Hashing-Based-Estimators for Accelerating Machine Learning Primitives

Paris Siminelakis

Stanford University

Dawn Seminar @ Stanford, CA

Dec 13, 2017

Outline of the talk

Part 1

1. Machine Learning Primitives:
   - **Kernel Density Estimation**
   - **Partition Function Estimation**
   - **Stochastic Gradient**
2. Importance sampling

Part 2

1. Hashing-Based-Estimators (HBE)
2. Extensions

Outline of the talk

Part 1

1. Machine Learning Primitives:
   - **Kernel Density Estimation**
   - **Partition Function Estimation**
   - **Stochastic Gradient**
2. Importance sampling

Part 2

1. Hashing-Based-Estimators (HBE)
2. Extensions

Outline of the talk

Part 1

1. Machine Learning Primitives:
   - **Kernel Density Estimation**
   - **Partition Function Estimation**
   - **Stochastic Gradient**
2. Importance sampling

Part 2

1. Hashing-Based-Estimators (HBE)
2. Extensions

Outline of the talk

Part 1

1. Machine Learning Primitives:
   - **Kernel Density Estimation**
   - **Partition Function Estimation**
   - **Stochastic Gradient**
2. Importance sampling

Part 2

1. Hashing-Based-Estimators (HBE)
2. Extensions

Outline of the talk

Part 1

1. Machine Learning Primitives:
   - **Kernel Density Estimation**
   - **Partition Function Estimation**
   - **Stochastic Gradient**
2. Importance sampling

Part 2

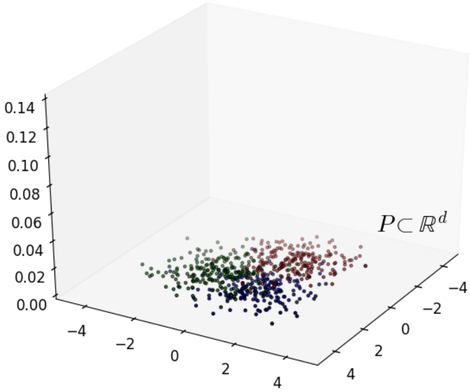
1. Hashing-Based-Estimators (HBE)
2. Extensions

Part 1:

Machine Learning Primitives

Density Estimation

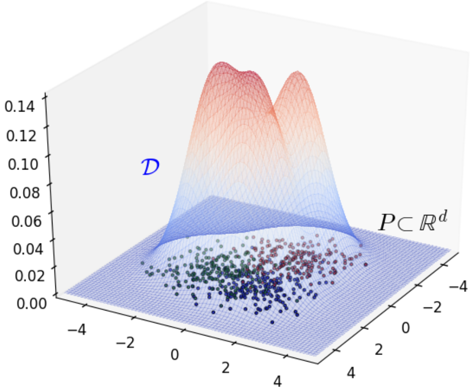
Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?



Non-parametric

Density Estimation

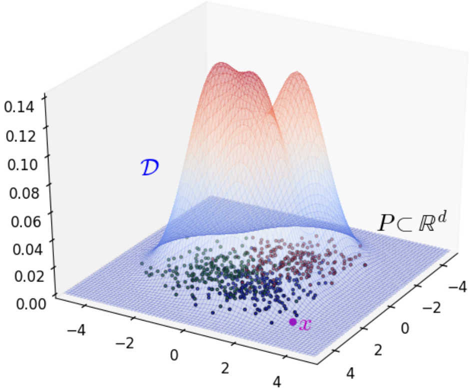
Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?



Non-parametric

Density Estimation

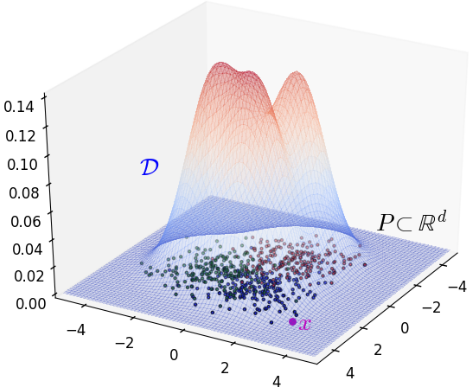
Given $\mathbf{P} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^d$ sampled from $\mathcal{D}$, what is the *probability* of a point $\mathbf{x} \in \mathbb{R}^d$?



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

Basic idea:

- Assign high value to “dense” regions of the space
- Assign low value to “sparse” regions

Kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, bandwidth $\sigma > 0$

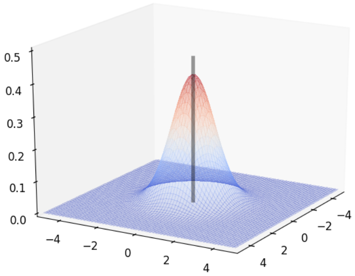
- Gaussian
  $k_\sigma(x, y) = \exp(-\|x - y\|^2/\sigma^2)$
- Exponential
  $k_\sigma(x, y) = \exp(-\|x - y\|_2/\sigma)$
- **Generalized t-student**
  $k_\sigma(x, y) = \frac{1}{1+\|x-y\|^t/\sigma^t}$

Kernel Density Estimation

Basic idea:

- Assign high value to "dense" regions of the space
- Assign low value to "sparse" regions

Kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, bandwidth $\sigma > 0$



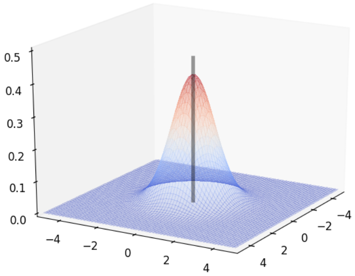
- Gaussian
  $k_\sigma(x, y) = \exp(-\|x - y\|^2/\sigma^2)$
- Exponential
  $k_\sigma(x, y) = \exp(-\|x - y\|_2/\sigma)$
- **Generalized *t*-student**
  $k_\sigma(x, y) = \frac{1}{1+\|x-y\|^t/\sigma^t}$

Kernel Density Estimation

Basic idea:

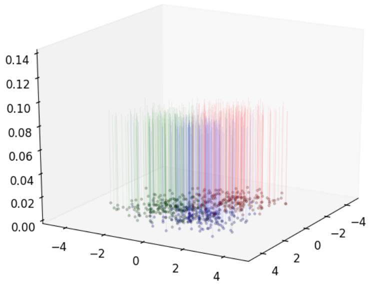
- Assign high value to "dense" regions of the space
- Assign low value to "sparse" regions

Kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$, bandwidth $\sigma > 0$



- Gaussian
  $k_\sigma(x,y) = \exp(-\|x-y\|^2/\sigma^2)$
- Exponential
  $k_\sigma(x,y) = \exp(-\|x-y\|_2/\sigma)$
- **Generalized t-student**
  $k_\sigma(x,y) = \frac{1}{1+\|x-y\|^t/\sigma^t}$

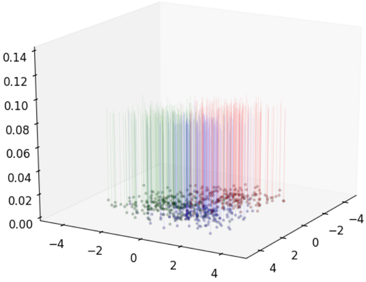
Kernel Density Estimate



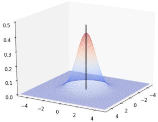
dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

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

Kernel Density Estimate



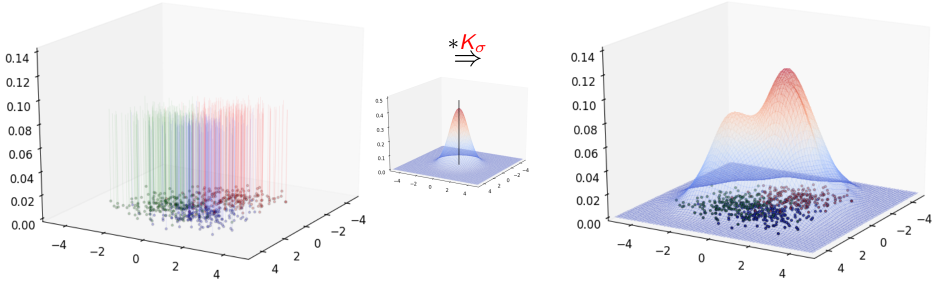
$\overset{*K_\sigma}{\Longrightarrow}$



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

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

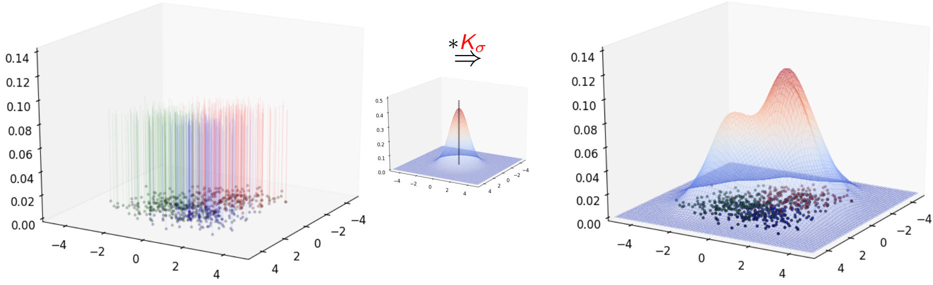
Kernel Density Estimate



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

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

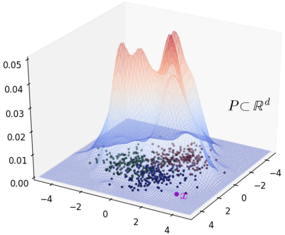
Kernel Density Estimate



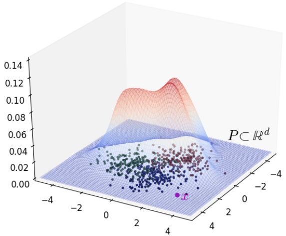
dataset $\mathbf{P} \subset \mathbb{R}^d$, kernel $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, query $\mathbf{x}$

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

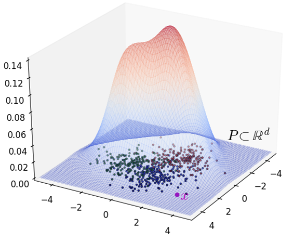
Kernel Density Estimate



$\sigma_1 = \frac{1}{2} \cdot \mathrm{std}$



$\sigma_2 = \frac{3}{4} \cdot \mathrm{std}$



$\sigma_3 = \mathrm{std}$

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

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

Kernel Density Estimation

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

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

- **Statistical problem:** $(\mathbf{P}$, smoothness $\mathcal{D}) \Rightarrow K_\sigma$
- **Computational problem**: $(\mathbf{P}, K_\sigma$, query $x) \Rightarrow \mathrm{KDE}_P(\mathbf{x})$

Problem 1: approximate $\mathrm{KDE}_P(\mathbf{x})$ for any query!

Kernel Density Estimation

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

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

- **Statistical problem:** $(\mathbf{P}$, smoothness $\mathcal{D}) \Rightarrow K_\sigma$
- **Computational problem**: $(\mathbf{P}, K_\sigma$, query $x) \Rightarrow \mathrm{KDE}_P(\mathbf{x})$

Problem 1: approximate $\mathrm{KDE}_P(\mathbf{x})$ for any query!

Kernel Density Estimation

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

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

- **Statistical problem:** ($\mathbf{P}$, smoothness $\mathcal{D}$) $\Rightarrow K_\sigma$
- **Computational problem**: ($\mathbf{P}$, K_σ, query x) $\Rightarrow \mathrm{KDE}_P(\mathbf{x})$

Problem 1: approximate $\mathrm{KDE}_P(\mathbf{x})$ for any query!

Kernel Density Estimation

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

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

- **Statistical problem:** $(\mathbf{P}$, smoothness $\mathcal{D}) \Rightarrow K_\sigma$
- **Computational problem**: $(\mathbf{P}$, K_σ, query $x) \Rightarrow \mathrm{KDE}_P(\mathbf{x})$

Problem 1: approximate $\mathrm{KDE}_P(\mathbf{x})$ for any query!

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.

