ALGORITHMS AND CORESETS FOR LARGE-SCALE KERNEL SMOOTHING

by

Yan Zheng

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

August 2017

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Yan Zheng**

has been approved by the following supervisory committee members:

Jeffrey Phillips ,	Chair(s)	**April 20 2017** Date Approved
Feifei Li ,	Member	**April 20 2017** Date Approved
Suresh Venkatasubramanian ,	Member	**April 20 2017** Date Approved
Vivek Srikumar ,	Member	**April 20 2017** Date Approved
Clayton Scott ,	Member	**May 1 2017** Date Approved

by **Ross Whitaker** , Chair/Dean of

the Department/College/School of **Computing**

and by **David B. Kieda** , Dean of The Graduate School.

ABSTRACT

Kernel smoothing provides a simple way of finding structures in data sets without the imposition of a parametric model, for example, nonparametric regression and density estimates. However, in many data-intensive applications, the data set could be large. Thus, evaluating a kernel density estimate or kernel regression over the data set directly can be prohibitively expensive in big data. This dissertation is working on how to efficiently find a smaller data set that can approximate the original data set with a theoretical guarantee in the kernel smoothing setting and how to extend it to more general smooth range spaces.

For kernel density estimates, we propose randomized and deterministic algorithms with quality guarantees that are orders of magnitude more efficient than previous algorithms, which do not require knowledge of the kernel or its bandwidth parameter and are easily parallelizable. Our algorithms are applicable to any large-scale data processing framework.

We then further investigate how to measure the error between two kernel density estimates, which is usually measured either in L_1 or L_2 error. In this dissertation, we investigate the challenges in using a stronger error, L_∞ (or worst case) error. We present efficient solutions for how to estimate the L_∞ error and how to choose the bandwidth parameter for a kernel density estimate built on a subsample of a large data set.

We next extend smoothed versions of geometric range spaces from kernel range spaces to more general types of ranges, so that an element of the ground set can be contained in a range with a non-binary value in $[0, 1]$. We investigate the approximation of these range spaces through ε-nets and ε-samples.

Finally, we study coresets algorithms for kernel regression. The size of the coresets are independent of the size of the data set, rather they only depend on the error guarantee, and in some cases the size of domain and amount of smoothing. We evaluate our methods on very large time series and spatial data, demonstrate that they can be constructed extremely efficiently, and allow for great computational gains.

CONTENTS

ABSTRACT . . . . **iii**

LIST OF FIGURES . . . . **vii**

ACKNOWLEDGEMENTS . . . . **ix**

CHAPTERS

1. INTRODUCTION **1**

1.1 Kernel Density Estimates and Kernel Regression 1
1.2 Coresets 5
1.3 Kernel Range Spaces and KDE Coreset 6
1.4 KDE in Geometric Inference 7
1.5 Outline 9

2. EFFICIENT CORESETS FOR KERNEL DENSITY ESTIMATES **11**

2.1 Background and Related Work 11
2.2 Warm-up: One Dimension 13
2.2.1 Efficient Evaluation 15
2.3 Two Dimensions 16
2.3.1 Baseline Methods 16
2.3.2 Randomized MergeReduce Algorithm 17
2.3.2.1 The MergeReduce framework 17
2.3.2.2 More Efficient Reduce Step 19
2.3.2.3 Streaming and Distributed MergeReduce 22
2.3.2.4 Other Extension 23
2.3.3 Deterministic Z-Order Selection 23
2.3.4 Efficient Query Evaluation 25
2.3.5 Parallel and Distributed Implementation 25
2.3.6 Higher Dimensions 26
2.4 Experiments 26
2.4.1 Two Dimensions: Centralized 28
2.4.2 Two Dimensions: Parallel and Distributed 31
2.4.3 Two Dimensions: Is RS Good Enough? 33
2.4.4 One Dimension: Parallel and Distributed 34

3. $L\infty$ ERROR AND BANDWIDTH SELECTION FOR KDES **36**

3.1 Background and Motivation 38
3.1.1 Unit Kernels or Normalized Kernels? 38
3.1.2 Why σ Is Given? 38
3.1.3 Why L_∞ Error? 39

3.1.4 Computing Coreset Q of P 40
3.1.5 Related Work on Bandwidth Selection 41
3.2 Computing $err(P, Q)$ Error 42
3.2.1 Approximation Strategy 42
3.2.2 Generation of Evaluation Points 46
3.3 Bandwidth Selection 48
3.3.1 Refining the Bandwidth for Coresets 48
3.3.2 Lipschitz Properties of h 49
3.3.3 Random Golden Section Search 50
3.4 Experiments 51
3.4.1 Data Sets 51
3.4.2 Evaluating Point Generation for err_X 53
3.4.3 Choosing New Bandwidth Evaluation 55
3.4.4 Compare with the Traditional Bandwidth Selection Method 58
3.5 Conclusion 59

4. GENERALIZED KERNEL RANGE SPACE AND (ε, τ)-NET 60

4.1 Definitions and Background 62
4.1.1 Kernels 62
4.2 New Definitions 63
4.2.1 Smoothed Range Spaces 64
4.2.2 ε-Sample in a Smoothed Range Space 65
4.2.3 (ε, τ)-Net in a Smoothed Range Space 66
4.3 Linking and Properties of (ε, τ)-Nets 67
4.3.1 Relations between Smoothed Range Spaces and Linked Binary Range Spaces 67
4.4 Min-Cost Matchings within Cubes 69
4.5 Constructing ε-Samples for Smoothed Range Spaces 73
4.5.1 Discrepancy for Smoothed Halfspaces 74
4.5.2 From a Single Smoothed Halfspace to a Smoothed Range Space 75
4.5.3 ε-Samples for Smoothed Halfspaces 75
4.5.4 Generalization to d Dimensions 76

5. CORESETS FOR KERNEL REGRESSION 77

5.1 Basic Definitions 77
5.1.1 Coresets for Kernel Regression 77
5.1.2 Our Approach 78
5.1.3 Our Results 79
5.1.4 Related Work 80
5.2 Subset Selection Methods 81
5.2.1 Progressive Grid-Based Approaches 82
5.3 Analysis 83
5.3.1 Accuracy of Random Sampling 86
5.3.2 Accuracy of Grid-Based Approaches 89
5.4 Experiments 90
5.4.1 Data Sets 90
5.4.2 Effectiveness of Coresets 91

5.4.3 Efficiency of Coresets 93
5.4.4 Consistency with Bandwidth 94
5.4.5 Progressive Grid-Based Approaches 95
5.4.6 High Dimension 96
5.5 Conclusion 96

6. CONCLUSION 97

APPENDIX: PROOFS FOR CORESET ANALYSIS 99

REFERENCES 103

LIST OF FIGURES

1.1 Kernel density estimate (KDE). 2

1.2 Kernel regression of synthetic data with bandwidth 50 (left) and 200 (right). . . 5

1.3 Range Spaces and an example of ε-sample. 6

1.4 Example with 10,000 points in $[0,1]^2$ generated on a circle or line with $N(0,0.005)$ noise; 25% of points are uniform background noise. The generating function is reconstructed with KDE with $\sigma = 0.05$ (upper left), and its persistence diagram based on the superlevel set filtration is shown (upper middle). A coreset [149] of the same data set with only 1,384 points (lower left) and persistence diagram (lower middle) are shown, again using KDE. This associated confidence interval contains the dimension 1 homology features (red triangles) suggesting they are noise; this is because it models data as iid – but the coreset data is not iid, it subsamples more intelligently. We also show persistence diagrams of the original data based on the sublevel set filtration of the standard distance function (upper right, with no useful features due to noise) and the kernel distance (lower right). 9

2.1 Intersection between a matching and a disk, solid red lines are included in $E_M \cap B$. Left shows Grid matching and right shows Z-order with every other edge in a matching. 20

2.2 Effect of guaranteed error ε on the centralized G-MR, G-MR+RS, Z, Z+RS. . . 29

2.3 Centralized G-MR, B-MR, Greedy-MR, KH. 30

2.4 Effect of measured ℓ_∞ error *err* and N on centralized IFGT, RS, G-MR+RS, Z+RS. 31

2.5 Effect of measured ℓ_∞ error *err* on distributed and parallel G-MR+RS, Z+RS, RS. 32

2.6 Effect of n on G-MR+RS, Z+RS, RS. 33

2.7 Effect of ε and ℓ_∞ error *err* on G-MR+RS, Z+RS, RS for high precision case. . . . 34

2.8 Effect of varying measured *err* on RS, SS on one-dimensional data. 35

3.1 KDEs with different bandwidths showing daily, weekly, and yearly trends. Left shows a full year of data, and right shows one week of data. 39

3.2 Illustration of the Minkowski sum of a ball of radius σ and convex hull of $P \cup Q$. 43

3.3 Example with need for larger ω for coreset Q. 48

3.4 Two curves, one of which is Lipschitz. 49

3.5 Results on one-dimensional synthetic data to choose best evaluating point generation techniques. 52

3.6 Visualization of KDE_P and KDE_Q for two-dimensional synthetic data using different bandwidth. 52

3.7 Visualization of KDE_P and KDE_Q for two-dimensional real data using different bandwidth.. 54

3.8 Results on one-dimensional real data to choose the best evaluating point generation techniques. 54

3.9 Choosing the best evaluation set X for on two-dimensional synthetic (left) and real (right) data. 56

3.10 $\omega^* = \arg\min_\omega err_X(P, \sigma, Q, \omega)$ in $\mathbb{R}^1$. 57

3.11 $\omega^* = \arg\min_\omega \exp_X(P, \sigma, Q, \omega)$ in $\mathbb{R}^2$. 57

3.12 $\omega^* = \arg\min_\omega \exp_X(P, \sigma, Q, \omega)$ in $\mathbb{R}^2$. 58

4.1 Illustration of the smoothed halfspace, and smoothed polynomial surface, with function value of three points $\{p_1, p_2, p_3\}$ defined using a triangle kernel. 65

4.2 (T3) edges. 70

5.1 Example improvement of **Aggregate-Neighbor** (right) over **G-Aggregate** (left). Input $P = \{(1\,100), (2\,40), (3\,0), (15\,50), (16\,50), (17\,50)\}$. With **G-Aggregate** with $\gamma = 2$, the coreset $Q = \{(1.5\,70), (3\,0), (15.5\,50), (17\,50)\}$. The largest errors occur at $x < 0$ and around $x = 9$. If we add the extra points at the empty grid cells with nonempty neighbor grids, i.e., **Aggregate-Neighbor**, then L_∞ is significantly reduced. We add three points $(-1\ 98.3124), (5\ 3.2559)$, and $(13\,50)$. 83

5.2 The maximum L_∞ error found based on the number of evaluation points on real (left) and synthetic data (middle) when $P_x \subset \mathbb{R}^1$, and real data (right) when $P_x \subset \mathbb{R}^2$. 92

5.3 L_∞ error for coresets when also testing sparse regions on real data(left) and synthetic data(middle) when $P_x \subset \mathbb{R}$, and real data(right) when $P_x \subset \mathbb{R}^2$. 92

5.4 Comparison of construction time and query time for real data set with $P_x \subset \mathbb{R}$. 94

5.5 Comparison of construction time and query time for real data set with $P_x \subset \mathbb{R}^2$. 94

5.6 The relation between L_∞ error and bandwidth for real data (left), synthetic data (middle) for $P_x \subset \mathbb{R}$ and real data (right) for $P_x \subset \mathbb{R}^2$. 95

5.7 The relation between window size and L_∞ error for progressive G-Aggregate method. 95

5.8 Left: L_∞ error for coresets of high dimensional data sets. Right: Running time to generate the coresets. 96

ACKNOWLEDGEMENTS

I could never have enjoyed my last five years so much without the guidance, support, and efforts of so many people with whom I have interacted.

First, I am extremely grateful to have been under the guidance of my great advisor, Jeff Phillips, who has demonstrated to me how to be an excellent researcher, advisor, teacher, and friend. I greatly appreciate his patience while I learned to follow his clear and rigorous approach to geometry and research. I have learned from him not just research and thinking skills, but also principles that I want to follow in my future life.

I also want to thank the rest of my committee, Feifei Li, Suresh Venkatasubramanian, Vivek Srikumar, and Clayton Scott, for providing meaningful guidance on the process towards my dissertation. Especially I want to thank Feifei for showing me how to design rigorous experiments and how to apply kernel smoothing to the database area. I thank Suresh for welcoming me to his group meeting during my early stages of research. He can always simplify complicated problems and his contagious enthusiasm in research encourages me. I thank Vivek and Clay for helping me to connect my research to machine learning and giving me different aspects to understand my research.

I have been so lucky to work in the wonderful Data Group at the University of Utah, surrounded by many great colleagues. With a ton of fresh and encouraging discussions on research and other fun, I have enjoyed working here every single day. To list a few: Mina Ghashami, Jeffrey Jestes, Robert Christensen, Mingwang Tang, Min Du, Michael Matheny, Pingfan Tang, Wangchao Le, John Moeller, Chi Zhang, Chenxu Ding, Samira Daruki, Dustin Webb, Waiming Tai, Guineng Zheng, and Sunipa Dev. Thank you all for providing an interesting and collaborative research environment. I also want to thank people in the machine learning and NLP group, Haibo Ding, Xingyuan Pan, Jie Cao, and Ruihong Huang, for your useful discussions for inspirations.

