) - \mathrm{MIPS} \to (1 + \frac{\epsilon}{r^2}) - \mathrm{ANN}$$

to solve the linear case for $x, y \in r\mathcal{S}^{d-1}$ where $L(\phi) = r^2$.

1. **Empirically outperforms** baseline methods.
2. ANN data structure by Andoni-Razenshteyn [STOC'15] :

$$n^{1-O(\frac{\epsilon}{r^2})} = e^{2L(\phi) - O(\epsilon)}$$

3. This is tight in worst case Andoni et al. [SODA'17]

Mussmann, Chen, Ermon [UAI'17]: lazy evaluation $\Rightarrow$ speed up.

Previous Work

Mussman and Ermon [ICML'16] employed a series of reductions:

$$(1 \pm \epsilon) Z^{-1}(y) \to \log(1+\epsilon) - \text{MIPS} \to (1 + \frac{\epsilon}{r^2}) - \text{ANN}$$

to solve the linear case for $x, y \in r\mathcal{S}^{d-1}$ where $L(\phi) = r^2$.

1. **Empirically outperforms** baseline methods.
2. ANN data structure by Andoni-Razenshteyn [STOC'15] :

$$n^{1-O(\frac{\epsilon}{r^2})} = e^{2L(\phi) - O(\epsilon)}$$

3. This is tight in worst case Andoni et al. [SODA'17]

Mussmann, Chen, Ermon [UAI'17]: lazy evaluation $\Rightarrow$ speed up.

Previous Work

Mussman and Ermon [ICML'16] employed a series of reductions:

$$(1 \pm \epsilon)Z^{-1}(y) \to \log(1+\epsilon) - \mathrm{MIPS} \to (1 + \frac{\epsilon}{r^2}) - \mathrm{ANN}$$

to solve the linear case for $x, y \in r\mathcal{S}^{d-1}$ where $L(\phi) = r^2$.

1. **Empirically outperforms** baseline methods.
2. ANN data structure by Andoni-Razenshteyn [STOC'15] :

$$n^{1-O(\frac{\epsilon}{r^2})} = e^{2L(\phi)-O(\epsilon)}$$

3. This is tight in worst case Andoni et al. [SODA'17]

Mussmann, Chen, Ermon [UAI'17]: lazy evaluation $\Rightarrow$ speed up.

Previous Work

Mussman and Ermon [ICML'16] employed a series of reductions:

$$(1 \pm \epsilon) Z^{-1}(y) \rightarrow \log(1+\epsilon) - \mathrm{MIPS} \rightarrow (1 + \frac{\epsilon}{r^2}) - \mathrm{ANN}$$

to solve the linear case for $x, y \in r\mathcal{S}^{d-1}$ where $L(\phi) = r^2$.

1. **Empirically outperforms** baseline methods.
2. ANN data structure by Andoni-Razenshteyn [STOC'15] :

$$n^{1-O(\frac{\epsilon}{r^2})} = e^{2L(\phi) - O(\epsilon)}$$

3. This is tight in worst case Andoni et al. [SODA'17]

Mussmann, Chen, Ermon [UAI'17]: lazy evaluation $\Rightarrow$ speed up.

Main Result

Theorem [S, Charikar'18]

For any convex function ϕ there exists a **data structure** using

- preprocessing time/space $O(\frac{1}{\epsilon^2} e^{L(\phi)} \log(\frac{1}{\delta}) \cdot dn)$
- answers any query y in $O(M_\phi \frac{1}{\epsilon^2} \frac{1}{\sqrt{\mu}} \log(\frac{1}{\delta}) d)$ time.

where $\mu = Z(y)$ and $M_\phi = \exp\left(\{L(\phi)\log(L(\phi))\}^{\frac{2}{3}}\right)$,

Main Result

Theorem [S, Charikar'18]

For any convex function ϕ there exists a **data structure** using

- preprocessing time/space $O(\frac{1}{\epsilon^2}e^{L(\phi)}\log(\frac{1}{\delta})\cdot dn)$
- answers any query y in $O(M_\phi \frac{1}{\epsilon^2}\frac{1}{\sqrt{\mu}}\log(\frac{1}{\delta})d)$ time.

where $\mu = Z(y)$ and $M_\phi = \exp\left(\{L(\phi)\log(L(\phi))\}^{\frac{2}{3}}\right)$,

Assuming $n = \Theta(e^{2L(\phi)})$ and $\mu = \frac{1}{n}$

Method	Space	Query
Random Sampl.	$O(n)$	$O(n)$
MIPS-ANN	$O(n^{2-O(\epsilon)})$	$O(n^{1-O(\epsilon)})$
Ours	$O(n^{\frac{3}{2}+o(1)})$	$O(n^{\frac{1}{2}+o(1)})$

Main Result

Theorem [S, Charikar'18]

For any convex function ϕ there exists a **data structure** using

- preprocessing time/space $O(\frac{1}{\epsilon^2} e^{L(\phi)} \log(\frac{1}{\delta}) \cdot dn)$
- answers any query y in $O(M_\phi \frac{1}{\epsilon^2} \frac{1}{\sqrt{\mu}} \log(\frac{1}{\delta}) d)$ time.