Parition Function

Log-linear models: $\Omega \subset \mathbb{R}^d$, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ (feature), $w \in \mathbb{R}^{d'}$

$$p_w(x) = \frac{1}{Z(w)} e^{\langle w,\ \phi(x) \rangle}$$

Normalizing constant is called the **Partition function**

$$Z(w) = \int_\Omega e^{\langle w, \phi(x) \rangle} dx$$

Discrete approx: $Q = \{y_1, \ldots, y_m\}$, let $\mathrm{PF}_Q(w) = \sum_{i=1}^m e^{\langle w, y_i \rangle}$

Problem 2: fast approximation to $\mathrm{PF}_Q(w)$!

Parition Function

Log-linear models: $\Omega \subset \mathbb{R}^d$, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ (feature), $w \in \mathbb{R}^{d'}$

$$p_w(x) = \frac{1}{Z(w)} e^{\langle w,\ \phi(x) \rangle}$$

Normalizing constant is called the **Partition function**

$$Z(w) = \int_\Omega e^{\langle w, \phi(x) \rangle} dx$$

Discrete approx: $Q = \{y_1, \ldots, y_m\}$, let $\mathrm{PF}_Q(w) = \sum_{i=1}^m e^{\langle w, y_i \rangle}$

Problem 2: fast approximation to $\mathrm{PF}_Q(w)$!

Parition Function

Log-linear models: $\Omega \subset \mathbb{R}^d$, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ (feature), $w \in \mathbb{R}^{d'}$

$$p_w(x) = \frac{1}{Z(w)} e^{\langle w,\ \phi(x) \rangle}$$

Normalizing constant is called the **Partition function**

$$Z(w) = \int_\Omega e^{\langle w, \phi(x) \rangle} dx$$

Discrete approx: $Q = \{y_1, \ldots, y_m\}$, let $\mathrm{PF}_Q(w) = \sum_{i=1}^m e^{\langle w, y_i \rangle}$

Problem 2: fast approximation to $\mathrm{PF}_Q(w)$!

Parition Function

Log-linear models: $\Omega \subset \mathbb{R}^d$, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ (feature), $w \in \mathbb{R}^{d'}$

$$p_w(x) = \frac{1}{Z(w)} e^{\langle w,\ \phi(x)\rangle}$$

Normalizing constant is called the **Partition function**

$$Z(w) = \int_\Omega e^{\langle w, \phi(x)\rangle} dx$$

Discrete approx: $Q = \{y_1, \ldots, y_m\}$, let $\mathrm{PF}_Q(w) = \sum_{i=1}^m e^{\langle w, y_i\rangle}$

Problem 2: fast approximation to $\mathrm{PF}_Q(w)$!

Applications of PFE

- **Hypothesis testing:** $w_1, w_2 \in \mathbb{R}^d$, dataset $\mathbf{P}$, which one to chose?

$$\log\left(\frac{p_{w_1}(\mathbf{P})}{p_{w_2}(\mathbf{P})}\right) = \left\langle w_1 - w_2, \sum_{i=1}^{n} \phi(x_i) \right\rangle - \log\left(\frac{Z(w_1)}{Z(w_2)}\right) \geq t$$

- **Bayesian statistics:** prior π, hyperparameter tuning (Metropolis-Hastings MCMC), similar ratio.
- **Maximum Likelihood:** $L(w) = \log(p_w(\mathbf{P}))$, gradient

$$\nabla_w L(w) \approx \sum_{x \in P} \phi(x) - \frac{1}{Z(w)} \sum_{y \in Q} \phi(y) e^{\langle w, \phi(y) \rangle}$$

Applications of PFE

- **Hypothesis testing:** $w_1, w_2 \in \mathbb{R}^d$, dataset $\mathbf{P}$, which one to chose?

$$\log\left(\frac{p_{w_1}(\mathbf{P})}{p_{w_2}(\mathbf{P})}\right) = \left\langle w_1 - w_2, \sum_{i=1}^{n} \phi(x_i)\right\rangle - \log\left(\frac{Z(w_1)}{Z(w_2)}\right) \geq t$$

- **Bayesian statistics:** prior π, hyperparameter tuning (Metropolis-Hastings MCMC), similar ratio.
- **Maximum Likelihood:** $L(w) = \log(p_w(\mathbf{P}))$, gradient

$$\nabla_w L(w) \approx \sum_{x \in P} \phi(x) - \frac{1}{Z(w)} \sum_{y \in Q} \phi(y) e^{\langle w, \phi(y)\rangle}$$

Applications of PFE

- **Hypothesis testing:** $w_1, w_2 \in \mathbb{R}^d$, dataset $\mathbf{P}$, which one to chose?

$$\log\left(\frac{p_{w_1}(\mathbf{P})}{p_{w_2}(\mathbf{P})}\right) = \left\langle w_1 - w_2, \sum_{i=1}^{n} \phi(x_i) \right\rangle - \log\left(\frac{Z(w_1)}{Z(w_2)}\right) \geq t$$

- **Bayesian statistics:** prior π, hyperparameter tuning (Metropolis-Hastings MCMC), similar ratio.
- **Maximum Likelihood:** $L(w) = \log(p_w(\mathbf{P}))$, gradient

$$\nabla_w L(w) \approx \sum_{x \in P} \phi(x) - \frac{1}{Z(w)} \sum_{y \in Q} \phi(y) e^{\langle w, \phi(y) \rangle}$$

Applications of PFE

- **Hypothesis testing:** $w_1, w_2 \in \mathbb{R}^d$, dataset **P**, which one to chose?

$$\log\left(\frac{p_{w_1}(\mathbf{P})}{p_{w_2}(\mathbf{P})}\right) = \left\langle w_1 - w_2, \sum_{i=1}^{n} \phi(x_i) \right\rangle - \log\left(\frac{Z(w_1)}{Z(w_2)}\right) \geq t$$

- **Bayesian statistics:** prior π, hyperparameter tuning (Metropolis-Hastings MCMC), similar ratio.
- **Maximum Likelihood:** $L(w) = \log(p_w(\mathbf{P}))$, gradient

$$\nabla_w L(w) \approx \sum_{x \in P} \phi(x) - \frac{1}{Z(w)} \sum_{y \in Q} \phi(y) e^{\langle w, \phi(y) \rangle}$$

Empirical Risk Minimization

Logistic Regression features $x_1, \dots, x_n \in \mathbb{R}^d$, labels $y_1, \dots, y_n \in \{-1, +1\}$, find $w \in \mathbb{R}^d$:

$$\min \qquad L(w) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i \langle w, x_i \rangle}\right)$$