I would also like to thank my dear friends I met in the School of Computing, Xia Dong, Shuying Liang, Xing Lin, Liang Zhou, Shuqin Zeng, Peihong Zhu, Wei Liu, Rui Dai, Limou

Wang, and Yi Ou, for either helping me in my research and study or frequently teaching me the real wisdom of life, and with whom I shared so many vivid memories in Utah.

Many thanks to the staff at the School of Computing in the University of Utah. Special thanks go to Karen Feinauer, Ann Carlstrom, and Robert Barber for your immense support, so that I could entirely concentrate on my research.

I also want to thank people in the Atmospheric Sciences department, my second home in the University of Utah. I thank John Horel for supporting me during my master study and gap between master and Ph.D. study, and offering me such interesting projects to work on, which helped me develop my research interests in data mining and machine learning. I thank Chris Galli for guiding me towards the frontier of technology.

I also need to thank those at Nankai University for helping shape my interests in computer science and research, especially Qingren Wang, Guangshun Shi, Jiyi Li, and Yuanfang Liu. Guangshun not only introduced me to research but also cultivated my interest and skills; he taught me the importance of organization.

I also acknowledge the funding that made this dissertation possible. I thank the National Science Foundation and School of Computing for directly funding my research. I am also thankful for the funding I received through grants to my advisor Jeff Phillips from NSF CCF-1350888, IIS-1251019, and ACI-1443046.

Finally, I want to thank my parents, Fuxiang Zheng and Chunmei Cai, and parents-in-law, Jianquan Gao and Heping Shang, for their constant support, confidence, and pride. Their encouragement throughout this process led to resolved and constant progress. Special thanks to my husband, Yang Gao, for his love, care, and support, not only in life, but also in study and research. Last but not least, my two adorable girls, Sunny Gao and Annie Gao, your love teaches me how to be a better mom, your smiles and sweetness are good seasonings to my tense Ph.D. study.

CHAPTER 1

INTRODUCTION

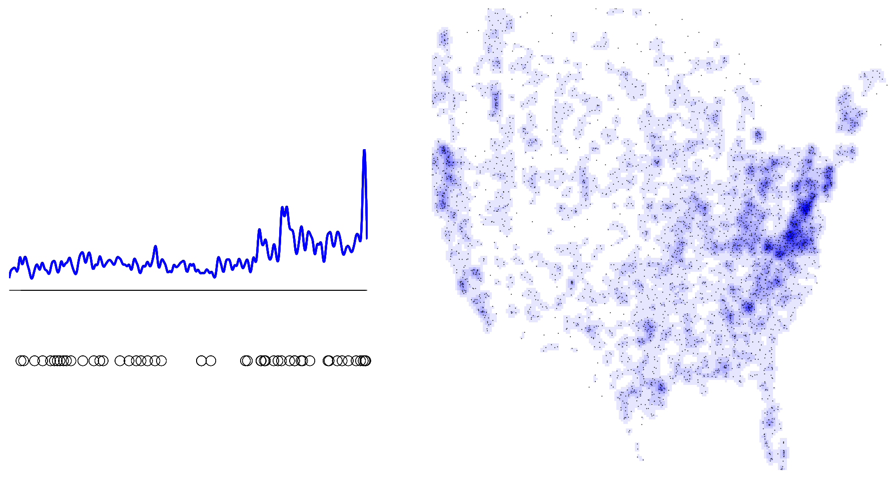
1.1 Kernel Density Estimates and Kernel Regression

A kernel density estimate (KDE) is a statistically-sound method to estimate a continuous distribution from a finite set of points. This is an increasingly common task in many areas, such as outlier detection [20,21], human motion tracking [17], financial data modeling [13], geometric inference [106] and anomaly detection [120]. In many scientific computing and data-intensive applications, the input data set P is a finite number of observations or measurements made for some real-world phenomena that can be best described by some random variable V with an unknown probability distribution function (pdf) f.

Given a data set P of size n consisting of values from a domain $\mathcal{M}$, a kernel density estimate is a function f_P that for any input in $\mathcal{M}$ (not necessarily in P) describes *the density* at that location. It is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample. That said, we view P as a finite, independent and identically distributed (iid) data sample drawn from a random variable V that is governed by an unknown distribution f. We are interested in estimating the shape of this function f. The kernel density estimate f_P approximates the density of f at any possible input point $x \in \mathcal{M}$ [100,116]. Figure 1.1 visualizes the kernel density estimate (KDE) in both 1 and 2 dimensions, using real data sets (a web trace in 1D and a spatial data set from openstreetmap in 2D). Black dots represent a subset of points from P, and the blue curves or regions represent the KDE constructed from P.

We restrict the *kernels* $K_\sigma : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ that satisfy the following properties:

(K1) **symmetric and shift invariance:** There exists a function $k : \mathbb{R}^+ \to \mathbb{R}^+$ such that for any $p, x \in \mathbb{R}^d$, we have $K(p,x) = K(x,p) = k(\|p - x\|)$, where $\|p - x\|$ denotes the ℓ_2 distance between p and x.



(a) KDE in 1D. (b) KDE in 2D.

Figure 1.1. Kernel density estimate (KDE).

(K2) **monotonicity:** For $z < z'$ then $k(z) > k(z')$.

In addition to the above properties of the kernels, it is convenient to enforce one of two other properties. A *normalized* kernel satisfies

$$\int_{x \in \mathbb{R}^d} K_\sigma(p, x) dx = 1, \tag{1.1}$$

so that the kernel and the kernel density estimate are probability distributions. A *unit* kernel satisfies

$$K_\sigma(x, x) = 1 \text{ so that } 0 \leq K_\sigma(p, x) \leq 1, \tag{1.2}$$

which ensures that $\text{KDE}_P(x) \leq 1$. Unlike normalized kernel, the changing of bandwidth does not affect the coefficient of kernel function, so $K_\sigma(p, x) = k(\|p - x\|/\sigma)$. In this dissertation, we will enforce the kernel to be either *normalized* kernel or *unit* kernel in different chapters.

Examples of kernels include (described here for $\mathbb{R}^d$):

- Gaussian Kernel: $K_\sigma(p, x) = \frac{1}{\sigma^d (2\pi)^{d/2}} \exp(-\|x - p\|^2/\sigma^2)$,
- Laplacian Kernel: $K_\sigma(p, x) = \frac{1}{\sigma^d c_d d!} \exp(-\|x - p\|/\sigma)$,
- Triangular Kernel: $K_\sigma(p, x) = \frac{d}{\sigma^d c_{d-1}} \max\{0, 1 - \|x - p\|/\sigma\}$,
- Epanechnikov Kernel: $K_\sigma(p, x) = \frac{d+2}{2\sigma^d c_d} \max\{0, 1 - \|x - p\|^2/\sigma^2\}$, or
- Ball Kernel: $K_\sigma(p, x) = \{\frac{1}{\sigma^d c_{d-1}} \text{ if } \|p - x\| \leq \sigma; \text{o.w. } 0\}$,

where $c_d = \frac{r^d \pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}$ is the volume of the unit d-dimensional sphere. These are shown as normalized kernels; to make them unit kernels, the coefficient is simply set to 1.

We use the Gaussian kernel by default throughout the dissertation (the most widely used kernel in the literature), although some scenarios favor the Epanechnikov kernel [122, 126]. All kernel definitions have a σ term to controls the amount of data smoothing. Choosing the appropriate σ is an important problem and there is a large amount of literature on doing so [77, 113, 132]. In Chapter 3, we will investigate how to choose the bandwidth parameter for a kernel density estimate built on a subsample of a large data set.

- **Kernel density estimate**: Given such a kernel K_σ and a point set P in d-dimensions, a kernel density estimate is formally defined as a function KDE_P that for *any query* $x \in \mathbb{R}^d$ evaluates as

$$\text{KDE}_P(x) = \frac{1}{|P|} \sum_{p \in P} K_\sigma(p, x). \tag{1.3}$$

- **Kernel distance**: The kernel distance [52, 70, 78, 105] is a metric [95, 127] between two point sets P, Q (as long as the kernel used is characteristic [127], a slight restriction of being positive definite [6, 140], this includes the Gaussian and Laplace kernels). Define a *similarity* between the two point sets as

$$\kappa(P, Q) = \frac{1}{|P|} \frac{1}{|Q|} \sum_{p \in P} \sum_{q \in Q} K_\sigma(p, q). \tag{1.4}$$

Then the kernel distance between two point sets is defined as

$$D_K(P, Q) = \sqrt{\kappa(P, P) + \kappa(Q, Q) - 2\kappa(P, Q)}. \tag{1.5}$$

When we let point set Q be a single point x, then $\kappa(P, x) = \text{KDE}_P(x)$.

If K_σ is positive definite, it is said to have the reproducing property [6,140]. This implies that $K_\sigma(p, x)$ is an inner product in some reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$. Specifically, there is a lifting map $\phi : \mathbb{R}^d \to \mathcal{H}_K$ so that $K_\sigma(p, x) = \langle \phi(p), \phi(x) \rangle_{\mathcal{H}_K}$, and moreover, the entire set P can be represented as $\Phi(P) = \sum_{p \in P} \phi(p)$, which is a single element of $\mathcal{H}_K$ and has a norm $\|\Phi(P)\|_{\mathcal{H}_K} = \sqrt{\kappa(P, P)}$. A single point $x \in \mathbb{R}^d$ also has a norm $\|\phi(x)\|_{\mathcal{H}_K} = \sqrt{K(x, x)}$ in this space.

- **Kernel regression**: Kernel regression [96, 142] is a powerful non-parametric technique for understanding scalar-valued 1-dimensional (and higher dimensional) data sets. It has distinct advantages over linear or polynomial regression techniques in that it does not impose a possibly restrictive or over-fitting model on the data. Rather it uses a kernel similarity function to describe a smooth weighted average over the points. This allows the predicted function to locally adapt to the values of the data. These advantages have led to wide use of the kernel regression to predict, model, and visualize data from stocks [144] to weather monitoring [124] to quantified self [131].

We now consider an input data set $P \subset \mathbb{R}^{d+1}$. We decompose this into the first d explanatory coordinates denoted $P_x \subset \mathbb{R}^d$ and the last dependent one $P_y \subset \mathbb{R}$. Most examples in this dissertation we discuss have $d = 1$ where it is common to think of these data items P_x as times, but many approaches generalize for larger values of d. Then each data item $p \in P$ is also associated with a scalar data value p_y (the set of these comprises P_y). In this setting, P_x is the same as P in the previous KDE definition. We will only consider $P \subset \mathbb{R}^{d+1}$ in the circumstance of kernel regression, mainly in Chapter 5, and keep $P \subset \mathbb{R}^d$ in all the other chapters.

With the new setting, for any query point $q \in \mathbb{R}^d$,

$$\textsc{kde}_P(q) = \frac{1}{|P|} \sum_{p \in P} K(p_x, q). \tag{1.6}$$

We say a *weighted kernel density estimate* (WKDE) replaces this with a weighted sum

$$\textsc{wkde}_P(q) = \frac{1}{|P|} \sum_{p \in P} K(p_x, q) p_y. \tag{1.7}$$

Finally, the (Nadaraya-Watson) *kernel regression* function is defined for a query point $q \in \mathbb{R}^d$ as

$$\textsc{kr}_P(q) = \frac{\sum_{p \in P} K(p_x, q) p_y}{\sum_{p \in P} K(p_x, q)} = \frac{\textsc{wkde}_P(q)}{\textsc{kde}_P(q)}. \tag{1.8}$$

This maps each domain point in $\mathbb{R}^d$ to an estimate in the space $\mathbb{R}$ of scalar values; it takes a weighted average (defined by kernel similarity) of the scalar values nearby. Figure 1.2 visualizes the kernel regression and its original data of a synthetic data set with bandwidth 50 and 200.

There are various other forms of kernel regression [112], but in this dissertation we focus on the Nadaraya-Watson variety [96,142] as it has an important, long history and has been widely used in areas such as image processing [129] and economics [12]. Moreover,

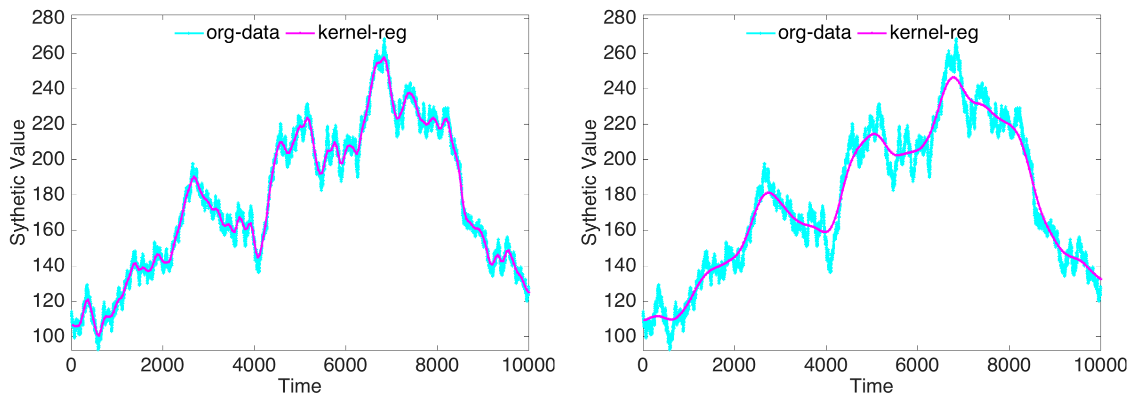


Figure 1.2. Kernel regression of synthetic data with bandwidth 50 (left) and 200 (right).