where $\mu = Z(y)$ and $M_\phi = \exp\left(\{L(\phi) \log(L(\phi))\}^{\frac{2}{3}}\right)$,

Kernel	$\phi(\rho)$	$L(\phi)$
$e^{\langle x,y \rangle}$	$r^2 \rho$	r^2
$e^{-\|x-y\|_2^2}$	$2r^2(\rho - 1)$	$2r^2$
$(\|x-y\|_2^2 + 1)^{-1}$	$-\log(1 + (1-\rho)2r^2)$	$2r^2$
$\frac{1}{1+e^{-\langle x,y \rangle}}$	$-\log(1 + e^{-r^2 \rho})$	r^2
$(\langle x, y \rangle + cr^2)^{-k}$	$-k \log(r^2(\rho + c))$	$\frac{k}{c-1}$

Extensions

We show two reductions:

1. **Euclidean space** $\rightarrow$ Sphere:

   partition in thin annuli and round vectors a la Andoni-Razhenshteyn [STOC'15]

2. **Vector sums** $\rightarrow$ sum of norms:

   Estimate vector sums at least as well as estimating the sum of norms

Data Structure

Ingredients:

- T hashing schemes $\mathcal{H}_1, \ldots, \mathcal{H}_T$ with collision probabilities $p_t(\langle x, y \rangle) = \mathbb{P}_{h \sim \mathcal{H}_t}[h(x) = h(y)]$. [Distance Sensitive]
- T weight functions $w_1, \ldots, w_T$ such that $e^{\phi(\langle x,y \rangle)} = \sum_t w_t(\langle x, y \rangle)$ for all x, y. [Convex Decomposition]

Preprocessing:

- Sample $h_t \sim \mathcal{H}_t$ and create hash table H_t for dataset X.

Query Algorithm:

- Let X_t be a uniform sample from $H_t(q)$ (hash bucket of q)
- Form an unbiased estimator by reweighting:

$$Z(y) = \sum_{t=1}^{T} \frac{w_t(\langle X_t, y \rangle)}{p_t(\langle X_t, y \rangle)} |H_t(y)|$$

Data Structure

Ingredients:

- T hashing schemes $\mathcal{H}_1, \ldots, \mathcal{H}_T$ with collision probabilities $p_t(\langle x, y\rangle) = \mathbb{P}_{h \sim \mathcal{H}_t}[h(x) = h(y)]$. [Distance Sensitive]
- T weight functions $w_1, \ldots, w_T$ such that $e^{\phi(\langle x,y\rangle)} = \sum_t w_t(\langle x, y\rangle)$ for all x, y. [Convex Decomposition]

Preprocessing:

- Sample $h_t \sim \mathcal{H}_t$ and create hash table H_t for dataset X.

Query Algorithm:

- Let X_t be a uniform sample from $H_t(q)$ (hash bucket of q)
- Form an unbiased estimator by reweighting:

$$Z(y) = \sum_{t=1}^{T} \frac{w_t(\langle X_t, y\rangle)}{p_t(\langle X_t, y\rangle)} |H_t(y)|$$

Data Structure

Ingredients:

- T hashing schemes $\mathcal{H}_1, \ldots, \mathcal{H}_T$ with collision probabilities $p_t(\langle x, y\rangle) = \mathbb{P}_{h\sim\mathcal{H}_t}[h(x) = h(y)]$. [Distance Sensitive]
- T weight functions $w_1, \ldots, w_T$ such that $e^{\phi(\langle x,y\rangle)} = \sum_t w_t(\langle x, y\rangle)$ for all x, y. [Convex Decomposition]

Preprocessing:

- Sample $h_t \sim \mathcal{H}_t$ and create hash table H_t for dataset X.

Query Algorithm:

- Let X_t be a uniform sample from $H_t(q)$ (hash bucket of q)
- Form an unbiased estimator by reweighting:

$$Z(y) = \sum_{t=1}^{T} \frac{w_t(\langle X_t, y\rangle)}{p_t(\langle X_t, y\rangle)} |H_t(y)|$$

Multi-resolution HBE

Data-structure: median-of-means on unbiased estimator

$$Z(y) = \sum_{t=1}^{T} \frac{w_t(\langle X_t, y\rangle)}{p_t(\langle X_t, y\rangle)} |H_t(y)|$$

that we call Multi-resolution Hashing-Based-Estimators.

Challenges:

- Specify weighting scheme depending on convex fun ϕ
- Select hashing schemes depending on convex fun ϕ.
- Provably **bound the variance** of the overall estimator.

Proof Ideas

Primer on Importance sampling

Setting: weights $w_1, \ldots, w_n$ e.g. $w_i = K(x_i, y)$,
Goal: approximate $\mu = \sum_{i=1}^{n} w_i$

Importance Sampling
Black box Q, returns index i with probability q_i.

