QUALITY AND EFFICIENCY IN KERNEL DENSITY ESTIMATES FOR LARGE DATA

Yan Zheng  Jeffrey Jестes  Jeff M. Philips  Feifei Li

University of Utah
Kernel Density Estimates

• Let $P$ be an input point set of size $n$ in domain $\mu$ from an unknown distribution $f$. $P \in \mathbb{R}^1$ or $P \in \mathbb{R}^2$ in our experiments.

• Kernel Density Estimates (KDE) function approximates the density of $f$ at any possible input point $x \in \mu$

$$\text{KDE}_P(x) = \frac{1}{|P|} \sum_{p \in P} K(p, x)$$

• KDE represents a continuous distribution from a finite set of points.
Kernel Density Estimates
Drawback of Histograms

- Query value change significantly across the boundary of the bin
- The choice of origin can have quite an effect
Example of Kernel Density Estimates
Examples of kernels (2D)

- **Gaussian:**
  \[
  K(p, x) = \frac{1}{2\pi \sigma^2} \exp \left( \frac{-\|p-x\|^2}{2\sigma^2} \right)
  \]

- **Triangle:**
  \[
  K(p, x) = \frac{3}{\pi \sigma^2} \max \left\{ 0, 1 - \frac{\|p-x\|}{\sigma} \right\}
  \]

- **Epanechnikov:**
  \[
  K(p, x) = \frac{2}{\pi \sigma^2} \max \left\{ 0, 1 - \frac{\|p-x\|^2}{\sigma^2} \right\}
  \]

- **Ball**
  \[
  K(p, x) = \begin{cases} 
  1/\pi \sigma^2 & \text{if } \|p-x\| < \sigma \\
  0 & \text{otherwise}
  \end{cases}
  \]
Comparing of Different Kernels
Approximate Kernel Density Estimates

- **Kernel Density Estimates**
  \[ \text{KDE}_P(x) = \frac{1}{|P|} \sum_{p \in P} K(p, x) \]

- **KDE is very expensive for large dataset.**
  - Each query takes \( O(n) \)

- **\( \varepsilon \)-approximation**
  - Given input \( P, \sigma \) and error \( \varepsilon \), the goal is to produce a small set \( Q \) to ensure:
  \[
  \max_{x \in \mathbb{R}^d} \left| \text{KDE}_P(x) - \text{KDE}_Q(x) \right| = \| \text{KDE}_P - \text{KDE}_Q \|_\infty \leq \varepsilon
  \]
MergeReduce (MR) framework

- Initialization phase:
  - Arbitrarily decompose $P$ into disjoint sets of size $k$ $P_1$,

- Combination phase
  - Merge Step $k+k \rightarrow 2k$
  - Reduce Step $2k \rightarrow k$ (Matching Pairs & Randomly Select one)
  - Proceeds in $\log(n/k)$ rounds

Diagram:

- $P_1$, $P_2$, $P_3$, $P_4$, ..., $P_i$, $P_j$
- $P_{12}$, $P_{34}$, ..., $P_{ij}$
- $P_{1234}$
- $Q$
Matching

- Min-cost Matching [Phillips SODA13]
  - Edmonds’ Blossom algorithm $O(n^3)$
  - 2–approximation Greedy algorithm $O(n^2 \log n)$
- Introduce More Efficient Matching
  - Edge Map $E_M = \{e(p, q) \mid (p, q) \in M\}$
  - Given a disk $B$,
    
    \[
    E_M \cap B = \{e(p, q) \cap B \mid e(p, q) \in E_M\}
    \]

    \[
    C_{M,B} = \sum_{e(p,q) \in E_M \cap B} \|p - q\|^2 \quad C_M = \max_B C_{M,B}.
    \]

    Min-cost Matching $C_M = O(1)$
    Our Matching $C_M = O(\log(1/\varepsilon))$
Grid Matching