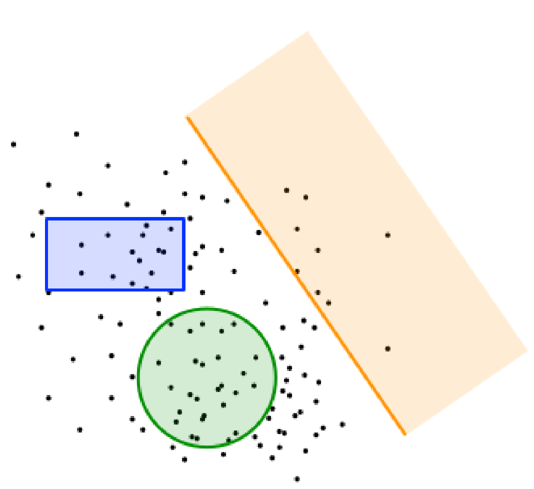
since it does not try to pass through every data point, it is the most robust to outliers that are pervasive in large data, which often by necessity cannot be carefully filtered.

1.2 Coresets

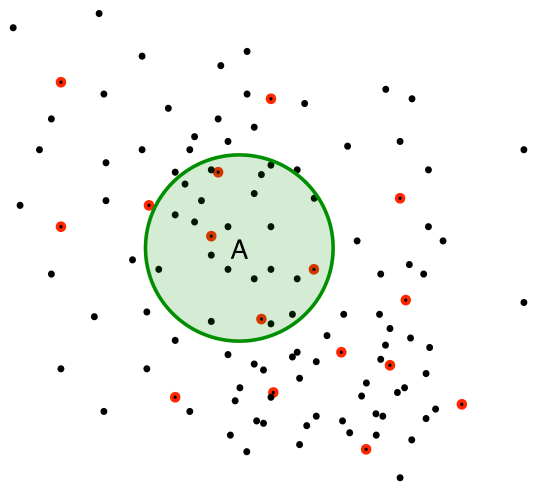
A *coreset* is a reduced data set that can be used as proxy for the full data set; the same algorithm can be run on the coreset as the full data set, and the result on the coreset approximates that on the full data set. It is often required or desired that the coreset is a subset of the original data set, but in some cases, this is relaxed. A weighted coreset is one where each point is assigned a weight, perhaps different than it had in the original set. A strong coreset provides error guarantees for all queries. ε-net and ε-sample are two examples of coresets defined in range space.

- **Range space**: A range space $(P, \mathcal{A})$ consists of a ground set P and a family of ranges $\mathcal{A}$ of subsets from P. In this dissertation, we consider ranges that are defined geometrically, for instance when P is a point set and $\mathcal{A}$ are all subsets defined by a ball, that is any subset of P which coincides with $P \cap B$ for any ball B. $\mathcal{A}$ can also be all subsets defined by an axis-aligned rectangle or a half space (Figure 1.3(a)).
- **ε-approximation (or ε-sample)**: Given a range space $(P, \mathcal{A})$, it is a subset Q such that $\left| \frac{|A \cap P|}{|P|} - \frac{|A \cap Q|}{|Q|} \right| \leq \varepsilon$ for any $A \in \mathcal{A}$.
- **ε-net**: Given a range space $(P, \mathcal{A})$, it is a subset $Q \subset P$, so for any $A \in \mathcal{A}$ such that $|A \cap P| \geq \varepsilon |P|$, then $A \cap Q \neq \emptyset$.

For instance, the data set P in Figure 1.3(b) may be a point set describing the location of



(a) Different types of ranges.



(b) An example of ε-sample, where the black points are original data points and the set of red points is an ε-sample.

Figure 1.3. Range Spaces and an example of ε-sample.

twitter users in Utah, and the family of subsets $\mathcal{A}$ may be all subsets of twitter users within a fixed radius from a query point. Such a subset is shaded in green in Figure 1.3(a). An ε-sample is a subset $Q \subset P$ of the ground set such that for any range $A \in \mathcal{A}$, the fraction of points from Q in A is different from the fraction of points from P in A by at most ε. Thus if the set of red points in Figure 1.3(b) is an ε-sample of twitter users in Utah, and we ask what fraction of the twitter users in Utah are within 20 miles of Salt Lake City, we can give an answer within ε error using the set of red points. Furthermore, the same number of samples would work for the entire state of Utah with the same guarantees and for any query disk (e.g., the fraction of Utah twitter users within 40 miles of Provo).

For the case of ε-net, P is still the point set describing the location of twitter users in Utah. Now we care about large enough events with many tweets, the area A such that $|A \cap P| \geq \varepsilon |P|$. Thus Q is a subset of twitter users that witness every large enough event.

1.3 Kernel Range Spaces and KDE Coreset

By smoothing out the boundary of the binary ranges, we introduce the smoother family of range spaces called kernel range spaces to deal with noisy data. Thus we cannot simply say a point is in a range or not; instead, we assign a value between $[0, 1]$ to a point in a range.

- **Kernel range space**: A kernel range space [78] is an extension of the combinational concept of a range space. $(P, \mathcal{K})$ defined by a point set $P \subset \mathbb{R}^d$ and a set of kernels $K(x, \cdot)$ represented by a fixed kernel K and an arbitrary center point $x \in \mathbb{R}^d$.

In the binary range space, for a range A centered at x, the fraction of the points in A is represented by $\frac{|A \cap P|}{|P|}$. In the kernel range space, it turns to be $\frac{1}{|P|} \sum_{p \in P} K_\sigma(p, x)$, where $K_\sigma(x, \cdot)$ is the corresponding range centered at x with bandwidth σ, which is exactly the definition of $\text{KDE}_P(x)$. Thus the definition of ε-sample of kernel density estimates (KDE coreset) is the same as ε-sample in kernel range spaces.

- **ε-sample in kernel range spaces (ε-sample of kernel density estimates)** [103]: Given a kernel range space $(P, \mathcal{K})$, it is a subset $Q \subset P$, such that

$$\max_{x \in \mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_Q(x)| = \|\text{KDE}_P - \text{KDE}_Q\|_\infty \le \varepsilon, \tag{1.9}$$

ε-sample in kernel range spaces can be very useful in many data-intensive applications, since evaluating a kernel density estimate over P directly takes $O(n)$, which can be prohibitively expensive in big data. So ε-sample in kernel range spaces gives us another point set Q, such that KDE_Q approximates KDE_P well. The error is guaranteed within user-defined parameter ε.

1.4 KDE in Geometric Inference

As we discussed before, kernel density estimates can be used in many areas. Here we give an example to show its power in geometric inference and topological data analysis.

Geometry and topology have become essential tools in modern data analysis: geometry to handle spatial noise and topology to identify the core structure. Topological data analysis (TDA) has found applications spanning protein structure analysis [46,86] to heart modeling [51] to leaf science [110], and is the central tool of identifying quantities like connectedness, cyclic structure and intersections at various scales. Yet it can suffer from spatial noise in data, particularly outliers.

Given an unknown compact set $S \subset \mathbb{R}^d$ and a finite point cloud $P \subset \mathbb{R}^d$ that comes from S under some process, geometric inference aims to recover topological and geometric properties of S from P. The offset-based (and more generally, the distance function-based) approach for geometric inference reconstructs a geometric and topological approximation of S by offsets from P (e.g., [23–27]).

Kernel density estimates were brought to geometric inference and TDA by paper [106]; as a coauthor of this paper, we first analyzed the stability of kernel density estimates and the kernel distance in the context of geometric inference. We accomplished this by showing that a similar set of properties hold for the kernel distance with respect to a measure μ, (in place of distance to a measure $d^{\mathrm{CCM}}_{\mu,m_0}$ [25]), defined as

$$d^K_\mu(x) = D_K(\mu, x) = \sqrt{\kappa(\mu,\mu) + \kappa(x,x) - 2\kappa(\mu,x)}. \tag{1.10}$$

It satisfies all the *distance-like* properties and there are further advantages of the kernel distance. (i) Its sublevel sets conveniently map to the superlevel sets of a kernel density estimate. (ii) It is Lipschitz with respect to the smoothing parameter σ when the input x is fixed. (iii) As σ tends to ∞ for any two probability measures μ, ν, the kernel distance is bounded by the Wasserstein distance: $\lim_{\sigma\to\infty} D_K(\mu,\nu) \leq W_2(\mu,\nu)$. Here W_2 is the *Wasserstein distance* [137]: $W_2(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \left(\int_{\mathbb{R}^d\times\mathbb{R}^d} ||x-y||^2 \mathrm{d}\pi(x,y)\right)^{1/2}$ between two measures, where $\mathrm{d}\pi(x,y)$ measures the amount of mass transferred from location x to location y and $\pi \in \Pi(\mu,\nu)$ is a transference plan [137].

Most importantly, it has a small coreset representation, which allows for sparse representation and efficient, scalable computation. In particular, an ε-sample in kernel range spaces for a measure μ: Q of μ is a finite point set whose size only depends on $\varepsilon > 0$, such that $\max_{x\in\mathbb{R}^d} |\mathrm{KDE}_\mu(x) - \mathrm{KDE}_Q(x)| = \|\mathrm{KDE}_\mu - \mathrm{KDE}_Q\|_\infty \leq \varepsilon$. These coresets preserve inference results, such as shapes, Betti numbers and persistence diagrams. Persistence diagrams summarize the persistence of all homological features, such as connected components, tunnels, voids, etc. of a topological space in a single diagram. The birth and death times are the y- and x-coordinates of the persistence diagram, thus significant features with high persistence are far from the diagonal. To calculate the persistent homology of a space, the space should first be represented as a simplicial complex, which is very time consuming. While there exist simplicial sparsification approaches, which reduce the size akin to coresets, they still involve the complicated process of building a simplicial complex of the full data set. Using KDE coresets is the first simplification result that preserves topological features without constructing the simplicial complex on the full data set, and thus can speed up TDA for large data significantly. Figure 1.4 give us such an example with persistence diagrams generated by TDA package for R [49].

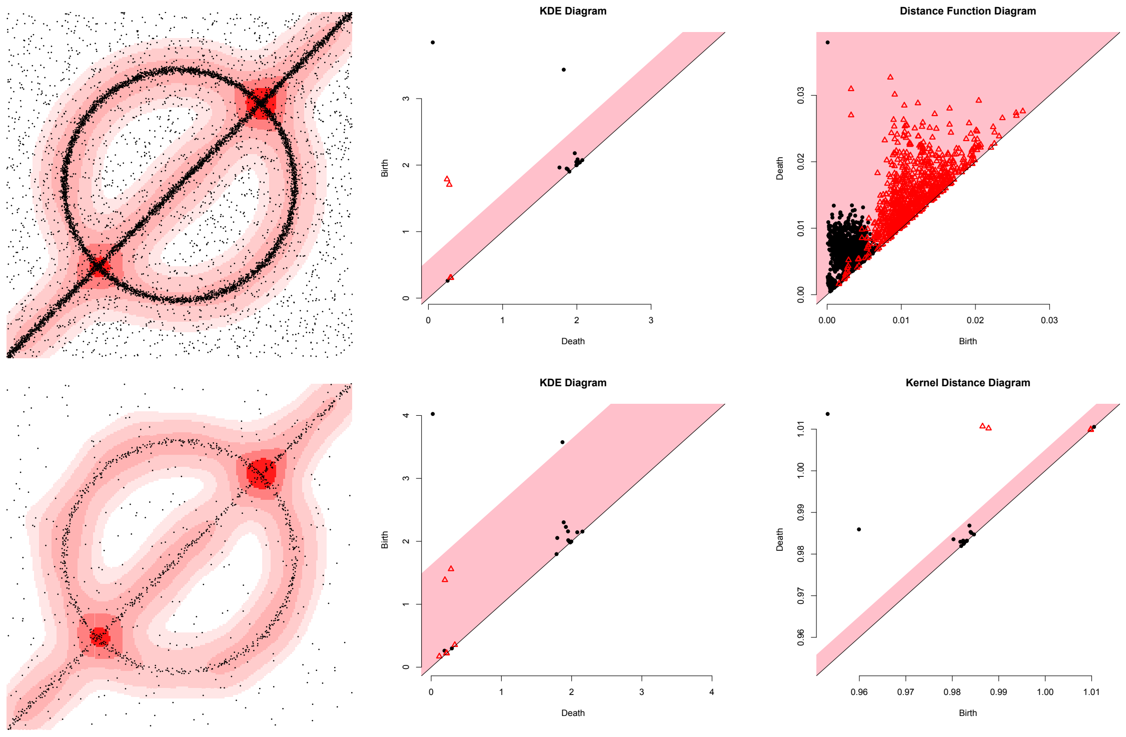


Figure 1.4. Example with 10,000 points in $[0,1]^2$ generated on a circle or line with $N(0,0.005)$ noise; 25% of points are uniform background noise. The generating function is reconstructed with KDE with $\sigma = 0.05$ (upper left), and its persistence diagram based on the superlevel set filtration is shown (upper middle). A coreset [149] of the same data set with only 1,384 points (lower left) and persistence diagram (lower middle) are shown, again using KDE. This associated confidence interval contains the dimension 1 homology features (red triangles) suggesting they are noise; this is because it models data as iid – but the coreset data is not iid, it subsamples more intelligently. We also show persistence diagrams of the original data based on the sublevel set filtration of the standard distance function (upper right, with no useful features due to noise) and the kernel distance (lower right).