- **Unbiased estimator:** let $I \sim Q$ then $Z_Q = \frac{w_I}{q_I}$

$$\mathbb{E}[Z_Q] = \sum_{i=1}^{n} q_i \frac{w_i}{q_i} = \sum_{i=1}^{n} w_i$$

- **Variance:** controlled by $\mathbb{E}[Z_Q^2] = \sum_{i=1}^{n} \frac{w_i^2}{q_i}$

---

unbiased estimators of low variance and **median of means**

Primer on Importance sampling

Setting: weights $w_1, \ldots, w_n$ e.g. $w_i = K(x_i, y)$,
Goal: approximate $\mu = \sum_{i=1}^n w_i$

Importance Sampling

Black box Q, returns index i with probability q_i.

- **Unbiased estimator:** let $I \sim Q$ then $Z_Q = \frac{w_I}{q_I}$

$$\mathbb{E}[Z_Q] = \sum_{i=1}^{n} q_i \frac{w_i}{q_i} = \sum_{i=1}^{n} w_i$$

- **Variance:** controlled by $\mathbb{E}[Z_Q^2] = \sum_{i=1}^n \frac{w_i^2}{q_i}$

---

unbiased estimators of low variance and **median of means**

Primer on Importance sampling

Setting: weights $w_1, \dots, w_n$ e.g. $w_i = K(x_i, y)$,
Goal: approximate $\mu = \sum_{i=1}^{n} w_i$

Importance Sampling

Black box Q, returns index i with probability q_i.

- **Unbiased estimator:** let $I \sim Q$ then $Z_Q = \frac{w_I}{q_I}$

$$\mathbb{E}[Z_Q] = \sum_{i=1}^{n} q_i \frac{w_i}{q_i} = \sum_{i=1}^{n} w_i$$

- **Variance:** controlled by $\mathbb{E}[Z_Q^2] = \sum_{i=1}^{n} \frac{w_i^2}{q_i}$

---

unbiased estimators of low variance and **median of means**

Primer on Importance sampling

Setting: weights $w_1, \ldots, w_n$ e.g. $w_i = K(x_i, y)$,
Goal: approximate $\mu = \sum_{i=1}^{n} w_i$

Importance Sampling

Black box Q, returns index i with probability q_i.

- **Unbiased estimator:** let $I \sim Q$ then $Z_Q = \frac{w_I}{q_I}$

$$\mathbb{E}[Z_Q] = \sum_{i=1}^{n} q_i \frac{w_i}{q_i} = \sum_{i=1}^{n} w_i$$

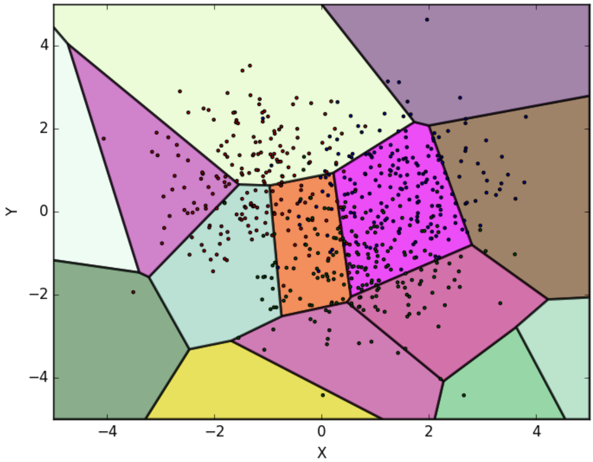
- **Variance:** controlled by $\mathbb{E}[Z_Q^2] = \sum_{i=1}^{n} \frac{w_i^2}{q_i}$

---

unbiased estimators of low variance and **median of means**

Locality Sensitive Hashing

Randomized Space Partitions $\mathbb{P}[h(x) = h(y)] = f(\|x - y\|)$



Algorithmic Framework

Hashing-based-Estimators [Charikar, S.,FOCS'17]:

- Collision probability $p(x, y) = \Theta(\sqrt{K(x, y)})$ then one can get an estimator $\hat{Z}(y)$ with relative variance $O(\frac{1}{\sqrt{\mu}})$.

$$\hat{Z}(y) = \frac{1}{n} \frac{K(X, y)}{p(X, y)} |H(y)|, \qquad X \sim H(y)$$

- **Scale-free** Property is hard to attain.

Algorithmic Framework

Hashing-based-Estimators [Charikar, S.,FOCS'17]:

- Collision probability $p(x, y) = \Theta(\sqrt{K(x, y)})$ then one can get an estimator $\hat{Z}(y)$ with relative variance $O(\frac{1}{\sqrt{\mu}})$.

$$\hat{Z}(y) = \frac{1}{n}\frac{K(X, y)}{p(X, y)}|H(y)|, \qquad X \sim H(y)$$

- **Scale-free** Property is hard to attain.

Algorithmic Framework

Hashing-based-Estimators [Charikar, S.,FOCS'17]:

- Collision probability $p(x, y) = \Theta(\sqrt{K(x, y)})$ then one can get an estimator $\hat{Z}(y)$ with relative variance $O(\frac{1}{\sqrt{\mu}})$.

$$\hat{Z}(y) = \frac{1}{n}\frac{K(X, y)}{p(X, y)}|H(y)|, \qquad X \sim H(y)$$

- **Scale-free** Property is hard to attain.