Empirical Risk Minimization loss function $\ell(\langle w, y_i x_i \rangle)$, find w:

$$\min \qquad L(w) = \sum_{i=1}^{n} \ell(\langle w, y_i x_i \rangle)$$

Empirical Risk Minimization

Logistic Regression features $x_1, \dots, x_n \in \mathbb{R}^d$, labels $y_1, \dots, y_n \in \{-1, +1\}$, find $w \in \mathbb{R}^d$:

$$\min \qquad L(w) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i \langle w, x_i \rangle}\right)$$

Empirical Risk Minimization loss function $\ell(\langle w, y_i x_i \rangle)$, find w:

$$\min \qquad L(w) = \sum_{i=1}^{n} \ell(\langle w, y_i x_i \rangle)$$

Stochastic Gradient

Let $r_i = \langle w, y_i x_i \rangle$, then $\nabla L(w) = \sum_{i=1}^{n} \left\{ y_i x_i \cdot \ell'(r_i) \right\}$

- **Gradient estimation:** $I \sim [n]$, let $\hat{g} = x_I y_I \ell'(r_I)$

$$\mathbb{E}[\hat{g}] = \frac{1}{n} \sum_{i=1}^{n} x_i y_i \ell'(r_i) = \frac{1}{n} \nabla_w L(w)$$

- **Variance:** assuming $\|x_i\|^2 = \mathrm{const}$

$$\mathbb{E}\|\hat{g}\|^2 = \frac{c^2}{n} \sum_{i=1}^{n} (\ell'(r_i))^2$$

Lower variance $\Rightarrow$ faster convergence, better generaliz. [HRS'15]

Problem 3: Find estimator of gradient with Lower Variance!

Stochastic Gradient

Let $r_i = \langle w, y_i x_i \rangle$, then $\nabla L(w) = \sum_{i=1}^{n} \left\{ y_i x_i \cdot \ell'(r_i) \right\}$

- **Gradient estimation:** $I \sim [n]$, let $\hat{g} = x_I y_I \ell'(r_I)$

$$\mathbb{E}[\hat{g}] = \frac{1}{n} \sum_{i=1}^{n} x_i y_i \ell'(r_i) = \frac{1}{n} \nabla_w L(w)$$

- **Variance:** assuming $\|x_i\|^2 = \mathrm{const}$

$$\mathbb{E}\|\hat{g}\|^2 = \frac{c^2}{n} \sum_{i=1}^{n} (\ell'(r_i))^2$$

Lower variance $\Rightarrow$ faster convergence, better generaliz. [HRS'15]

Problem 3: Find estimator of gradient with Lower Variance!

Stochastic Gradient

Let $r_i = \langle w, y_i x_i \rangle$, then $\nabla L(w) = \sum_{i=1}^{n} \left\{ y_i x_i \cdot \ell'(r_i) \right\}$

- **Gradient estimation:** $I \sim [n]$, let $\hat{g} = x_I y_I \ell'(r_I)$

$$\mathbb{E}[\hat{g}] = \frac{1}{n} \sum_{i=1}^{n} x_i y_i \ell'(r_i) = \frac{1}{n} \nabla_w L(w)$$

- **Variance:** assuming $\|x_i\|^2 = \mathrm{const}$

$$\mathbb{E}\|\hat{g}\|^2 = \frac{c^2}{n} \sum_{i=1}^{n} (\ell'(r_i))^2$$

Lower variance $\Rightarrow$ faster convergence, better generaliz. [HRS'15]

Problem 3: Find estimator of gradient with Lower Variance!

Stochastic Gradient

Let $r_i = \langle w, y_i x_i \rangle$, then $\nabla L(w) = \sum_{i=1}^{n} \left\{ y_i x_i \cdot \ell'(r_i) \right\}$

- **Gradient estimation:** $I \sim [n]$, let $\hat{g} = x_I y_I \ell'(r_I)$

$$\mathbb{E}[\hat{g}] = \frac{1}{n} \sum_{i=1}^{n} x_i y_i \ell'(r_i) = \frac{1}{n} \nabla_w L(w)$$

- **Variance:** assuming $\|x_i\|^2 = \mathrm{const}$

$$\mathbb{E}\|\hat{g}\|^2 = \frac{c^2}{n} \sum_{i=1}^{n} (\ell'(r_i))^2$$

Lower variance $\Rightarrow$ faster convergence, better generaliz. [HRS'15]

Problem 3: Find estimator of gradient with Lower Variance!

Stochastic Gradient

Let $r_i = \langle w, y_i x_i \rangle$, then $\nabla L(w) = \sum_{i=1}^{n} \left\{ y_i x_i \cdot \ell'(r_i) \right\}$

- **Gradient estimation:** $I \sim [n]$, let $\hat{g} = x_I y_I \ell'(r_I)$

$$\mathbb{E}[\hat{g}] = \frac{1}{n} \sum_{i=1}^{n} x_i y_i \ell'(r_i) = \frac{1}{n} \nabla_w L(w)$$

- **Variance:** assuming $\|x_i\|^2 = \mathrm{const}$

$$\mathbb{E}\|\hat{g}\|^2 = \frac{c^2}{n} \sum_{i=1}^{n} (\ell'(r_i))^2$$

Lower variance $\Rightarrow$ faster convergence, better generaliz. [HRS'15]

Problem 3: Find estimator of gradient with Lower Variance!

Three Problems: Structured Sums

Given dataset $P = \{x_1, \ldots, x_n\}$ and query $w \in \mathbb{R}^d$:

- **Problem 1:** Kernel Density Estimation

$$\mathrm{KDE}_P(w) = \frac{1}{n} \sum_{i=1}^{n} K(w, x_i)$$

- **Problem 2:** Partition Function Estimation

$$\mathrm{PF}_P(w) = \frac{1}{n} \sum_{i=1}^{n} e^{\langle w, x_i \rangle}$$

- **Problem 3:** Variance reduction in Stochastic Gradient

$$\nabla_w L(w) = \sum_{i=1}^{n} \{y_i x_i \ell'(r_i)\}$$

Unbiased estimators and Median-of-means

Let μ be the quantity we wish to approximate.