1.5 Outline

This dissertation improves on the construction of coresets in kernel density estimates and kernel regression in several ways relevant to improve the efficiency and effectiveness. It builds several algorithmic techniques that can be used for coreset construction problem in kernel smoothing setting. It also illustrates these techniques on several large data sets.

Chapter 2 describes randomized and deterministic algorithms for computing ε-samples for kernel density estimates with quality guarantees that are orders of magnitude more efficient than previous algorithms. These algorithms do not require knowledge of the kernel or its bandwidth parameter and are easily parallelizable. We demonstrate how to implement our ideas in a centralized setting and in MapReduce, although our algorithms are applicable to any large-scale data processing framework. Extensive experiments on large real data sets demonstrate the quality, efficiency and scalability of our techniques. This chapter is mainly based on [149] with Jeffery Jestes, Jeff M. Phillips and Feifei Li.

Chapter 3 is an extension of the problem described in Chapter 1, which investigates the challenges in using L_∞ (or worst case) error, a stronger measure than L_1 or L_2. This chapter presents efficient solutions to two linked challenges: how to evaluate the L_∞ error between two kernel density estimates and how to choose the bandwidth parameter for a kernel density estimate built on a subsample of a large data set. This chapter is based on [150] with Jeff M. Phillips.

Chapter 4 extends smoothed versions of geometric range spaces to more general types of ranges and then considers approximation of these range spaces through ε-nets and ε-samples. We characterize when size bounds for ε-samples on kernels can be extended to these more general smoothed range spaces. We also describe new generalizations for ε-nets to these range spaces and show when results from binary range spaces can carry over to these smoothed ones. This chapter is based on [107] with Jeff M. Phillips.

Chapter 5 describes coresets for kernel regression. Kernel regression is an essential and ubiquitous tool for non-parametric data analysis, particularly popular among time series and spatial data. The size of the coresets for kernel regression is also independent of the raw number of data points, rather they only depend on the error guarantee, and in some cases the size of domain and amount of smoothing. We evaluate our methods on very large time series and spatial data, and demonstrate that they incur negligible error, can be constructed extremely efficiently and allow for great computational gains. This chapter is based on [151] with Jeff M. Phillips.

Chapter 6 concludes the dissertation by providing some applications and future directions.

CHAPTER 2

EFFICIENT CORESETS FOR KERNEL DENSITY ESTIMATES

2.1 Background and Related Work

Around the 1980s, kernel density estimates became the defacto way in statistics to represent a continuous distribution from a discrete point set [126], with the study initiated much earlier [116]. However, this work often implied brute force ($O(n)$ time) solutions to most queries.

The problem of evaluating kernel density estimates is a central problem in statistical physics and numerical analysis. These problems are often posed as n-body simulations where the force-interactions between all pairs of points need to be calculated [4], and the pairwise force is up to a constant described by the Gaussian kernel. This has resulted in many indexing type techniques that up to constant precisions can evaluate the kernel density estimate at all n points in roughly $O(n)$ time. These techniques are sometimes called *fast multipole methods* [19], and in practice these are typically implemented as quad trees that calculate the distance to roots of subtrees instead of all pairs when the distance becomes far enough. Numerical approximation techniques called the (Improved) Fast Gauss Transform (IFGT) [56,114,145,146] can further improve these approaches. However, the IFGT approach (in general fast multipole methods) is based on heuristics and does not offer formal theoretical guarantees on the approximation-time trade-off.

In order to have a formal theoretical guarantee to derive an (ε, δ)-approximation, random sampling is a baseline method, but it requires $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples to be included in Q, which could lead to expensive query evaluations, especially for small ε and/or δ values.

A recent technique using discrepancy theory [103] creates a small representation of a kernel density estimate (smaller than the random sampling approach) while still bounding the ℓ_∞ error. It works by creating a min-cost matching of points in P; that is, P is decomposed into $|P|/2$ pairs so that the sum over all distances between paired points is minimized. Then it randomly removes one point from each pair, reducing the size of P by half. This process is repeated until either the desired size subset or the tolerable error level is reached. However, computing the min-cost matching [48] is expensive, so this approach is only of theoretical interest and not directly feasible for large data. Yet this will serve as the basis for a family of our proposed algorithms.

A powerful type of kernel is a *reproducing kernel* [6, 101] (an example is the Gaussian kernel) that has the property that $K(p,q) = \langle p,q \rangle_{\mathcal{H}_K}$; that is, the similarity between objects p and q defines an inner-product in a *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}_K$. This inner-product structure (the so-called "kernel trick") has led to many powerful techniques in machine learning; see [119, 125] and references therein. Most of these techniques are not specifically interested in the kernel density estimate; however, the RKHS offers the property that a single element of this space essentially represents the entire kernel density estimate. Motivated by the task of constructing samples from Markov random fields, Chen et al. [30] introduced a technique called kernel herding suitable for characteristic kernels (including Gaussian and Laplace kernels). Bach et al. [7] showed that this algorithm can be interpreted under the Frank-Wolfe framework [33]. Harvey and Samadi [68] further revisited kernel herding in the context of a general mean approximation problem in $\mathbb{R}^{d'}$. That is, consider a set P' of n points in $\mathbb{R}^{d'}$, find a subset $Q' \subset P'$ so that $\|\overline{P'} - \overline{Q'}\| \leq \varepsilon$, where $\overline{P'}$ and $\overline{Q'}$ are the Euclidean averages of P' and Q', respectively.

Kernel density estimates have been used in the database and data mining community for density and selectivity estimations, e.g., [61, 147]. However, the focus in these works is how to use kernel density estimates for approximating range queries and performing selectivity estimation, rather than computing approximate kernel density estimates for fast evaluations. When the end-objective is to use a kernel density estimate to do density or selectivity estimation, one may also use histograms [60, 75, 81, 109] or range queries [54, 55, 71, 139] to achieve similar goals, but these do not have the same smoothness and statistical properties of kernel density estimates [126]. Nevertheless, the focus of this work is on computing approximate kernel density estimates that enable fast query evaluations, rather than exploring how to use kernel density estimates in different application scenarios (which is a well-explored topic in the literature).

2.2 Warm-up: One Dimension

Efficient construction of approximate kernel density estimates in one dimension is fairly straightforward. However, it is still worth investigating these procedures in more detail since to our knowledge, this has not been done at truly large scale before, and the techniques developed will be useful in understanding the higher dimensional version.

- **Baseline method: random sampling (RS).** A baseline method for constructing an approximate kernel density estimate in one dimension is random sampling. It is well known that [30,103] if we let Q be a random sample from P of size $O((1/\varepsilon^2)\log(1/\delta))$, then with probability at least $1-\delta$, the random sample Q ensures that $\|\text{KDE}_P - \text{KDE}_Q\|_\infty \le \varepsilon$.

That said, the first technique (RS) follows from this observation directly and just randomly samples $O((1/\varepsilon^2)\log(1/\delta))$ points from P to construct a set Q. In the centralized setting, we can employ the one pass reservoir sampling technique [138] to implement RS efficiently. For large data that is stored in distributed nodes, RS can still be implemented efficiently using the recent results on generating random samples from distributed streams [35].

The construction cost is $O(n)$. The approximate KDE has a size $O((1/\varepsilon^2)\log(1/\delta))$, and its query cost (to evaluate $\text{KDE}_Q(x)$ for any input x) is $O((1/\varepsilon^2)\log(1/\delta))$.

RS can be used as a preprocessing step for any other technique, e.g., for any technique that constructs a KDE over P, we run that technique over a random sample from P instead. This may be especially efficient at extremely large scale (where $n \gg 1/\varepsilon^2$) and where sampling can be done in an efficient manner. This may require that we initially sample a larger set Q than the final output to meet the approximation quality required by other techniques.

- **Grouping selection (GS).** A limitation in RS is that it requires a large sample size (sometimes the entire set) in order to guarantee a desired level of accuracy. As a result, its size and query cost becomes expensive for small ε and δ.

Hence, we introduce another method, called the *grouping selection* (GS) method. It leverages the following lemma, known as the γ-perturbation.

Lemma 1. *Consider n arbitrary values* $\{\gamma_1, \gamma_2, \dots, \gamma_n\}$ *such that* $\|\gamma_i\| \le \gamma$ *for each* $i \in [n]$. *Then* *let $Q = \{q_1, q_2, \ldots, q_n\}$ such that $q_i = p_i + \gamma_i$ for all $p_i \in P$. Then $\|\text{KDE}_P - \text{KDE}_Q\|_\infty \leq \gamma/\sigma$.*

Proof. This follows directly from the $(1/\sigma)$-Lipschitz condition on kernel K (which states that the maximum gradient of K is $(1/\sigma)$), hence perturbing all points by at most γ affects the average by at most γ/σ. □

Using Lemma 1, we can select one point q in every segment ℓ of length $\varepsilon\sigma$ from P and assign a weight to q that is proportional to the number of points from P in ℓ, to construct an ε-sample KDE of P. Specifically, GS is implemented as follows. After sorting P if it is not already sorted, we sweep points from smallest to largest. When we encounter p_i, we scan until we reach the first p_j such that $p_i + \varepsilon\sigma < p_j$. Then we put p_i (or the centroid of p_i through p_{j-1}) in Q with weight $w(p_i) = (j - i)/n$. Since Q constructed by GS is weighted, the evaluation of $\text{KDE}_Q(x)$ should follow the weighted query evaluation as specified in equation 1.7.

Theorem 1. *The method GS gives an ε-sample of kernel density estimate of P.*

Proof. The weighted output of GS Q corresponds to a point set Q' that has $w(q)$ unweighted points at the same location of each $q \in Q$; then $\text{KDE}_Q = \text{KDE}_{Q'}$. We claim that Q' is an $\varepsilon\sigma$-perturbation of P, which implies the theorem.

To see this claim, we consider any set $\{p_i, p_{i+1}, \ldots, p_{j-1}\}$ of points that are grouped to a single point $q \in Q$. Since all of these points are within an interval of length at most $\varepsilon\sigma$, each $p_{i+\ell}$ is perturbed to a distinct point $q'_{i+\ell} \in Q$ (at location q) that is at distance $\gamma_{i+\ell} \leq \varepsilon\sigma$. □

GS's construction cost is $O(n)$ if P is sorted, or $O(n \log n)$ otherwise. Its query cost is $O(|Q|)$, which in the worst case could be $|Q| = |P|$. And Q may be large depending on how densely points in P are co-located and the values of ε and σ. However, GS can be used as a postprocessing step on top of any other method, such as using GS over the output of RS. This takes little overhead if the points are already sorted, such as in the output of SS (see below).

- **Sort-selection (SS).** Our best method (SS) offers an ε-sample of kernel density estimate using only $O(\frac{1}{\varepsilon})$ samples in one dimension, a significant improvement over random sampling. Consider a one-dimensional sorted point set $P = \{p_1, p_2, \ldots, p_n\}$ where $p_i \leq$

p_{i+1} for all i. Let $P_j = \{p_i \in P \mid (j-1)\varepsilon n < i \leq j\varepsilon n\}$ for integer $j \in [1, \lceil 1/\varepsilon \rceil]$ such that $P = \cup P_j$. Then the coreset $Q = \{q_1, q_2, \ldots, q_{\lceil 1/\varepsilon \rceil}\}$ such that each $q_j = p_{\lceil (j-\frac{1}{2})\varepsilon n \rceil}$ from P.

With this coreset, it has the following result:

Lemma 2. *The method SS gives an ε-sample of kernel density estimate of P, such that $\|\text{KDE}_P - \text{KDE}_Q\|_\infty \leq \varepsilon$.*

We now can construct the SS method. It simply selects $p_{\lceil (j-\frac{1}{2})\varepsilon n \rceil}$ from P into Q for each $j \in [1, \lceil 1/\varepsilon \rceil]$. This requires that P is sorted, and this can be done efficiently at very large scale using external merge sort. However, we can do better. Note that ε-approximate quantiles for $\lceil 1/\varepsilon \rceil$ quantile values are sufficient to construct Q, and we can use an efficient streaming or distributed algorithm for computing these ε-approximate quantiles [34, 57, 74, 87, 88]. In particular, we only need to find the $\frac{\varepsilon n}{2}$th, $\frac{3\varepsilon n}{2}$th, $\frac{5\varepsilon n}{2}$th, $\ldots$, $(n - \frac{\varepsilon n}{2})$th quantile values from P. And it is easy to verify that ε-approximations of these quantiles are sufficient to establish the result in Lemma 2.

Using the ε-approximate quantiles, SS has a construction cost of $O(n \log \frac{1}{\varepsilon})$; otherwise its construction cost is $O(n \log n)$. In either case, its size is only $O(\frac{1}{\varepsilon})$ and its query cost is also just $O(\frac{1}{\varepsilon})$.

2.2.1 Efficient Evaluation

Once we have obtained a set Q above so KDE_Q approximates KDE_P, we need to efficiently answer queries of $\text{KDE}_Q(x)$ for any $x \in \mathbb{R}$. The first obvious choice is a brute force computation (BF) where we directly calculate $\frac{1}{|Q|}\sum_{q \in Q} K(q, x)$. This has little overhead and its cost is obviously $O(|Q|)$. It is most efficient if $|Q|$ is particular small.