Algorithmic Framework

Hashing-based-Estimators [Charikar, S.,FOCS'17]:

- Collision probability $p(x, y) = \Theta(\sqrt{K(x, y)})$ then one can get an estimator $\hat{Z}(y)$ with relative variance $O(\frac{1}{\sqrt{\mu}})$.

$$\hat{Z}(y) = \frac{1}{n} \frac{K(X, y)}{p(X, y)} |H(y)|, \qquad X \sim H(y)$$

- **Scale-free** Property is hard to attain.

Limitations of HBE

Scale-free Property is hard to attain:

$$p(x,y) = \Theta(\sqrt{K(x,y)})$$

- Gaussian, Exponential and "polynomial" using **LSH**.
- Collision prob. that near 0 or $\gg 1$ exhibited the desired (exponential, gaussian or polynomial) decay with distance.
- **Machine Learning** and **Optimization** we care more about Inner Products rather than distance.

Limitations of HBE

Scale-free Property is hard to attain:

$$p(x, y) = \Theta(\sqrt{K(x, y)})$$

- Gaussian, Exponential and "polynomial" using **LSH**.
- Collision prob. that near 0 or $\gg 1$ exhibited the desired (exponential, gaussian or polynomial) decay with distance.
- **Machine Learning** and **Optimization** we care more about Inner Products rather than distance.

Limitations of HBE

Scale-free Property is hard to attain:

$$p(x, y) = \Theta(\sqrt{K(x, y)})$$

- Gaussian, Exponential and "polynomial" using **LSH**.
- Collision prob. that near 0 or $\gg 1$ exhibited the desired (exponential, gaussian or polynomial) decay with distance.
- **Machine Learning** and **Optimization** we care more about Inner Products rather than distance.

Limitations of HBE

Scale-free Property is hard to attain:

$$p(x, y) = \Theta(\sqrt{K(x, y)})$$

- Gaussian, Exponential and “polynomial” using **LSH**.
- Collision prob. that near 0 or $\gg 1$ exhibited the desired (exponential, gaussian or polynomial) decay with distance.
- **Machine Learning** and **Optimization** we care more about Inner Products rather than distance.

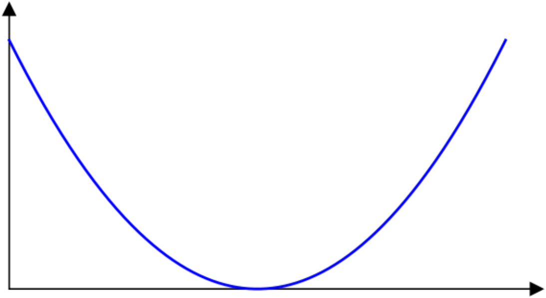
Main contributions

- Generalize results on HBE to **Multi-resolution HBE** .
- Distance Sensitive Hashing on the **Sphere** instead of LSH.
- **Approximation Theory** for Log-convex functions on Sphere.

Intuition

Given a function $w_0 : [-1, 1] \to \mathbb{R}$ want to **approximate**

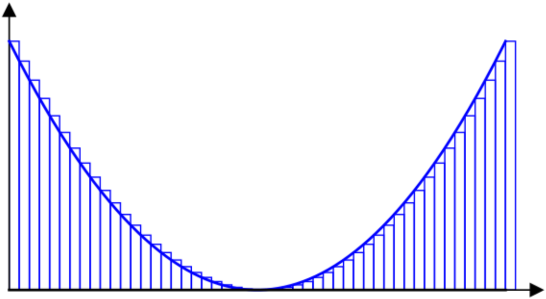
$$\int_{-1}^{1} w_0(\rho) d\rho$$



Intuition

Given a function $w_0 : [-1,1] \to \mathbb{R}$ want to **approximate**

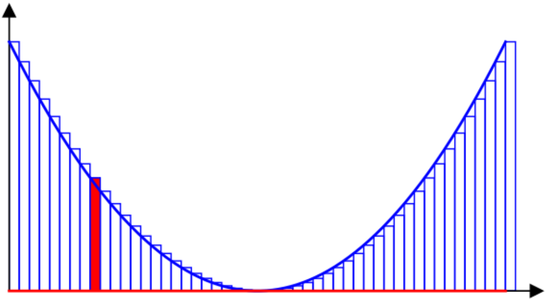
$$\int_{-1}^{1} w_0(\rho) d\rho = \int_{-1}^{1} \left(\sum_{t \in [T]} w_t^*(\rho) \right) d\rho$$



Intuition

Given a function $w_0 : [-1, 1] \to \mathbb{R}$ want to **approximate**

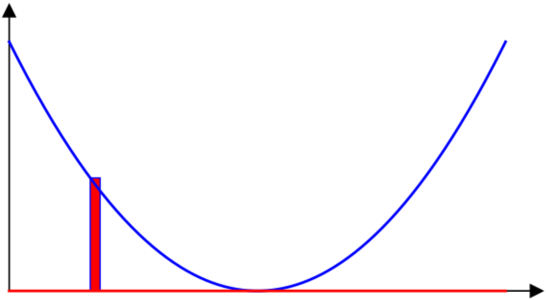
$$\int_{-1}^{1} w_0(\rho) d\rho = \int_{-1}^{1} \left(\sum_{t \in [T]} w_t^*(\rho) \right) d\rho$$