- **Unbiased estimator:** random variable Z with $\mathbb{E}[Z] = \mu$
- **Variance bound:** let $V > 0$ such that $\mathbb{E}[Z^2] \leq V \cdot \mathbb{E}[Z]^2$

Median-of-means

- **Means** of $\frac{6}{\epsilon^2} V$ independent realizations $Z^{(i)}$
- **Median** of $9 \log(\frac{1}{\delta})$ such means

$$\mathbb{P}[|\hat{Z} - \mu| \leq \epsilon\mu] \geq 1 - \delta$$

Goal: design unbiased estimators with small variance!

Unbiased estimators and Median-of-means

Let μ be the quantity we wish to approximate.

- **Unbiased estimator:** random variable Z with $\mathbb{E}[Z] = \mu$
- **Variance bound:** let $V > 0$ such that $\mathbb{E}[Z^2] \le V \cdot \mathbb{E}[Z]^2$

Median-of-means

- **Means** of $\frac{6}{\epsilon^2} V$ independent realizations $Z^{(i)}$
- **Median** of $9\log(\frac{1}{\delta})$ such means

$$\mathbb{P}[|\hat{Z} - \mu| \le \epsilon\mu] \ge 1 - \delta$$

Goal: design unbiased estimators with small variance!

Unbiased estimators and Median-of-means

Let μ be the quantity we wish to approximate.

- **Unbiased estimator:** random variable Z with $\mathbb{E}[Z] = \mu$
- **Variance bound:** let $V > 0$ such that $\mathbb{E}[Z^2] \leq V \cdot \mathbb{E}[Z]^2$

Median-of-means

- **Means** of $\frac{6}{\epsilon^2} V$ independent realizations $Z^{(i)}$
- **Median** of $9 \log(\frac{1}{\delta})$ such means

$$\mathbb{P}[|\hat{Z} - \mu| \leq \epsilon\mu] \geq 1 - \delta$$

Goal: design unbiased estimators with small variance!

Reducing the Variance

All of these problems have a common characteristic:

- KDE: points closer to w have larger contribution!
- PFE: points aligned with w have larger contribution
- **SG:** points where $\ell'(r_i)$ is larger have **high norm** gradients.

Importance sampling: sample proportional to the importance!

General technique to implement such schemes!

Reducing the Variance

All of these problems have a common characteristic:

- KDE: points closer to w have larger contribution!
- PFE: points aligned with w have larger contribution
- **SG:** points where $\ell'(r_i)$ is larger have **high norm** gradients.

Importance sampling: sample proportional to the importance!

General technique to implement such schemes!

Reducing the Variance

All of these problems have a common characteristic:

- KDE: points closer to w have larger contribution!
- PFE: points aligned with w have larger contribution
- **SG:** points where $\ell'(r_i)$ is larger have **high norm** gradients.

Importance sampling: sample proportional to the importance!

General technique to implement such schemes!

Primer on Importance sampling

Setting: weights $u_1, \ldots, u_n$ e.g. $u_i = K(w, x_i)$,
Goal: approximate $\mu = \sum_{i=1}^{n} u_i$

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

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

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

- **Variance:** controlled by the quantity

$$\mathbb{E}[Z_Q^2] = \sum_{i=1}^{n} \frac{u_i^2}{q_i}$$

Primer on Importance sampling

Setting: weights $u_1, \ldots, u_n$ e.g. $u_i = K(w, x_i)$,
Goal: approximate $\mu = \sum_{i=1}^{n} u_i$

Importance Sampling

Black box Q, returns index i with probability q_i.

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

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

- **Variance:** controlled by the quantity

$$\mathbb{E}[Z_Q^2] = \sum_{i=1}^{n} \frac{u_i^2}{q_i}$$

Primer on Importance sampling

Setting: weights $u_1, \dots, u_n$ e.g. $u_i = K(w, x_i)$,
Goal: approximate $\mu = \sum_{i=1}^{n} u_i$

Importance Sampling

Black box Q, returns index i with probability q_i.

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

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

- **Variance:** controlled by the quantity

$$\mathbb{E}[Z_Q^2] = \sum_{i=1}^{n} \frac{u_i^2}{q_i}$$

Ideal Importance sampling

Setting: weights $u_1, \ldots, u_n$ e.g. $u_i = K(w, x_i)$,
Goal: approximate $\mu = \sum_{i=1}^{n} u_i$

Importance Sampling
Scheme that minimizes the variance satisfies:

$$q_i \propto u_i \Rightarrow q_i = \frac{u_i}{\sum_{j=1}^{n} u_j}$$

Three caveats:

- Each $u_i = k(w, x_i)$ is query dependent!
- probabilities can vary dramatically with the query!

Importance sampling through hashing!

Ideal Importance sampling

Setting: weights $u_1, \ldots, u_n$ e.g. $u_i = K(w, x_i)$,
Goal: approximate $\mu = \sum_{i=1}^{n} u_i$

Importance Sampling
Scheme that minimizes the variance satisfies:

$$q_i \propto u_i \Rightarrow q_i = \frac{u_i}{\sum_{j=1}^{n} u_j}$$

Three caveats:

- Each $u_i = k(w, x_i)$ is query dependent!
- probabilities can vary dramatically with the query!

Importance sampling through hashing!

Ideal Importance sampling

Setting: weights $u_1, \ldots, u_n$ e.g. $u_i = K(w, x_i)$,
Goal: approximate $\mu = \sum_{i=1}^{n} u_i$

Importance Sampling

Scheme that minimizes the variance satisfies:

$$q_i \propto u_i \Rightarrow q_i = \frac{u_i}{\sum_{j=1}^{n} u_j}$$

Three caveats:

- Each $u_i = k(w, x_i)$ is query dependent!
- probabilities can vary dramatically with the query!

Importance sampling through hashing!

Part 2:

Hashing-Based-Estimators

Motivation: Locality Sensitive Hashing (LSH)

Colission probability: family $\mathcal{H}$, distrib. ν on $\mathcal{H}$,sample $h \sim \nu$

$$p(w, x) = \mathbb{P}[h(w) = h(x)]$$

Locality Sensitive Hashing: used to solve ANN with hash func.

Collision probability is decreasing with distance!

Main contribution:

Use LSH to get IS scheme with provable guarantees!

Example: Euclidean LSH

Basic principle:

close points when projected on a random line, remain close!