A second approach (MP) is to use the one-dimensional variant of multipole methods. We build a B-tree (or binary tree if Q fits in memory) on Q. Each node v will store the weight (or count) w_v of all nodes in the subtree and the smallest v_s and largest v_l coordinates of the subtree. We traverse the tree as follows, starting at the root. If $x \in [w_s, w_l]$, visit each child and return their sum to the parent. If $|K(v_s, x) - K(v_l, x)| \leq \varepsilon$, then return $w_v K((v_l - v_s)/2, x)$ to the parent. Else, visit each child and return their sum to the parent. This approach may improve the query evaluation time in practice, especially when $|Q|$ is large.

2.3 Two Dimensions

In $\mathbb{R}^2$, we first describe baseline methods from the literature or based on simple extensions to existing methods. We then introduce our new methods. The first uses a randomized technique based on matchings, and the second is deterministic and based on space-filling curves.

2.3.1 Baseline Methods

- **Random Sampling (RS).** The first baseline (labeled RS) is to simply random sample a set Q from P. The same bound of $O((1/\varepsilon^2)\log(1/\delta))$ on the sample size from one dimension still holds in two-dimensions, although the constant in the big-Oh is likely larger by a factor of about 2.
- **Improved Fast Gauss Transform (IFGT).** A class of methods is based on *fast multipole methods* [19]. In practice in two dimensions these are implemented as quad trees which calculate the distance to roots of subtrees instead of all pairs when the distance becomes far enough. The Improved Fast Gauss Transform (IFGT) [56,114,145,146], is the state-of-the-art for fast construction and evaluation of approximate kernel density estimates (although only with Gaussian kernels). It improves multipole approaches by first building a k-center clustering over the data set P, and then just retaining a Hermite expansion of the kernel density estimates for the points associated with each k-centers. However, the IFGT method is based on heuristics and does not offer any formal theoretical guarantees on the approximation-size trade-off. As a result, it involves a number of parameters that cannot be easily and intuitively set in order to derive a desired level of efficiency and accuracy tradeoffs.
- **Kernel herding (KH).** Yet another possible approach is to explore the *reproducing kernel Hilbert space* (RKHS). As discussed in Section 2.1, the technique "kernel herding" [7,30, 68] showed that iteratively and greedily choosing the point $p \in P$, which when added to Q most decreases the quantity $\|\mathrm{KDE}_P - \mathrm{KDE}_Q\|$ in RKHS. However, this still takes $O(|Q|n)$ time to construct $|Q|$ since at each step each point in $P \setminus Q$ needs to be evaluated to determine how much it will decrease the error.

2.3.2 Randomized MergeReduce Algorithm

An interesting theoretical result built on discrepancy [28, 92] theory was recently proposed in [103] for constructing a small set Q so KDE_Q approximates KDE_P. It extends a classic method for creating ε-approximations of Chazelle and Matousek [29] (see also [28, 92]), to ε-sample of kernel density estimates. These results are mostly of theoretical interest, the straightforward adaption is highly inefficient. Next we explain and overcome these inefficiencies.

2.3.2.1 The MergeReduce framework

Our algorithm leverages the framework of Chazelle and Matousek [29] and its generalizations. We first describe our overall framework, and then elaborate the most critical *reduce step* in further details. Roughly speaking, our algorithm repeatedly runs a merge-then-reduce step. Hence, we denote this framework as the *MergeReduce* algorithm.

Suppose the desired size of the compressed set Q is $|Q| = k$. The framework proceeds in three phases: an initialization phase, the combination phase, and the optional clean-up phase. The first phase is implicit, and the last phase is of theoretical interest only.

In the initialization phase, we arbitrarily decompose P into disjoint sets of size k; call these $P_1, P_2, \ldots, P_{n/k}$. Since this is arbitrary, we can group data that is stored together into the same partition. This works well for distributed data or streaming data where consecutively encountered data are put in the same partition.

The combination phase proceeds in $\log(n/k)$ rounds. In each round, of the remaining sets of size k, it *arbitrarily* pairs them together, which we dub the *merge step*. For each pair, say P_i and P_j, it runs *a reduce step* on the union of $2k$ points to create a single set of size k. At a high level, the *reduce step* has two parts. The first part is what we call a *matching operation*. It produces k pairs of points in a certain fashion over the $2k$ input points. The second part is trivial: it randomly selects one point from each pair to produce the final output of size k. The *merging operation* is the most critical part in a reduce step, and we will elaborate after presenting the overall MergeReduce framework.

That said, after i rounds of *merge and reduce*, there are $(n/k)/2^i$ sets of size k. This continues until there is one set remaining. Note that again, the fact that we can pair sets (P_i and P_j) arbitrarily in a *merge step* is extremely convenient in a distributed or streaming setting.

The clean-up phase is not needed if the combination phase is run as above; see Section 2.3.2.4 for remaining details.

Importantly, we also note that this entire MergeReduce framework can be preceded by randomly sampling $O(1/\varepsilon^2)$ points from P which are then treated as the input. Then only $\log(1/\varepsilon)$ merge-reduce rounds will be needed.

- **The min-cost matching.** The key part of this framework is to construct a *matching* in a set of points P. Suppose $|P| = n$, a *matching* consists of $\frac{n}{2}$ pairs of points, and every point in P belongs to exactly one pair in a *matching*.

The recent result from [103] implies that one can use the min-cost matching to derive an ε-sample of kernel density estimate. Note that a min-cost matching is a matching so that the sum of distances between matched points is minimized. Unfortunately, by using a min-cost matching, the algorithm in [103] is impractical. Here is why.

It is well known that a min-cost matching over n points can be done in $O(n^3)$ time using Edmonds' Blossom algorithm [48]. This quite complicated algorithm involves non-regular recursion, and it is clearly not scalable for large data sets. We use the state-of-the-art implementation [80] as a baseline for the matching operation and label it as Blossom-MR (MergeReduce with Blossom min-cost matching). This implementation of the Blossom algorithm requires first calculating $K(p, p')$ for all $O(n^2)$ pairs $p, p' \in P$ as input, which is part of the overall cost (which is $O(n^3)$).

There have been theoretical improvements [136] to Edmonds' algorithm for points in $\mathbb{R}^2$. These algorithms are considerably more complicated than that of Edmonds and no known efficient implementation exists; most likely the improvements are theoretical in nature only.

We have the following results concerning the Blossom-MR algorithm, and the Blossom-MR+RS algorithm that first randomly samples $O(1/\varepsilon^2)$ points.

Theorem 2. *For a point set $P \subset \mathbb{R}^2$ with n points, we can construct Q giving an ε-sample of KDE (with constant probability) in*

- $O(\frac{n}{\varepsilon^2}\log^2 n \log\frac{1}{\varepsilon})$ *time and* $|Q| = O(\frac{1}{\varepsilon}\log n\sqrt{\log\frac{1}{\varepsilon}})$ *using Blossom-MR, and*
- $O(n + \frac{1}{\varepsilon^4}\log^3\frac{1}{\varepsilon})$ *time and* $|Q| = O(\frac{1}{\varepsilon}\log^{1.5}\frac{1}{\varepsilon})$ *using Blossom-MR+RS.*

Lastly, the following greedy algorithm provides a two-approximation to the cost of the min-cost matching [42]. It finds the closest pair of points in P, matches them, removes them from P, and repeats. This algorithm can be implemented in $O(n^2 \log n)$ time as follows. Calculate all $O(n^2)$ pairwise distances and place them in a priority queue. Repeatedly remove the smallest pair from the queue (in $O(\log n)$ time). If both points are still in P, match them and mark them as no longer in P. We refer to the merge-reduce framework with this matching algorithm as the Greedy-MR method. Greedy-MR does improve the running time over Blossom-MR, however, it is still quite expensive and not scalable for large data. Furthermore, the result produced by Greedy-MR is not known to provide any approximation guarantees on the kernel density estimate.

2.3.2.2 More Efficient Reduce Step

The Blossom-MR algorithm and its heuristic variant Greedy-MR are too expensive to be useful for large data. Thus, we design a much more efficient *matching operation*, while still ensuring an ε-sample of kernel density estimate.

For any matching M, we produce an *edge map* E_M of that matching M as $E_M = \{e(p,q) \mid (p,q) \in M\}$ where $e(p,q)$ is an undirected edge connecting p and q. Given a disk B, for $e(p,q) \in E_M$ define $e(p,q) \cap B$ as follows.

- If both p and q are not covered by B, $e(p,q) \cap B = \emptyset$.
- If both p and q are covered by B, $e(p,q) \cap B = e(p,q)$.
- If either p or q is covered by B but not both. Suppose p is covered by B, and $e(p,q)$ intersects the boundary of B at a point s, $e(p,q) \cap B = e(p,s)$.

Then, we define $E_M \cap B$ as:

$$E_M \cap B = \{e(p,q) \cap B \mid e(p,q) \in E_m\}. \tag{2.1}$$

An example of $E_M \cap B$ is shown in Figure 2.1. Essentially, we only want it to consider edges with at least one endpoint within B, and only the subset of the edge that is within B. We observe that in order to produce an ε-sample of kernel density estimate, the property the matching requires is in regards to any unit disk B. Specifically, we want

$$C_{M,B} = \sum_{e(p,q) \in E_M \cap B} \|p - q\|^2 \tag{2.2}$$

to be small. Let $C_M = \max_B C_{M,B}$.

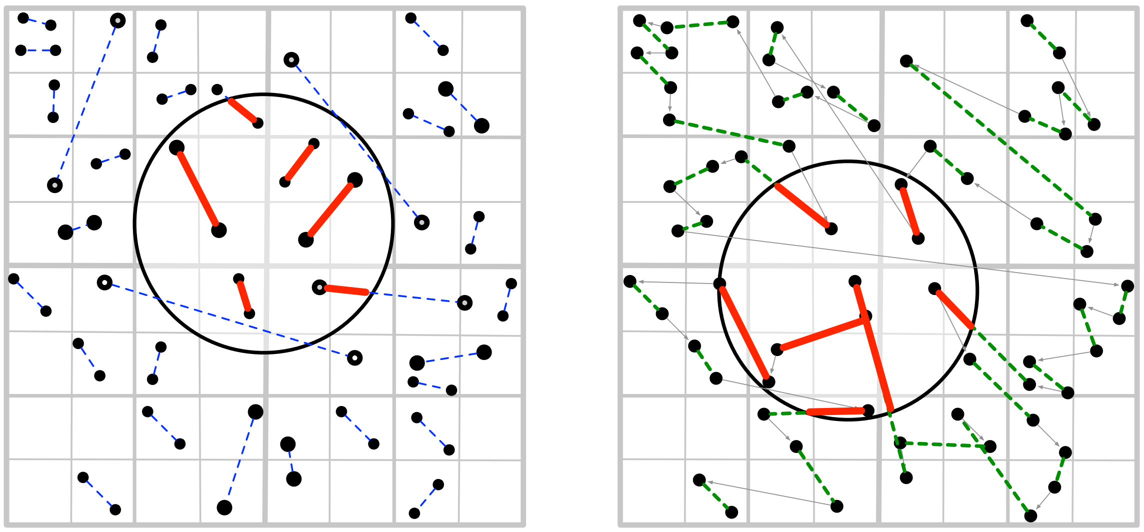


Figure 2.1. Intersection between a matching and a disk, solid red lines are included in $E_M \cap B$. Left shows Grid matching and right shows Z-order with every other edge in a matching.

The result in [103] says if M is the min-cost matching (minimizes $\sum_{(p,q)\in M} \|p - q\|$), then $C_M = O(1)$. But calculating the min-cost matching, both exactly and approximately, is expensive as we have shown in Section 2.3.2.1.

- **The grid matching.** Here we present a novel solution which does have a bound on C_M and is efficient, including at very large scales. We progress in rounds until all points are matched. Starting with $i = 0$, in round i, we consider a grid G_i on $\mathbb{R}^2$ where each grid cell has edge length $l_{\varepsilon,i} = \sqrt{2}\sigma\varepsilon 2^{i-2}$. Define cell $g_{r,c} \in G_i$ as $[rl_{\varepsilon,i}, (r+1)l_{\varepsilon,i}] \times [cl_{\varepsilon,i}, (c+1)l_{\varepsilon,i}]$ for integers r, c. Inside of each cell, match points arbitrarily. Only the unmatched points (one from any cell with an odd number of points) survive to the next round. Each cell in G_{i+1} is the union of 4 cells from G_i, thus in rounds $i > 0$, it can have at most 4 points. We refer to this matching algorithm as Grid; see an example in Figure 2.1. For simpler analysis, we assume $\sigma > \varepsilon^c$ for some constant $c \geq 1$; typically $\sigma \gg \varepsilon$ and $\varepsilon < 1$, so this assumption is very mild. Alternatively, if we do not know σ ahead of time, we can recursively divide points that fall in the same grid cell into smaller and smaller levels until at most two are left in each cell (like a quad tree), and then move back up the recursion stack only with unmatched points; a careful implementation can be done in $O(n \log n)$ time [65]. Let P_0 be unmatched points after round 0, let $P' = P \setminus P_0$, and Q

the final output.