Intuition

Given a function $w_0 : [-1, 1] \to \mathbb{R}$ want to **approximate**

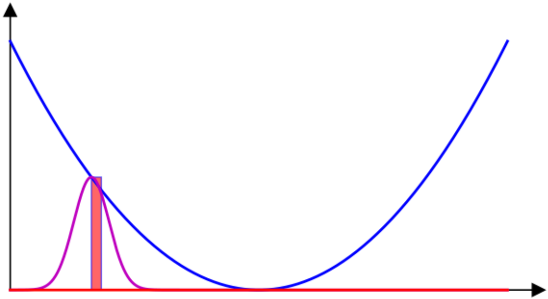
$$\int_{-1}^{1} w_0(\rho) d\rho = \int_{-1}^{1} \left(\sum_{t \in [T]} w_t^*(\rho) \right) d\rho$$



Intuition

Given a function $w_0 : [-1, 1] \to \mathbb{R}$ want to **approximate**

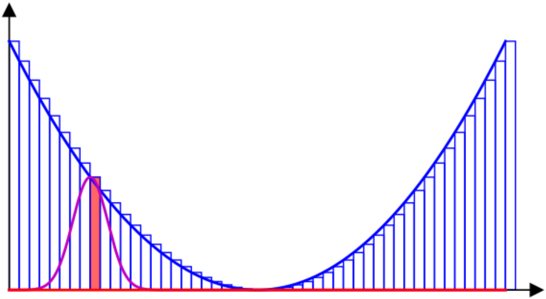
$$\int_{-1}^{1} w_0(\rho) d\rho = \int_{-1}^{1} \left(\sum_{t \in [T]} w_t(\rho) \right) d\rho$$



Intuition

Given a function $w_0 : [-1, 1] \to \mathbb{R}$ want to **approximate**

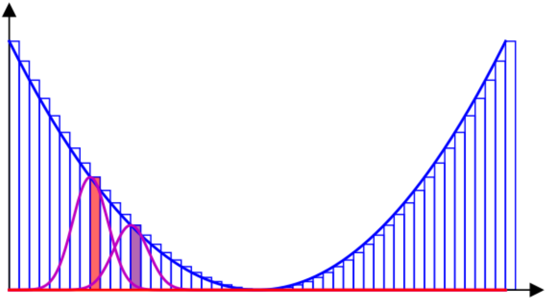
$$\int_{-1}^{1} w_0(\rho) d\rho = \int_{-1}^{1} \left(\sum_{t \in [T]} w_t(\rho) \right) d\rho$$



Intuition

Given a function $w_0 : [-1,1] \to \mathbb{R}$ want to **approximate**

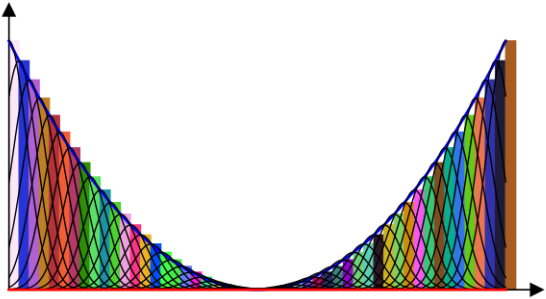
$$\int_{-1}^{1} w_0(\rho)d\rho = \int_{-1}^{1} \left(\sum_{t\in[T]} w_t(\rho) \right) d\rho$$



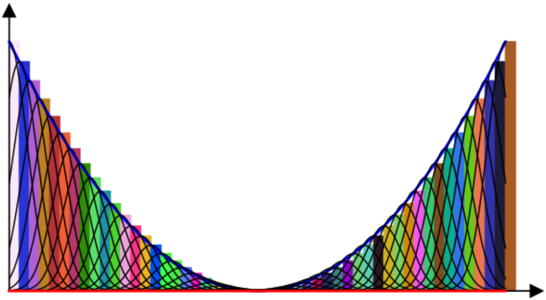
Intuition

Given a function $w_0 : [-1,1] \to \mathbb{R}$ want to **approximate**

$$\int_{-1}^{1} w_0(\rho)d\rho = \int_{-1}^{1} \left(\sum_{t\in[T]} w_t(\rho) \right) d\rho$$



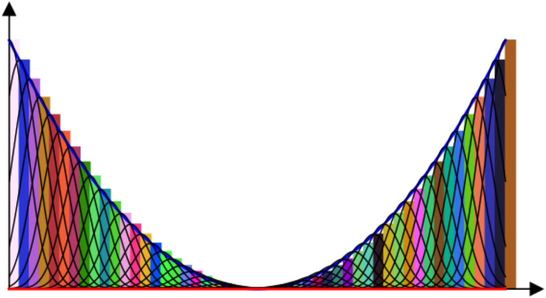
Intuition



$$w_0(\rho) = \sum_{t \in [T]} w_t(\rho)$$

- Find appropriate hashing probabilities $\{p_t\}_{t\in[T]}$.
- Design a **HBE for each** w_t (Multi-resolution HBE)
- Bound the variance of resulting estimators.