- Min-cost matching $C_M = O(1)$
- Grid Matching $C_M = O(\log(1/\varepsilon))$

Running time $O(n \log(1/\varepsilon))$

- Starting with $i = 0$, construct $G_i$, length $l_{\varepsilon,i} = \sqrt{2\sigma\varepsilon}2^{i-2}$
- Inside of each cell, match points arbitrarily.
- Only the unmatched points survive to the next round.
- Each cell in $G_{i+1}$ is the union of 4 cells from $G_i$
- After $\log(1/\sigma\varepsilon) + 1$ rounds, match points left arbitrarily.
Grid Matching

- Point set $P \subset \mathbb{R}^2$ with $n$ points, we can construct $Q$ giving an $\epsilon$-approximate KDE in

$$O(n \log \frac{1}{\epsilon})$$ running time and $|Q| = O\left(\frac{1}{\epsilon} \log n \log^{1.5} \frac{1}{\epsilon}\right)$

Using Grid-MR

- With Random sampling (RS).
  - Random sample a set $P'$ from $P$ with size $O\left((1/\epsilon^2) \log(1/\delta)\right)$

$$O(n + \frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$$ running time and $|Q| = O\left(\frac{1}{\epsilon} \log^{2.5} \frac{1}{\epsilon}\right)$

  Using Grid-MR+RS
Deterministic Z-Order Selection

- The levels of the Z-order curve are reminiscent of the grids
- Zrandom
  - Compute the Z-order of all points, and of every two points discard one at random.
  - Repeat this discarding of half the points until the remaining set is sufficiently small.
- Zorder
  - Select one point from each set of |P|/k points in the sorted order using $\varepsilon$-approximate quantiles algorithm
Other Baseline Methods

• Improved fast gauss transform (IFGT)
  • K-center clustering
  • Hermite expansion
  • For n points, m query points, reduce time from $O(mn)$ to “$O(m + n)$”
  • No error vs. size or error vs. time guarantees.
  • Only works for Gaussian Kernel

• Kernel herding (KH)
  • Explore the reproducing kernel Hilbert space.
  • It adds at each step the single point $p \in P \setminus Q$ to Q which most decreases $\| \text{KDE}_Q - \text{KDE}_P \|_2$
  • Running Time $O(|Q|n)$
  • Bounds only $\ell_2$ error.
Centralized Experiments

- **Data sets: OpenStreetMap**
  - 160 million records in 6.6GB

- **Default setting**
  - 10 million records
  - 10 random trials
  - $\delta = 0.001$
  - $\sigma = 200$ on a domain
    - $50,000 \times 50,000$

- **Test points**
  - 4000 randomly from $P$
  - 1000 from the domain of $P$
Centralized Experiments

- Compare with Baseline Methods
  - Grid-MR \( O(n \log(1/\varepsilon)) \)
  - Blossom-MR \( O(n^3) \)
  - Greedy-MR \( O(n^2 \log n) \)
  - Kernel Herding

- Using Random Sampling as preprocessing step.
  - Grid-MR
  - Zorder
  - Grid-MR+RS
  - Zorder+RS
Centralized Experiments

Our construction time and size beats IFGT
Just random sampling is slightly faster, but worse size bounds

(a) Construction time vs. err
(b) Size
Distributed Experiments

Default setting: 50 million records

(a) Communication cost vs. err
(b) Size vs. err

Best small approximate, easy to construct representation of data for kernel density estimates.
Thank you

http://www.cs.utah.edu/~yanzheng/kde/
Observed error vs. guaranteed error

(b) err vs. $\varepsilon$

Legend:
- $\text{GRID}$
- $\text{RS+GRID}$
- $\text{zKERN}$
- $\text{RS+zKERN}$
Compare with Random Sampling for High Accuracy

(b) Size vs. $\varepsilon$

G-MR+RS • Z+RS ⊕ RS

(b) Size vs. $\varepsilon$

Time (seconds)

0.05 0.1 0.5 1

$\varepsilon$ (x10$^{-3}$)

0 10 20 30

Size (bytes)

0.05 0.1 0.5 1

$\varepsilon$ (x10$^{-3}$)

10$^6$ 10$^7$ 10$^8$

0 10 20 30

G-MR+RS • Z+RS ⊕ RS

(d) Size vs. err

Time (seconds)

10$^{-6}$ 10$^{-5}$ 10$^{-4}$

err

10$^6$ 10$^7$ 10$^8$

0 10 20 30