Lemma 3. *Let $M'(P')$ be the matching on P' in Grid. For each edge $(p, q) \in M'(P')$, let $\hat{P}$ be where (w.l.o.g.) q is moved to location p. Then $\hat{P}$ is a $\varepsilon\sigma/2$-perturbation of P'.*

Proof. Since all points matched in round 0 are in a grid cell of size at most $\varepsilon\sigma\sqrt{2}/4$, the point q in any edge is moved at most $\sqrt{2} \cdot \varepsilon\sigma\sqrt{2}/4 = \varepsilon\sigma/2$. □

Lemma 4. *Let $M_0(P_0)$ be the matching by Grid on P_0. Then $C_{M_0} = O(\log(1/\varepsilon))$ and Grid runs in $O(n \log \frac{1}{\varepsilon})$ time.*

Proof. In each round, there are at most 2 matched pairs per grid cell. Each such pair has edge length at most $\sqrt{2} l_{\varepsilon,i} = \sigma\varepsilon 2^{i-1}$, and there are at most $(1/\sigma\varepsilon 2^{i-5/2})^2$ grid cells that intersect any unit ball. Thus the total squared-length of all matchings in round i is at most $((1/\sigma\varepsilon 2^{i-5/2})^2 \cdot (\sigma\varepsilon 2^{i-1})^2 = 2\sqrt{2}$. After $\log(1/\sigma\varepsilon) + 1$ rounds, the total length of all matchings is at most $2\sqrt{2}(\log(1/\sigma\varepsilon) + 1)$, and in any ball, there are at most 4 remaining unmatched points. The last 4 points can each account for at most a squared-length of 4 within a unit ball B, so the total weight of edges in any unit ball B is at most $C_M \leq 2\sqrt{2}\log(1/\sigma\varepsilon) + 19 = O(\log(1/\varepsilon))$. Each round takes $O(n)$ time, and we can match points arbitrarily in $O(n)$ time after the $\log(1/\sigma\varepsilon) + 1$ rounds. □

We observe in most common scenarios C_M is close to 1.

With the Grid matching algorithm, we can instantiate the MergeReduce framework to get a Grid-MR algorithm, or if we first sample a Grid-MR+RS algorithm.

Theorem 3. *For a point set $P \subset \mathbb{R}^2$ with n points, we can construct Q giving an ε-sample of KDE (with constant probability) in*

- *$O(n \log \frac{1}{\varepsilon})$ time and $|Q| = O(\frac{1}{\varepsilon} \log n \log^{1.5} \frac{1}{\varepsilon})$ using Grid-MR, and*
- *$O(n + \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})$ time and $|Q| = O(\frac{1}{\varepsilon} \log^{2.5} \frac{1}{\varepsilon})$ using Grid-MR+RS.*

Since Grid takes only $O(n \log \frac{1}{\varepsilon})$ time, the benefit of the initialization phase to split the data set into n/k pieces does not out-weigh its overhead in a centralized setting. In particular, we just run Grid once on all n_i points remaining in round i. This does not affect the runtime or error bounds.

Compared to the Blossom-MR and Greedy-MR algorithms, Grid-MR produces an ε-sample of kernel density estimate with about the same size, but with much cheaper construction cost. Grid-MR's running time only linearly depends on n, making it an ideal candidate to scale out to massive data.

2.3.2.3 Streaming and Distributed MergeReduce

Since the reduce step (the key computational component of the MergeReduce framework) is only run on select subsets, this allows the full framework to generalize to distributed and streaming settings.

- **Streaming variant.** The streaming algorithm follows techniques for approximate range counting [8, 128]. Consecutive points are assigned to the same partitions P_i, and we pair and reduce partitions whenever there are two that represent the same number of points (one that has been merged i times represents $k2^i$ points). This means we only need to keep $\log \frac{n}{k}$ partitions, and thus only $O(k \log \frac{n}{k})$ points in memory at any given time. The dependence on n can be completely removed by first randomly sampling.

The final structure of the streaming algorithm has weighted points, where if a point is in a partition that has been reduced i times, its weight is 2^i. These weights can be removed to create just $5|Q|$ points instead of $|Q| \log \frac{|P|}{|Q|}$ by running the in memory matching algorithm on weighted points [90].

In particular, we can modify a matching algorithm to work with weighted points, specifically consider the Grid algorithm. In the 0th phase of Grid, a point with weight 2^i represents 2^i points that all fall in the same cell, and can be matched with themselves (this can be done by ignoring this point until the ith phase when its weight is 1).

- **Distributed variant.** This framework is efficient on distributed data. Use the streaming algorithm on each distributed chunk of data, and then pass the entire output of the streaming algorithm to a central node, which can merge and reduce the union. The error caused by reducing on the central node is already accounted for in the analysis. Again, the dependence on $|P|$ can be removed by first sampling.

2.3.2.4 Other Extension

An alternative version of the combination phase is possible for the MergeReduce algorithm. Specifically, it considers some reduce step that takes time $O(n^\beta)$ on a set of size n, and instead sets $k = 4(\beta + 2)|Q|$ (where $|Q|$ is the desired size of the final output), and on every $(\beta + 3)$th round, pairs sets but does not reduce them. Then the clean-up phase is used to reduce the single remaining set repeatedly until it is of size $|Q|$. When β is a constant, this saves a $O(\log n)$ from the size of the output Q (or $O(\log \frac{1}{\varepsilon})$ if we sampled first) [29].

More specifically, the output of this MergeReduce variant is then a set of size $|Q| = O(C_M \frac{1}{\varepsilon} \log^{0.5} \frac{1}{\varepsilon})$ for a reduce step that uses a matching algorithm which produces an output with cost C_M. If we use Grid, then Grid-MR produces an output of size $|Q| = O(\frac{1}{\varepsilon} \log^{1.5} \frac{1}{\varepsilon})$. And Blossom-MR outputs Q of size $|Q| = O(\frac{1}{\varepsilon} \log^{0.5} \frac{1}{\varepsilon})$ in $O(\frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon})$ time.

But in practice, this variant is more complicated to implement and usually the overhead out-weighs its benefit. Hence we do not use this variant in this paper.

2.3.3 Deterministic Z-Order Selection

Inspired by one-dimensional sort-section (SS) and randomized two-dimensional Grid-MR algorithm, we propose a new deterministic two-dimensional technique based on space filling curves. A space filling curve puts a single order on two- (or higher-) dimensional points that preserves spatial locality. They have great uses in databases for approximate high-dimensional nearest-neighbor queries and range queries. The single order can be used for a (one-dimensional) B+ tree, which provides extremely efficient queries even on massive data sets that do not fit in memory.

In particular, the Z-order curve is a specific type of space filling curve that can be interpreted as implicitly ordering points based on the traversal order of a quad tree. That is, if all of the points are in $[0,1]^2$, then the top level of the quad tree has four children over the domains $c_1 = [0, \frac{1}{2}] \times [0, \frac{1}{2}]$, $c_2 = [\frac{1}{2}, 1] \times [0, \frac{1}{2}]$, $c_3 = [0, \frac{1}{2}] \times [\frac{1}{2}, 1]$, and $c_4 = [\frac{1}{2}, 1] \times [\frac{1}{2}, 1]$. And each child's four children itself divide symmetrically as such. Then the Z-order curve visits all points in the child c_1, then all points in c_2, then all points in c_3, and all points in c_4 (in the shape of a backwards 'Z'); and all points within each child are also visited in such a Z-shaped order. See an example in Figure 2.1. Thus, this defines a complete order on them, and the order preserves spatial locality as well as a quad tree does.

The levels of the Z-order curve (and associated quad tree) are reminiscent of the grids used in the matching technique Grid. This insight leads to the following algorithm.

Compute the Z-order of all points, and of every two points of rank $2i$ and $2i+1$, discard one at random; repeat this discarding of half the points until the remaining set is sufficiently small. This approach tends to match points in the same grid cell, as with Grid, but is also algorithmically wasteful since the Z-order does not change between rounds.

Thus, we can improve the algorithm by using insights from SS. In particular, we just run SS on the Z-order of points. So to collect $k = |Q|$ points, let the ith point retained $q_i \in Q$ be the point in rank order $\lceil (i - \frac{1}{2})\frac{|P|}{k} \rceil$. This selects one point from each set of $\frac{|P|}{k}$ points in the Z-order.

In fact, by setting $k = O(\frac{1}{\varepsilon} \log n \log^{1.5} \frac{1}{\varepsilon})$, if we randomly select one point from each Z-order rank range $[(i-1)\frac{|P|}{k}, i\frac{|P|}{k}]$ (call this algorithm Zrandom), then the resulting set Q has about the same guarantees as the Grid-MR algorithm, including Zrandom+RS, which preprocesses by random sampling.

Theorem 4. *For a point set $P \subset \mathbb{R}^2$ with n points, we can construct Q giving an ε-sample of KDE (with constant probability) in*

- *$O(n \log n)$ time and $|Q| = O(\frac{1}{\varepsilon} \log n \log^{1.5} \frac{1}{\varepsilon})$ using Zrandom, and*
- *$O(n + \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})$ time and $|Q| = O(\frac{1}{\varepsilon} \log^{2.5} \frac{1}{\varepsilon})$ using Zrandom+RS.*

Proof. We prove this result by imagining that Zrandom does something more complicated than it actually does in order to relate it to Grid-MR. That is, we pretend that instead of just selecting a single point at random from each range $[(i-1)|P|/|Q|, i|P|/|Q|]$ in the Z-order rank, we proceed in a series of $\log(|P|/|Q|)$ rounds, and in each round, the set P is reduced in size by half. Specifically, in each round, out of every two consecutive points in the Z-order (rank $2i$ and $2i+1$), we retain one at random. Since the Z-order is consistent across rounds, this results in a random point in the rank interval $[(i-1)|P|/|Q|, i|P|/|Q|]$ as desired.

Now if we consider the levels of the implicit quad tree defining the Z-order, this is equivalent to the grid used in Grid-MR. In each round, there are at most 3 edges within grid cell at level j, but that are not entirely in levels smaller than j. Since we still only care about $O(\log(1/\varepsilon))$ levels of the grid, the squared distance of these edges in that cell level accounts for at most $O(1)$ inside a unit square. Thus, C_M is still at most $O(\log(1/\varepsilon))$. The remainder of the proof is the same as in Theorem 3. $\square$

However, we find the implementation of the following deterministic algorithm to be more effective; but as the randomness is necessary for the proof, we do not provide bounds. The construction of the Z-order can be done efficiently using its interpretation as bit-interleaving. Specifically, the *z-value* of a point is calculated by interleaving bits from the most significant bit to the least significant bit in the binary representation of a point's coordinates. For example, a two-dimensional point $(3,5)$ expressed in its binary representation is $(011,101)$. Its *z-value* is then $(011011)_2 = 27$.

Then we do not need to completely sort all points by z-value in order to select the proper k points, we can just approximately do this so that we have one point selected from each set of $\frac{|P|}{k}$ points in the sorted order. This can be done using an ε-approximate quantiles algorithm that is accurate up to $\varepsilon = 1/k$. This guarantees the existence in the quantile structure and its identification of at least one point within every consecutive set of $\frac{|P|}{k}$ points. We can just return this point for each range $[(i-1)\frac{|P|}{k}, i\frac{|P|}{k}]$ of ranks. We refer to this method as Zorder. Note that, following the discussion for SS in Section 2.2, we can use any of the existing efficient, large-scale ε-approximate quantiles algorithms in either the centralized or the distributed setting.

2.3.4 Efficient Query Evaluation

We can do efficient evaluations in $\mathbb{R}^2$ similar to BF and MP in $\mathbb{R}^1$, as discussed in Section 2.2.1. In fact, BF is exactly the same. MP uses a quad tree instead of a binary tree. It stores a bounding box at each node instead of an interval, but uses the same test to see if the difference between $K(x,\cdot)$ is within ε on the furthest and closest point in the bounding box to decide if it needs to recur.

2.3.5 Parallel and Distributed Implementation

As with one-dimensional methods, our two-dimensional methods can be efficiently implemented in distributed and streaming settings. Any algorithm using the MergeReduce framework can run in distributed and parallel fashion. As a result, Grid-MR, Zrandom, and Zorder are very efficient in any distributed and parallel environments, and they extend especially well for the popular MapReduce framework where data in each split is processed in a streaming fashion.

Among the baseline methods, RS can easily be implemented in any distributed and parallel environments. It takes some effort to make IFGT run in distributed and parallel fashion, but it is possible; we omit the details. Lastly, the KH can be approximated (without any bounds) by running its greedy step in each local piece of data independently, and then merging the results from local nodes.

2.3.6 Higher Dimensions

All two-dimensional algorithms described can be extended to higher dimensions. In particular, KH, RS, Blossom-MR, Greedy-MR extend with no change, while Grid-MR and Zorder-MR extend in the obvious way of using a d-dimension grid or space-filling curve. In $\mathbb{R}^d$, the theoretical size bounds for RS increases to $O(\frac{1}{\varepsilon^2}(d + \log \frac{1}{\delta}))$ [78]; Grid-MR, and Zrandom-MR increases to $O(d^{d/2}/\varepsilon^{2-\frac{4}{d+2}} \log^{1+\frac{d}{d+2}} \frac{1}{\varepsilon} \log n)$ (and a $\log \frac{1}{\varepsilon}$ factor less for Blossom-MR). The increase in the second set is due to only being able to bound

$$C_M^d = \max_B \sum_{e(p,q)\in E_M \cap B} \|p - q\|^d$$

instead of equation (2.2) since the number of grid cells intersected by a unit ball now grows exponentially in d, and thus we need to balance that growth with the dth power of the edge lengths. The stated bounds, then results from [103] with an extra $\log n$ factor (which can be turned into a $\log \frac{1}{\varepsilon}$ by first random sampling) because we do not use the impractical process described in Section 2.3.2.4. In no case does the MergeReduce framework need to be altered, and so the construction times only increase by a factor d.