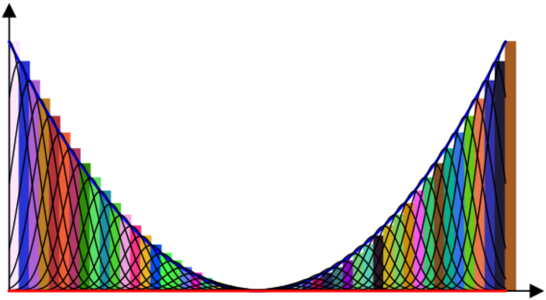
Intuition



$$w_0(\rho) = \sum_{t \in [T]} w_t(\rho)$$

- Find appropriate hashing probabilities $\{p_t\}_{t \in [T]}$.
- Design a **HBE for each** w_t (Multi-resolution HBE)
- Bound the variance of resulting estimators.

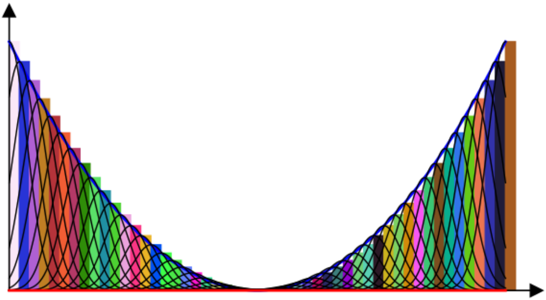
Intuition



$$w_0(\rho) = \sum_{t \in [T]} w_t(\rho)$$

- Find appropriate hashing probabilities $\{p_t\}_{t \in [T]}$.
- Design a **HBE for each** w_t (Multi-resolution HBE)
- Bound the variance of resulting estimators.

Intuition



$$w_0(\rho) = \sum_{t \in [T]} w_t(\rho)$$

- Find appropriate hashing probabilities $\{p_t\}_{t \in [T]}$.
- Design a **HBE for each** w_t (Multi-resolution HBE)
- Bound the variance of resulting estimators.

Distance Sensitive Hashing [Aumuller et al. 2017]

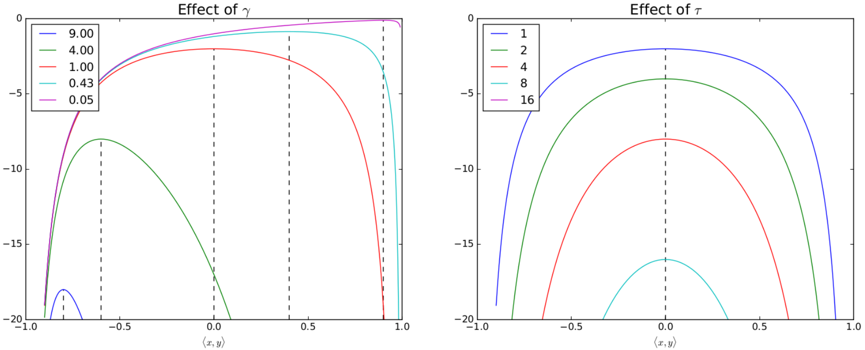
$$g_+, g_- \sim \mathcal{N}(0, I_d),\ \{\langle x, g_+\rangle \geq \tau \wedge \langle x, g_-\rangle \leq -\gamma\tau\},\ e^{O(\tau^2)} \text{ times}$$

$$\log(p_{\gamma,\tau}(\rho)) = \Theta\left(-\left(\frac{1-\rho}{1+\rho} + \gamma^2\frac{1+\rho}{1-\rho}\right)\frac{\tau^2}{2}\right)$$

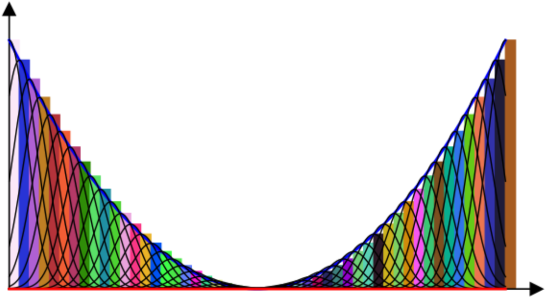
Distance Sensitive Hashing [Aumuller et al. 2017]

$$g_+, g_- \sim \mathcal{N}(0, I_d),\ \{\langle x, g_+\rangle \geq \tau \wedge \langle x, g_-\rangle \leq -\gamma\tau\},\ e^{O(\tau^2)} \text{ times}$$

$$\log(p_{\gamma,\tau}(\rho)) = \Theta\left(-\left(\frac{1-\rho}{1+\rho} + \gamma^2\frac{1+\rho}{1-\rho}\right)\frac{\tau^2}{2}\right)$$