Euclidean LSH [Datar et al.'04]

- Pick a Gaussian random vector $g \in \mathbb{R}^d$.
- Project $\langle g, x \rangle$
- Add a random shift: $b \sim [0, 1]$
- Pick a nominal scale $r > 0$

$$h_{g,b,r}(x) = \lceil \frac{\langle g, x \rangle}{r} + b \rceil$$

Simple and intuitive hash function. Cost $O(d \cdot n)$

Example: Euclidean LSH

Basic principle:

close points when projected on a random line, remain close!

Euclidean LSH [Datar et al.'04]

- Pick a Gaussian random vector $g \in \mathbb{R}^d$.
- Project $\langle g, x \rangle$
- Add a random shift: $b \sim [0, 1]$
- Pick a nominal scale $r > 0$

$$h_{g,b,r}(x) = \lceil \frac{\langle g, x \rangle}{r} + b \rceil$$

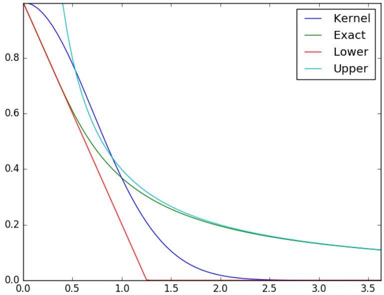
Simple and intuitive hash function. Cost $O(d \cdot n)$

Colission Probability of E2LSH

Let $c = \frac{\|x-y\|}{r}$, using **analytical arguments** (isotropy, random shift)

$$\mathbb{P}[h(x) = h(y)] = f_1(c)$$

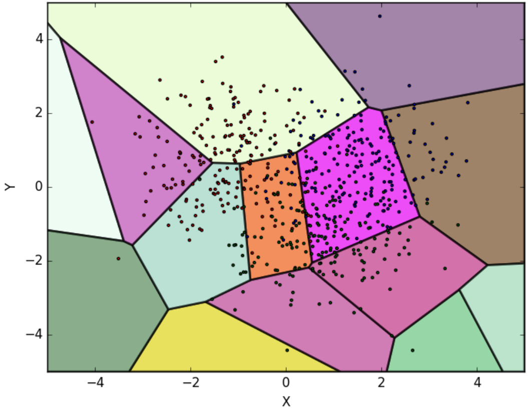
- Exponential decay: $c \ll 1$, $f_1(c) \asymp 1 - \sqrt{\frac{2}{\pi}}c \asymp e^{-\sqrt{\frac{2}{\pi}}c}$
- Polynomial decay: $c \geq 2$, $f_1(c) \asymp \frac{1}{\sqrt{2\pi}}\frac{1}{c}$



LSH as Randomized Space Partition

Close points more likely to be found in the same bucket.

Hash bucket of query $\Rightarrow$ biased sample!



Challenges

KDE problem: implement this idea we need to answer:

- Given a kernel, which hashing scheme should we pick?
- How should we use the information in the hash buckets?
- How should we tune the parameters?
- How many samples required?
- Additional structure in data?

Answer: Framework of Hashing-Based-Estimators.

Challenges

KDE problem: implement this idea we need to answer:

- Given a kernel, which hashing scheme should we pick?
- How should we use the information in the hash buckets?
- How should we tune the parameters?
- How many samples required?
- Additional structure in data?

Answer: Framework of Hashing-Based-Estimators.

Hashing-Based-Estimators

$P \subset \mathbb{R}^d$, $\mathcal{H}$, measure ν, collision probability $p(x, y)$, kernel K

Preprocessing

- Sample a number m of i.i.d. hash functions $h_1, \ldots, h_m \sim \nu$.
- create m hash tables $H_1, \ldots, H_m$ where $H_i = h_i(P)$

Unbiased estimator: query $w \in \mathbb{R}^d$ we may form

- let $H_j(w)$ be the hash bucket where w maps to.
- let $x_{(j)}$ be a uniform random point from $H_j(w)$,
- Return: $Z_j = \frac{K(w, x_{(j)})}{\frac{p(w, x_{(j)})}{|H_j(w)|}}$

The estimator is unbiased for all w and all $j = 1, \ldots, m$.

Hashing-Based-Estimators

$P \subset \mathbb{R}^d$, $\mathcal{H}$, measure ν, collision probability $p(x,y)$, kernel K

Preprocessing

- Sample a number m of i.i.d. hash functions $h_1, \dots, h_m \sim \nu$.
- create m hash tables $H_1, \dots, H_m$ where $H_i = h_i(P)$

Unbiased estimator: query $w \in \mathbb{R}^d$ we may form

- let $H_j(w)$ be the hash bucket where w maps to.
- let $x_{(j)}$ be a uniform random point from $H_j(w)$,
- Return: $Z_j = \frac{K(w, x_{(j)})}{\frac{p(w, x_{(j)})}{|H_j(w)|}}$

The estimator is unbiased for all w and all $j = 1, \dots, m$.

Hashing-Based-Estimators

$P \subset \mathbb{R}^d$, $\mathcal{H}$, measure ν, collision probability $p(x, y)$, kernel K

Preprocessing

- Sample a number m of i.i.d. hash functions $h_1, \ldots, h_m \sim \nu$.
- create m hash tables $H_1, \ldots, H_m$ where $H_i = h_i(P)$

Unbiased estimator: query $w \in \mathbb{R}^d$ we may form

- let $H_j(w)$ be the hash bucket where w maps to.
- let $x_{(j)}$ be a uniform random point from $H_j(w)$,
- Return: $Z_j = \frac{K(w, x_{(j)})}{\frac{p(w, x_{(j)})}{|H_j(w)|}}$

The estimator is unbiased for all w and all $j = 1, \ldots, m$.

Remarks

- Reservoir Sampling for each hash bucket
- **Sample compression:** store a single point and size for all non-empty hash buckets!
- Scalability: the above operations are completely decoupled for different j

But what can we say at this level of generality?

Remarks

- Reservoir Sampling for each hash bucket
- **Sample compression:** store a single point and size for all non-empty hash buckets!
- Scalability: the above operations are completely decoupled for different j

But what can we say at this level of generality?