2.4 Experiments

We test all methods in both the single-thread, centralized environment and the distributed and parallel setting. In the centralized case, we implemented all methods in C++ and obtained the latest implementation of the IFGT method from the authors in [114, 145, 146]. We then used MapReduce as the distributed and parallel programming framework. In particular, we implemented and tested all methods in a Hadoop cluster with 17 machines. The centralized experiments were executed over a Linux machine running a single Intel i7 cpu at 3.20GHz. It has 6GB main memory and an 1TB hard disk. The distributed and parallel experiments were executed over a cluster of 17 machines running Hadoop 1.0.3. One of the 17 machines has an Intel Xeon(R) E5649 cpu at 2.53 GHz, 100 GB of main memory, and a 2TB hard disk. It is configured as both the master node and the name node of the cluster. The other 16 machines in the Hadoop cluster (the slave nodes) share the same configuration as the machine we used for the centralized experiments. One TaskTracker and DataNode daemon run on each slave. A single NameNode and JobTracker run on the master. The default HDFS (Hadoop distributed file system) chunk size is 64MB.

- **Data sets.** We executed our experiments over two large real data sets. In two dimensions, we used the OpenStreet data from the OpenStreetMap project. Each data set represents the points of interest on the road network for a US state. The entire data set has the road networks for 50 states, containing more than 160 million records in 6.6GB. For our experiments, we focus on only the 48 contiguous states, excluding Hawaii and Alaska. Each record is a two-dimensional coordinate, represented as 2 4-byte floating points. We denote this data as the *US* data set.

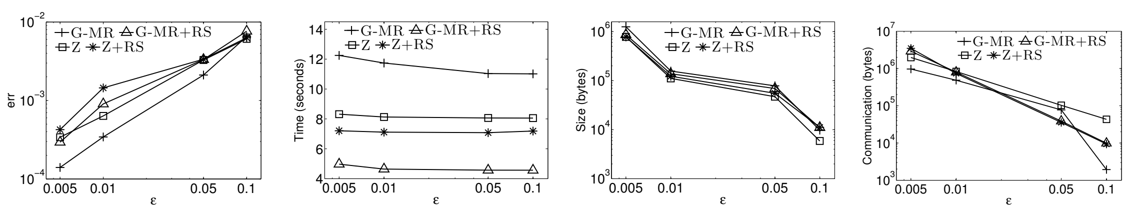
In one dimension, we employed the second real data set, *Meme*, which was obtained from the Memetracker project. It tracks popular quotes and phrases which appear from various sources on the Internet. The goal is to analyze how different quotes and phrases compete for coverage every day and how some quickly fade out of use while others persist for long periods of time. A record has 4 attributes, the URL of the website containing the memes, the time Memetracker observed the memes, a list of the observed memes, and links accessible from the website. We preprocess the *Meme* data set, and convert each record to have an 8-byte double to represent the *time* of a single observed meme and the URL of the website which published the meme, e.g. if an original record contained a list of 4 memes published at a given time at a website, 4 records would be produced in the new data set. We view these records as a set of timestamps in 1d, reflecting the distribution of the Meme's publication time. After this preprocessing step, the *Meme* data set contains more than 300 million records in 10.3GB.

In both 1d and 2d, whenever needed, we randomly sample records from the full *US* or the full *Meme* data set to obtain a data set of smaller size. Figure 1.1 visualizes the kernel density estimates built by our MergeReduce algorithms in 1d and 2d, over the full *Meme* and *US* data sets, respectively (but only very few data points were plotted, to ensure that the figures are legible).

- **General setup.** In all experiments, unless otherwise specified, we vary the values of one parameter of interest, while keeping the other important parameters at their default values. Also by default, we randomly generate 5,000 test points to evaluate the accuracy of an approximate kernel density estimate. Among these 5,000 points, 4,000 were randomly selected from the data set P, and the other 1,000 were randomly selected from the domain space $\mathcal{M}$ of P (but not from P itself). We use *err* to denote the *observed* ℓ_∞ *error* from an approximate kernel density estimate Q, which is computed from the evaluations of these 5,000 test points in KDE_Q and KDE_P, respectively. We try to compare different methods by setting a proper ε value (the desired error in theory) for each of them so that they achieve the same observed error. All experiments report the *average of 10 random trials*.

2.4.1 Two Dimensions: Centralized

- **Default setup.** Our default data set is a *US* data set with 10 million records. The default failure probability δ for the random sampling method (RS) is set to 0.001. To save space in the legend in all figures, we used G-MR and Z to denote the Grid-MR and Zorder methods, respectively, and method+RS to denote the version of running method over a random sample of P (of size $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$), instead of running method over P directly. The default σ value in any kernel density estimate is 200, on a domain of roughly 50,000 $\times$ 50,000.
- **Our method+RS.** We first study the effect of running our methods over a random sample of P, when compared against the results from running them directly over P. Figure 2.2 shows the results when we vary the value of the common input parameter for all of them, ε. Not surprisingly, as shown in Figure 2.2(a), the observed errors in all methods are smaller than the desired error ε. All methods produce smaller observed errors when smaller ε values were used. Furthermore, under the same ε value, G-MR+RS and Z+RS produce higher errors than their respective counterpart does, namely, G-MR and Z. However, the difference is fairly small. In fact, when *err* is about 10^{-2}, there are almost no difference between G-MR+RS (Z+RS) and G-MR (Z).



(a) *err* vs. ε. (b) Construction time. (c) Size. (d) Communication.

Figure 2.2. Effect of guaranteed error ε on the centralized G-MR, G-MR+RS, Z, Z+RS.

More importantly, however, running a method over a random sample of P saves valuable construction time as shown in Figure 2.2(b). Figure 2.2(c) indicates that the sizes of the final approximate kernel density estimates constructed by different methods are almost the same. This is because whether running a method over P or a random sample of P, the final size of the kernel density estimate Q is largely determined by ε only. This also means that the query time of these methods is almost the same, since the query time depends on only $|Q|$. Hence, we have omitted this figure.

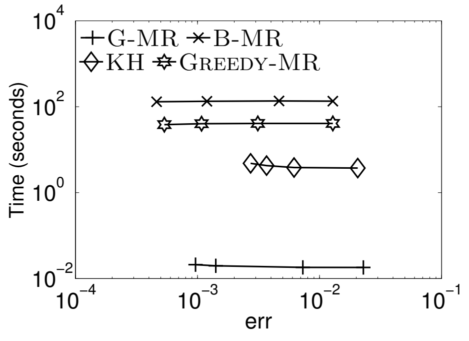
Finally, we also investigate the impact to the communication cost if we run these methods in a cluster. Figure 2.2(d) shows that this impact is not obvious. Since G-MR and Z are very effective in saving communication cost already, collecting a random sample first does not lead to communication savings. In contrast, doing so might even hurt their communication cost (if the sample size is larger than what they have to communicate in our MergeReduce framework). Nevertheless, all methods are very efficient in terms of the total bytes sent: a few megabytes in the worst case when $\varepsilon = 0.005$ over a cluster for 10 million records in P.

In conclusion, these results show that in practice, one can use our method+RS to achieve the best balance in construction time and accuracy for typically required observed errors.

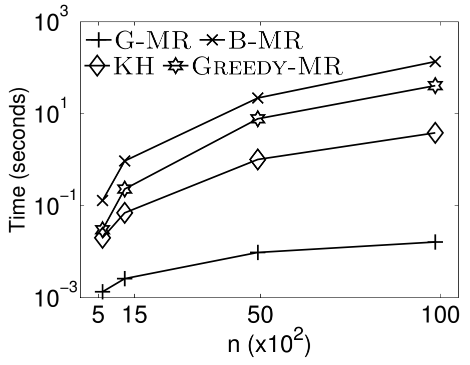
- **Our method vs. baseline methods.** We first investigate the construction cost of different alternatives in instantiating our MergeReduce framework, with a different algorithm for the matching operation. We also include Kernel Herding (KH) as introduced in Section 2.3.1. In order to let these baseline methods complete in a reasonable amount of time, we used smaller data sets here. From the analysis in Section 2.3.2, Blossom-MR (denoted as B-MR, representing the theoretical algorithm [103]) with a $O(nk^2)$ cost for output size k and Greedy-MR with a $O(nk \log n)$ cost are much more expensive than G-MR. Similarly, KH with a $O(nk)$ cost is also much more expensive than G-MR which runs in roughly $O(n \log k)$ time. This is even more pronounced in Figures 2.3(a) and 2.3(b), showing the construction time of different methods when varying the observed error *err* and the size of the data sets, respectively. G-MR is several orders of magnitude faster and more scalable than the other methods.

So the only competing baseline methods we need to consider further are the RS and IFGT methods. We compare these methods against our methods, G-MR+RS and Z+RS in Figure 2.4, using the default 10 million *US* data set. Figure 2.4(a) indicates that, to achieve the same observed error, RS is the most efficient method in terms of the construction time. However, our methods G-MR+RS and Z+RS are almost as efficient. In contrast, IFGT is almost one order of magnitude slower. In terms of the query time, Figure 2.4(b) shows that all methods achieve a similar query time given the same observed error, though IFGT does slightly better for very small *err* values. Note that we used the *multipole* (MP) query evaluation method for the kernel density estimates built from RS, G-MR+RS, and Z+RS. On the other hand, Figure 2.4(c) shows that the kernel density estimates built from both IFGT and RS have much larger size than that produced in our methods G-MR+RS and Z+RS, by 2 to 3, and 1 to 2 orders of magnitude, respectively. Finally, Figure 2.4(d) indicates that all methods have very good scalability in their construction time when the data set size increases from 1 million to 20 million, but G-MR+RS, Z+RS, and RS are clearly much more efficient than IFGT.

In conclusion, our methods are almost as efficient as RS in terms of building a kernel

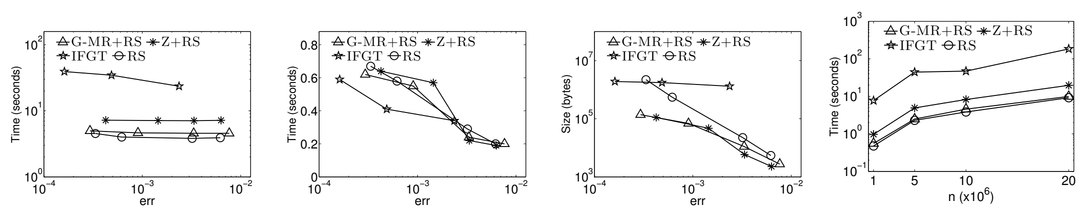


(a) Build time vs. *err*.



(b) Build time vs. *n*.

Figure 2.3. Centralized G-MR, B-MR, Greedy-MR, KH.



(a) Construction time vs *err*. (b) Query Time. (c) Size. (d) Construction time vs *n*.

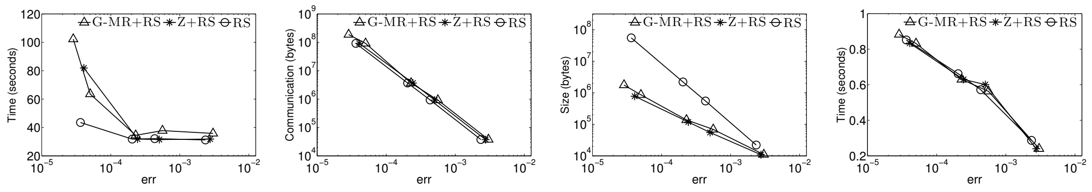
Figure 2.4. Effect of measured ℓ_∞ error *err* and N on centralized IFGT, RS, G-MR+RS, Z+RS.

density estimate, and they are much more efficient than IFGT. Our methods also share similar query time as IFGT and RS, while building much smaller kernel density estimates (in size) than both IFGT and RS.

2.4.2 Two Dimensions: Parallel and Distributed

- **Default setup.** In this case, we change the default data set to a *US* data set with 50 million records, while keeping the other settings the same as those in Section 2.4.1. Furthermore, since IFGT is much slower in building a kernel density estimate (even more so for larger data sets), and it is also a heuristic without theoretical guarantees (in contrast to the other 3 methods), in the distributed and parallel setting, we focus on comparing our methods against the RS method. Moreover, IFGT works only for the Gaussian kernel and needs to be provided the bandwidth σ ahead of time, whereas our methods and RS do not. For space, we omit experiments showing varying σ has mild effects on our algorithms and RS, but can lead to strange effects in IFGT.