Multi-resolution HBE

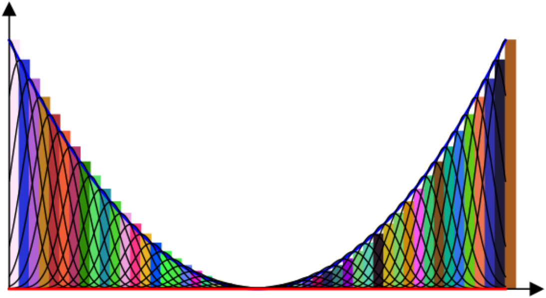


hashing schemes $\{\mathcal{H}_t\}$, coll. prob. $\{p_t\}$, and weight func. $\{w_t\}$.

$$Z_T(y) = \frac{1}{n} \sum_{t \in [T]} \frac{w_t(X_t, y)}{p_t(X_t, y)} |H_t(y)|, \; X_t \sim H_t(y) \text{ for } t \in [T]$$

Technique to bound variance from [Charikar, S., FOCS'17].

Multi-resolution HBE

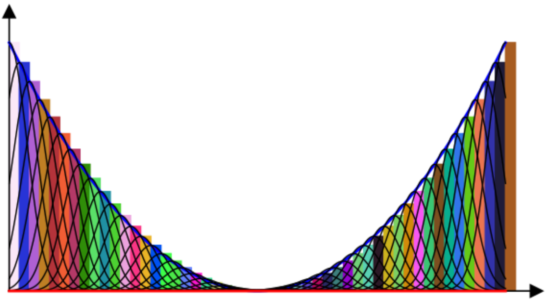


hashing schemes $\{\mathcal{H}_t\}$, coll. prob. $\{p_t\}$, and weight func. $\{w_t\}$.

$$Z_T(y) = \frac{1}{n}\sum_{t\in[T]} \frac{w_t(X_t, y)}{p_t(X_t, y)}|H_t(y)|,\ \ X_t \sim H_t(y) \text{ for } t \in [T]$$

Technique to bound variance from [Charikar, S., FOCS'17].

Multi-resolution HBE



hashing schemes $\{\mathcal{H}_t\}$, coll. prob. $\{p_t\}$, and weight func. $\{w_t\}$.

$$Z_T(y) = \frac{1}{n} \sum_{t \in [T]} \frac{w_t(X_t, y)}{p_t(X_t, y)} |H_t(y)|, \;\; X_t \sim H_t(y) \text{ for } t \in [T]$$

Technique to bound variance from [Charikar, S., FOCS'17].

p^2-weighting scheme

Key **design principle** $w_t(x, y) = \frac{p_t^2(x,y)}{\sum_{t'} p_{t'}^2(x,y)} \cdot w_0(x, y)$ results in

"*Variance of* **Multi-resolution HBE** *is bounded by Variance of HBE with collision probability* $p_*(x, y) = \max_{t \in [T]}\{p_t(x, y)\}$"

Goal: hashing scheme $p_*(x, y) = \Theta(\sqrt{w_0(x, y)}) = \Theta(e^{\frac{1}{2}\phi(x,y)})$.

Fortunately, $\frac{1}{2}\phi(x, y)$ remains convex and lipschitz.

p^2-weighting scheme

Key **design principle** $w_t(x,y) = \frac{p_t^2(x,y)}{\sum_{t'} p_{t'}^2(x,y)} \cdot w_0(x,y)$ results in

> "*Variance of* **Multi-resolution HBE** *is bounded by Variance of HBE with collision probability* $p_*(x,y) = \max_{t\in[T]}\{p_t(x,y)\}$"

Goal: hashing scheme $p_*(x,y) = \Theta(\sqrt{w_0(x,y)}) = \Theta(e^{\frac{1}{2}\phi(x,y)})$.

Fortunately, $\frac{1}{2}\phi(x,y)$ remains convex and lipschitz.

p^2-weighting scheme

Key **design principle** $w_t(x,y) = \frac{p_t^2(x,y)}{\sum_{t'} p_{t'}^2(x,y)} \cdot w_0(x,y)$ results in

> "*Variance of* **Multi-resolution HBE** *is bounded by Variance of HBE with collision probability* $p_*(x,y) = \max_{t\in[T]}\{p_t(x,y)\}$"

Goal: hashing scheme $p_*(x,y) = \Theta(\sqrt{w_0(x,y)}) = \Theta(e^{\frac{1}{2}\phi(x,y)})$.

Fortunately, $\frac{1}{2}\phi(x,y)$ remains convex and lipschitz.

Approximation of Convex Functions I

Goal, pick a set of parameters $\{(\gamma_t, \tau_t)\}_{t \in T}$ such that:

$$\left| \sup_{t \in T} \{\log(p_{\gamma_t,\tau_t}(\rho))\} - \frac{1}{2}\phi(\rho) \right| = O(1)$$

Approximation Theory of Convex Functions:

- Approximate Convex Func. by $O(\sqrt{L(\phi)})$ **Piecewise Linear** (Sandwich Algorithm [Rote'92])
- Approximate Linear func. using $O(\log(L(\phi)))$ hash functions.
- Trade-off evaluation time with approximation, apply result to $\tilde{\phi} = \phi / \{L(\phi)\log(L(\phi))\}^{1/3}$ and tensorize.