Variance of HBE

query w, $u_i = k(w, x_i)$, set $p_i = p(w, x_i)$, $\mu = \mathrm{KDE}_P(w)$

$$\mathbb{E}[Z^2] = \sum_{i=1}^{n} \frac{u_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| \,|\, i \in H(\mathbf{x})] \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{u_i^2}{p_i} \min\{p_i, p_j\}$$

Theorem 1 [Charikar, S., FOCS'17]

Worst case datasets for HBE have support on two points.

- Worsts case variance $\Rightarrow$ solution to an optimization problem!
- Quantifies **Compatibility** between $\{u_i\}$, $\{p_i\}$ at level μ
- Under no assumptions on k, p problem might be intractable
- In certain cases, computing variance reduces to case analysis!

Variance of HBE

query w, $u_i = k(w, x_i)$, set $p_i = p(w, x_i)$, $\mu = \mathrm{KDE}_P(w)$

$$\mathbb{E}[Z^2] = \sum_{i=1}^{n} \frac{u_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| | i \in H(\mathbf{x})] \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{u_i^2}{p_i} \min\{p_i, p_j\}$$

Theorem 1 [Charikar, S., FOCS'17]

Worst case datasets for HBE have support on two points.

- Worsts case variance $\Rightarrow$ solution to an optimization problem!
- Quantifies **Compatibility** between $\{u_i\}, \{p_i\}$ at level μ
- Under no assumptions on k, p problem might be intractable
- In certain cases, computing variance reduces to case analysis!

Variance of HBE

query w, $u_i = k(w, x_i)$, set $p_i = p(w, x_i)$, $\mu = \mathrm{KDE}_P(w)$

$$\mathbb{E}[Z^2] = \sum_{i=1}^{n} \frac{u_i^2}{p_i} \mathbb{E}[|H(\mathbf{x})| | i \in H(\mathbf{x})] \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{u_i^2}{p_i} \min\{p_i, p_j\}$$

Theorem 1 [Charikar, S., FOCS'17]

Worst case datasets for HBE have support on two points.

- Worsts case variance $\Rightarrow$ solution to an optimization problem!
- Quantifies **Compatibility** between $\{u_i\}, \{p_i\}$ at level μ
- Under no assumptions on k, p problem might be intractable
- In certain cases, computing variance reduces to case analysis!

Gaussian Kernel using Euclidean LSH

Theorem 2 [Charikar, S., FOCS'17]

There exists a **HBE** for the KDE under Gaussian Kernel using Euclidean LSH that has variance bounded by $O(\frac{1}{\mu^{3/4}} \cdot \mu^2)$

- uniform random sampling has variance bounded by $O(\frac{1}{\mu} \cdot \mu^2)$.
- $\frac{1}{\mu^{1/4}}$ improvement (μ is small, e.g. $\mu = n^{-\alpha}$)
- Intuition: exponential decay of the E2LSH to simulate Gaussian kernel by **concatenating** many hash functions!

Better estimator? Simple design principle?

Scale-free Estimators

We introduce (β, M) **scale-free** property of a HBE

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

Theorem 3 [Charikar, S., FOCS'17]

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

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

Scale-free Estimators

We introduce (β, M) **scale-free** property of a HBE

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

Theorem 3 [Charikar, S., FOCS'17]

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

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

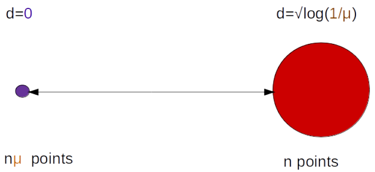
Scale-free Estimators

We introduce (β, M) **scale-free** property of a HBE

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

Theorem 3 [Charikar, S., FOCS'17]

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



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

Scale-free Estimators through LSH

Theorem 4 [Charikar, S., FOCS'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 4 [Charikar, S., FOCS'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!

Method of Scale-free HBE

Given a kernel $k(x, y)$ and a dataset P.

1. Construct hash function such that

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

2. Use Theorem to set $V(\mu) = 4M^3 \frac{1}{\sqrt{\mu}}$ in **Median-of-means**.
3. Adaptive procedure to estimate μ [Charikar, S., FOCS'17]

Efficient data structures that can answer queries!

Method of Scale-free HBE

Given a kernel $k(x, y)$ and a dataset P.

1. Construct hash function such that

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

2. Use Theorem to set $V(\mu) = 4M^3 \frac{1}{\sqrt{\mu}}$ in **Median-of-means**.
3. Adaptive procedure to estimate μ [Charikar, S., FOCS'17]

Efficient data structures that can answer queries!

Method of Scale-free HBE

Given a kernel $k(x, y)$ and a dataset P.

1. Construct hash function such that

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

2. Use Theorem to set $V(\mu) = 4M^3 \frac{1}{\sqrt{\mu}}$ in **Median-of-means**.
3. Adaptive procedure to estimate μ [Charikar, S., FOCS'17]

Efficient data structures that can answer queries!

Method of Scale-free HBE

Given a kernel $k(x, y)$ and a dataset P.

1. Construct hash function such that

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

2. Use Theorem to set $V(\mu) = 4M^3 \frac{1}{\sqrt{\mu}}$ in **Median-of-means**.
3. Adaptive procedure to estimate μ [Charikar, S., FOCS'17]

Efficient data structures that can answer queries!

Method of Scale-free HBE

Given a kernel $k(x, y)$ and a dataset P.

1. Construct hash function such that

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

2. Use Theorem to set $V(\mu) = 4M^3 \frac{1}{\sqrt{\mu}}$ in **Median-of-means**.
3. Adaptive procedure to estimate μ [Charikar, S., FOCS'17]

Efficient data structures that can answer queries!

Extensions

- The scale-free property quite **strong** and hard to achieve!

  Is there any other way?
- Method very sensitive to specific kernel/bandwidth:

  Single data-structure for different kernels in a family?
- What structural information can we exploit?

Extensions

- The scale-free property quite **strong** and hard to achieve!

  Is there any other way?

- Method very sensitive to specific kernel/bandwidth:

  Single data-structure for different kernels in a family?

- What structural information can we exploit?

Extensions

- The scale-free property quite **strong** and hard to achieve!

  Is there any other way?

- Method very sensitive to specific kernel/bandwidth:

  Single data-structure for different kernels in a family?

- What structural information can we exploit?