Approximation of Convex Functions I

Goal, pick a set of parameters $\{(\gamma_t, \tau_t)\}_{t \in T}$ such that:

$$\left| \sup_{t \in T} \{\log(p_{\gamma_t, \tau_t}(\rho))\} - \frac{1}{2}\phi(\rho) \right| = O(1)$$

Approximation Theory of Convex Functions:

- Approximate Convex Func. by $O(\sqrt{L(\phi)})$ **Piecewise Linear** (Sandwich Algorithm [Rote'92])
- Approximate Linear func. using $O(\log(L(\phi)))$ hash functions.
- Trade-off evaluation time with approximation, apply result to $\tilde{\phi} = \phi / \{L(\phi) \log(L(\phi))\}^{1/3}$ and tensorize.

Approximation of Convex Functions I

Goal, pick a set of parameters $\{(\gamma_t, \tau_t)\}_{t \in T}$ such that:

$$\left| \sup_{t \in T} \{\log(p_{\gamma_t, \tau_t}(\rho))\} - \frac{1}{2}\phi(\rho) \right| = O(1)$$

Approximation Theory of Convex Functions:

- Approximate Convex Func. by $O(\sqrt{L(\phi)})$ **Piecewise Linear** (Sandwich Algorithm [Rote'92])
- Approximate Linear func. using $O(\log(L(\phi)))$ hash functions.
- Trade-off evaluation time with approximation, apply result to $\tilde{\phi} = \phi / \{L(\phi) \log(L(\phi))\}^{1/3}$ and tensorize.

Approximation of Convex Functions I

Goal, pick a set of parameters $\{(\gamma_t, \tau_t)\}_{t \in T}$ such that:

$$\left| \sup_{t \in T} \{\log(p_{\gamma_t, \tau_t}(\rho))\} - \frac{1}{2}\phi(\rho) \right| = O(1)$$

Approximation Theory of Convex Functions:

- Approximate Convex Func. by $O(\sqrt{L(\phi)})$ **Piecewise Linear** (Sandwich Algorithm [Rote'92])
- Approximate Linear func. using $O(\log(L(\phi)))$ hash functions.
- Trade-off evaluation time with approximation, apply result to $\tilde{\phi} = \phi / \{L(\phi) \log(L(\phi))\}^{1/3}$ and tensorize.

Approximation of Convex Functions I

Goal, pick a set of parameters $\{(\gamma_t, \tau_t)\}_{t \in T}$ such that:

$$\left| \sup_{t \in T} \{ \log(p_{\gamma_t, \tau_t}(\rho)) \} - \frac{1}{2}\phi(\rho) \right| = O(1)$$

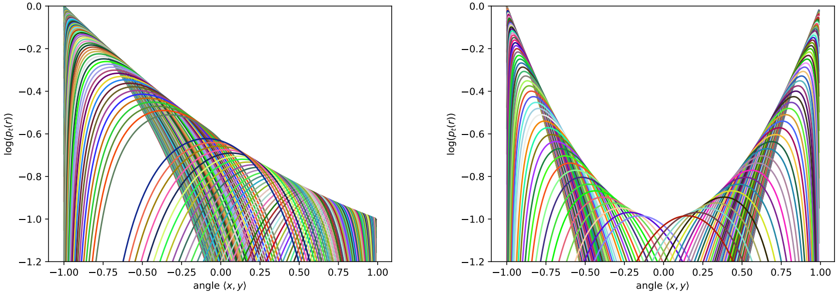
Approximation Theory of Convex Functions:

- Approximate Convex Func. by $O(\sqrt{L(\phi)})$ **Piecewise Linear** (Sandwich Algorithm [Rote'92])
- Approximate Linear func. using $O(\log(L(\phi)))$ hash functions.
- Trade-off evaluation time with approximation, apply result to $\tilde{\phi} = \phi / \{L(\phi) \log(L(\phi))\}^{1/3}$ and tensorize.

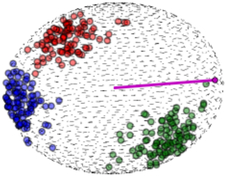
Approximation of Convex Functions II

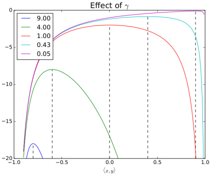
Goal, pick a set of parameters $\{(\gamma_t, \tau_t)\}_{t \in T}$ such that:

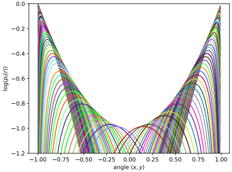
$$\left| \sup_{t \in T} \{\log(p_{\gamma_t,\tau_t}(\rho))\} - \frac{1}{2}\phi(\rho) \right| = O(1)$$



Recap







- Partition Function Estimation via Distance Sensitive Hashing.
- Improve upon state of the art by $\sqrt{n}$ factor.
- **Multi-resolution HBE** and Log-Convex Functions.

Future Work

- Design and implement more practical Hashing Schemes.
- Applications in **Optimization and Learning**.

Thank You!

psimin@stanford.edu