Multi-resolution HBE

- design a HBE that is scale-free for a kernel only locally
- construct many such HBE to **approximate** scale-free property.
- Sample points from each such estimator and weigh them appropriately.

$$K(w, x) = e^{\langle w, x \rangle - 1}$$

Multi-resolution HBE

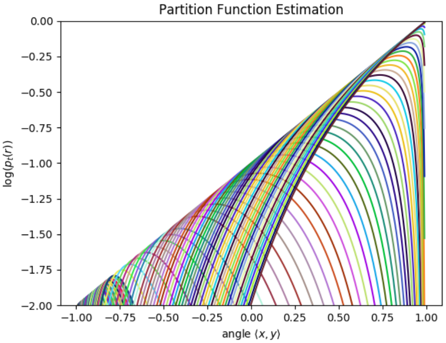
- design a HBE that is scale-free for a kernel only locally
- construct many such HBE to **approximate** scale-free property.
- Sample points from each such estimator and weigh them appropriately.

$$K(w, x) = e^{\langle w, x \rangle - 1}$$

Multi-resolution HBE

- design a HBE that is scale-free for a kernel only locally
- construct many such HBE to **approximate** scale-free property.
- Sample points from each such estimator and weigh them appropriately.

$$K(w,x) = e^{\langle w,x \rangle - 1}$$



Polynomial Kernels

Constants $L \geq 1$, $t \geq 0$, a kernel K is (L, t)-nice if $\forall w, y, x \in \mathbb{R}^d$:

$$\max\left\{\frac{K(w,y)}{K(w,x)}, \frac{K(w,x)}{K(w,y)}\right\} \leq L \cdot \max\left\{\frac{\|w-y\|}{\|w-x\|}, \right\}^t$$

Theorem [Backurs, Charikar, Indyk, S.'17]

There exist a data structure that that can answer queries for all (L, t)-nice kernels in time $2^{O(t)} L \frac{\log n}{\epsilon^2}$.

- Uses Projected quadtrees on dimension $O(t)$!
- **Trade-off** between computational amenability and rate of decay.

Polynomial Kernels

Constants $L \geq 1$, $t \geq 0$, a kernel K is (L, t)-nice if $\forall w, y, x \in \mathbb{R}^d$:

$$\max\left\{\frac{K(w,y)}{K(w,x)}, \frac{K(w,x)}{K(w,y)}\right\} \leq L \cdot \max\left\{\frac{\|w-y\|}{\|w-x\|}, \right\}^t$$

Theorem [Backurs, Charikar, Indyk, S.'17]

There exist a data structure that that can answer queries for all (L, t)-nice kernels in time $2^{O(t)} L \frac{\log n}{\epsilon^2}$.

- Uses Projected quadtrees on dimension $O(t)$!
- **Trade-off** between computational amenability and rate of decay.

Polynomial Kernels

Constants $L \geq 1$, $t \geq 0$, a kernel K is (L, t)-nice if $\forall w, y, x \in \mathbb{R}^d$:

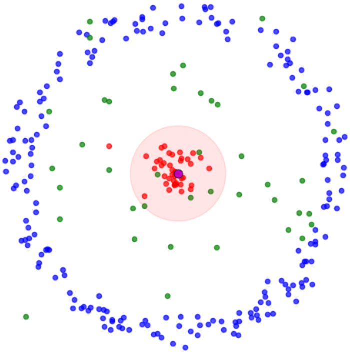
$$\max\left\{\frac{K(w,y)}{K(w,x)}, \frac{K(w,x)}{K(w,y)}\right\} \leq L \cdot \max\left\{\frac{\|w-y\|}{\|w-x\|}, \right\}^t$$

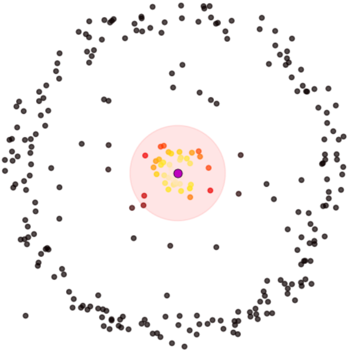
Theorem [Backurs, Charikar, Indyk, S.'17]

There exist a data structure that that can answer queries for all (L, t)-nice kernels in time $2^{O(t)} L \frac{\log n}{\epsilon^2}$.

- Uses Projected quadtrees on dimension $O(t)$!
- **Trade-off** between computational amenability and rate of decay.

Localized Queries





KDE through ANN

Hashing-based-estimators through LSH.

Given arbitrary ANN algorithm, can we use it for KDE?

Theorem 5 [Backurs, Charikar, Indyk, S.'17]

Every c-ANN algorithm can be used to solve KDE problems for (L, t)-radial kernels using $O(\frac{1}{\epsilon^5} c^{5t} \log^2(n))$ calls to ANN.

KDE through ANN

Hashing-based-estimators through LSH.

Given arbitrary ANN algorithm, can we use it for KDE?

Theorem 5 [Backurs, Charikar, Indyk, S.'17]

Every c-ANN algorithm can be used to solve KDE problems for (L, t)-radial kernels using $O(\frac{1}{\epsilon^5} c^{5t} \log^2(n))$ calls to ANN.

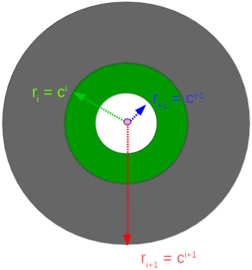
KDE through ANN

Hashing-based-estimators through LSH.

Given arbitrary ANN algorithm, can we use it for KDE?

Theorem 5 [Backurs, Charikar, Indyk, S.'17]

Every c-ANN algorithm can be used to solve KDE problems for (L, t)-radial kernels using $O(\frac{1}{\epsilon^5} c^{5t} \log^2(n))$ calls to ANN.



Spherical Inegration

Summary

- **Machine learning primitives:** computing query dependent sums.
- Importance sampling through Hashing-Based-Estimators.
- Method of Scale-free estimators.
- **Polynomial kernels:** simple and practical data-structure, trade-offs between cost and resolution!

Acknowledgments

Arturs Backurs (MIT) Moses Charikar Piotr Indyk (MIT)

Peter Bailis

Future work

- Implementations, Applications, Benchmarks!
- Analogs of Fast Multipole Methods using Doubling Dimension.
- Training of Neural Networks and Random Fourier Features.

Thank You!

psimin@stanford.edu