- **Our methods vs. RS.** We compare the performance of our methods, G-MR+RS and Z+RS, against RS on the aforementioned Hadoop cluster. Figure 2.5 reports the results when we vary the observed error *err* for different methods (by adjusting their ε values properly so they output roughly the same *err*). Figure 2.5(a) shows that RS is the most efficient method to construct an approximate kernel density estimate for small *err* values, but G-MR+RS and Z+RS become almost as efficient as RS once *err* is no smaller than 10^{-4} which is sufficient for many practical applications. In those cases, all three methods take less than 40 seconds to construct an approximate KDE for 50 million records. All methods are highly communication-efficient, as shown in Figure 2.5(b). There are almost



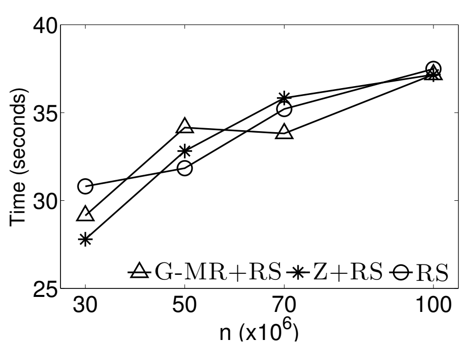
(a) Construction time. (b) Communication. (c) Size. (d) Query time.

Figure 2.5. Effect of measured ℓ_∞ error *err* on distributed and parallel G-MR+RS, Z+RS, RS.

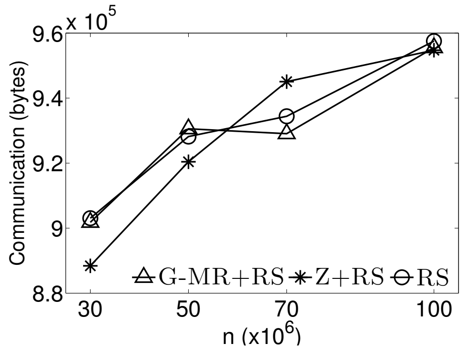
no difference among the 3 methods: they communicate only a few MBs over the cluster to achieve an *err* of 10^{-4}, and tens or hundreds of KBs for *err* between 10^{-2} and 10^{-3}. In terms of the size of the approximate KDE_Q, not surprisingly, RS is always the largest. By our analysis in Sections 2.3.2 and Sections 2.3.3, $|Q|$ is $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ for RS, and only $O(\frac{1}{\varepsilon}\log^{2.5}\frac{1}{\varepsilon})$ for both G-MR+RS and Z+RS. This is clearly reflected in Figure 2.5(c), where $|Q|$ is 1-2 orders of magnitude larger from RS than from G-MR+RS and Z+RS. Finally, we investigate the query time using Q in Figure 2.5(d). In general, the query time should be linear to $|Q|$. But in practice, since we have used the fast query evaluation technique, the *multipole* (MP) method as shown in Section 2.3.4, all three methods have similar query time.

We thus conclude that our methods, G-MR+RS and Z+RS, perform better than RS. More importantly, they also have much better theoretical guarantees (in order to achieve to same desired error ε in theory), which is critical in practice since users typically use a method by setting an ε value, and for the same ε value, our methods will outperform RS by orders of magnitude ($O(\frac{1}{\varepsilon}\log^{2.5}\frac{1}{\varepsilon})$ vs. $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$). Nevertheless, to be as fair as possible, we experimentally compared methods by first running a few trials to set a proper ε value so each method has roughly the same observed error *err*.

We also study the scalability of these methods in Figure 2.6 by varying the size of the data from 30 million to 100 million records. Not surprisingly, all three methods are extremely scalable in both their construction time (almost linear to n) and communication cost (almost constant to n). Communication only increases by 0.8×10^5 bytes when the total communication is about $9 - 10 \times 10^5$ bytes; such increase is due to the overhead in Hadoop to handle more splits (as n increases), rather than the behavior of our algorithms, which is independent from n in communication cost.



(a) Time.



(b) Communication.

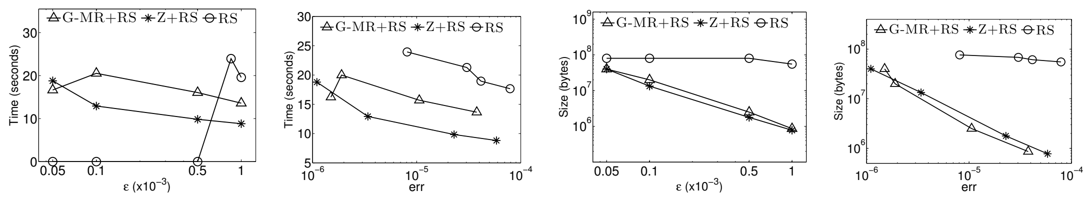
Figure 2.6. Effect of n on G-MR+RS, Z+RS, RS.

2.4.3 Two Dimensions: Is RS Good Enough?

To show our advantages over random sampling, we provide more centralized experiments here for higher precision. The trends in the parallel and distributed setting will be similar.

Getting higher precision results from a kernel density estimate can be very desirable. The error of kernel density estimates over samples is with respect to $\text{KDE}_P(x)$, but for large spatial data sets, often only a small fraction of points have non-negligible effect on a query x, so dividing by $|P|$ can make $\text{KDE}_P(x)$ quite small. Here it can be of critical importance to have very small error. Another use case of kernel density estimates is in n-body simulations in statistical physics [4], where high precision is required to determine the force vector at each step. Furthermore, note that a user typically uses these methods with a desirable error as the input, which is set as the input error parameter, the ε value; even though the observed error *err* on a data set may be smaller than ε. In that case, all of our methods have (roughly) a $O(\frac{1}{\varepsilon} / \log \frac{1}{\varepsilon})$ factor improvement in the KDE size, which is a critical improvement especially when ε needs to be small (for high precision applications).

We observe experiments in Figure 2.7 which compares G-MR+RS and Z+RS with RS in terms of construction time and size of the samples. (Note that figures for query time were omitted for the interest of space; but not surprisingly, they are roughly linear to the KDE size). We plot the figures based on input error parameter ε and observed error *err*. For the plot with respect to ε (Figure 2.7(a), 2.7(c)), when ε becomes smaller than $5 * 10^{-4}$, the construction time and size for RS remain constant as the sample size needed for RS becomes larger than the size of the original data set. Since we then don't need to sample,



(a) Construction Time. (b) Construction Time. (c) Size. (d) Size.

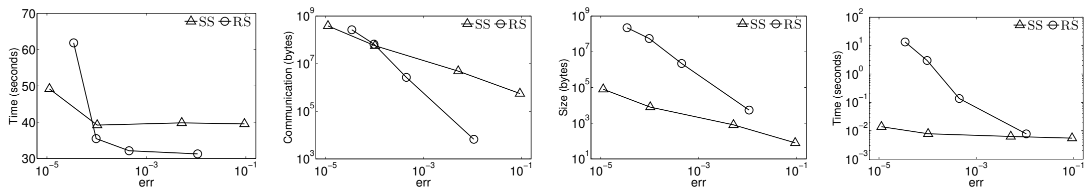
Figure 2.7. Effect of ε and ℓ_∞ error *err* on G-MR+RS, Z+RS, RS for high precision case.

the construction time and observed error for RS are 0. For small ε values, G-MR+RS and Z+RS are clever enough to test if random sampling is beneficial, and if not, the random sampling step is bypassed.

For the higher precision observed error, the results (Figures 2.7(b), 2.7(d)) clearly demonstrate the superiority of our proposed methods over RS, reducing both construction time and saving orders of magnitude in terms of both query time and size. We cannot achieve higher precision for RS when the observed error is smaller than 10^{-5}, since in those cases, ε is small enough that the random sample size is as big as the size of the original data set (i.e., KDE from a random sample consists of the entire original data set, leading to no error).

2.4.4 One Dimension: Parallel and Distributed

- **Default setup.** We now shift our attention in 1d. The default data set is *Meme* with 50 million records, $\delta = 0.001$, and σ at 1 day, over 273 days of data. Since GS (grouping selection) is a complementary technique that works with any other method, we focus on RS and SS (sort selection). We only report the results from the parallel and distributed setting. The trends in the centralized setting are similar.
- SS **vs.**RS**.** By varying the observed error *err*, Figure 2.8 compares the two methods across different metrics. To achieve smaller observed errors in constructing KDE_Q, SS is more efficient as shown in Figure 2.8(a), but RS becomes faster for larger observed errors. A similar trend is observed for the communication cost in Figure 2.8(b). In terms of reducing the size of Q, SS does a much better job than RS as shown in Figure 2.8(c), $|Q|$ in SS is 1-4 orders of magnitude smaller than $|Q|$ in RS, the gap is particularly large for smaller observed errors. This translates to the query time in Figure 2.8(d), where evaluating $\text{KDE}_Q(x)$ using Q produced by SS is much faster than doing so over Q from RS for most observed errors. When *err* becomes large, around 10^{-2}, the RS method



(a) Construction time. (b) Communication. (c) Size. (d) Query time.

Figure 2.8. Effect of varying measured *err* on RS, SS on one-dimensional data.

catches up, corresponding to the sizes of Q becoming closer.

In conclusion, SS in general performs better than or comparable to RS on observed error. But it has much better theoretical guarantees in size and query time as shown in Section 2.2 ($\frac{1}{\varepsilon}$ vs. $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$), which is critical in practice since users generally use a method by setting an ε value.

CHAPTER 3

$L\infty$ ERROR AND BANDWIDTH SELECTION FOR KDES

In the second chapter, in order to speed up evaluating kernel density estimate, we produce a *coreset* representation Q of the data which can be used as proxy for the true data P while guaranteeing approximation error on size and runtime. The size of Q depends only on the required error, not on any properties of P; these go beyond just randomly sampling Q from P. Written concretely, given P, and some error parameter $\varepsilon > 0$, the goal is to construct a point set Q to ensure

$$L_\infty(P, Q) = err(P, Q) = \max_{x \in \mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_Q(x)| \le \varepsilon, \tag{3.1}$$

or $err(P, \sigma, Q, \omega)$ if the bandwidths σ and ω for KDE_P and KDE_Q are under consideration. This line of work shows that an L_∞ error measure, compared with L_1 or L_2 error, is a more natural way to assess various properties about kernel density estimates. They assume σ is given and ω is always the same as σ. However, this is not necessarily true. In this chapter, we will investigate how to choose a bandwidth ω for $\text{KDE}_Q(x)$ under L_∞ error given P, σ, Q.

Thus, we empirically study two concrete problems:

1. Given two point sets $P, Q \subset \mathbb{R}^d$ and a kernel K, find the points $x = \arg\min_x err(P, Q)$.

2. Given two point sets $P, Q \subset \mathbb{R}^d$, a kernel K, and a bandwidth σ, estimate $\omega = \arg\min_\omega err(P, \sigma, Q, \omega)$.

It should be apparent the first problem is a key subproblem for the second, but it is also quite interesting in its own right. We will observe L_∞ is a strictly stronger measure than L_1 or L_2, yet can still be assessed. To the best of our knowledge, we provide the first rigorous empirical study of how to measure this L_∞ error in practice in an efficiently way, following theoretical investigations demonstrating it should be possible.

Bandwidth parameter is hugely important in the resulting KDE, and hence, there have been a plethora of proposed approaches [15, 38, 40, 41, 45, 50, 62–64, 73, 76, 82, 89, 117, 118, 121–123, 126, 130, 141, 148] to somehow automatically choose the "correct" value. These typically attempt to minimize the L_2 [122, 126] or L_1 error [40, 41] (or less commonly, other error measures [89]) between KDE_P and some unknown distribution μ that it is assumed P is randomly drawn from. Perhaps unsurprisingly, for such an abstract problem, many different methods can produce wildly different results. In practice, many practitioners choose a bandwidth value in an ad-hoc manner through visual inspection and domain knowledge.

In this chapter, we argue that the choice of bandwidth should not be completely uniquely selected. Rather, this value provides a choice of scale at which the data is inspected, and for some data sets, there can be more than one correct choice depending on the goal. We demonstrate this on real and synthetic data in one and two dimensions. As an intuitive one-dimensional example, given temperature data collected from a weather station, there are very obvious modal trends at the scale of 1 day and at the scale of 1 year, and depending on which phenomenon one wishes to study, the bandwidth parameter should be chosen along the corresponding scale, so it is totally reasonable if we assume σ for KDE_P is given.

Via examinations of problem (2), we observe that in some cases (but not all), given P, Q, and σ, we can choose a new bandwidth ω (with $\omega > \sigma$) so that $err(P, \sigma, Q, \omega)$ is significantly smaller than the default $err(P, \sigma, Q, \sigma)$. This corresponds with fine-grained phenomenon disappearing with less data (as $|Q| < |P|$), and has been prognosticated by theory about L_2 [126] or L_1 [40] error where the optimal bandwidth for KDE_Q is a strictly shrinking function of $|Q|$. Yet we urge more caution than this existing bandwidth theory indicates since we only observe this phenomenon in specific data sets with features present at different scales (like the daily/yearly temperature data example in Section 3.4).

- **Organization.** Section 3.1 formalizes and further motivates the problem. Section 3.2 addresses problem (1), and Section 3.3 problem (2). Then Section 3.4 describes detailed experimental validations of our proposed approaches. Finally, Section 3.5 provides some concluding thoughts.

3.1 Background and Motivation

3.1.1 Unit Kernels or Normalized Kernels?

Unit kernels are more natural to estimate the L_∞ errors of kernel density estimates [103, 146] since the range of values are in $[0, 1]$. For normalized kernels as σ varies, the only bound in the range is $[0, \infty)$.

Moreover, unit kernels, under a special case, correspond to the total variation distance of probability measures. In probability theory, the total variation distance for two probability measures P and Q on a sigma-algebra $\mathcal{F}$ of subsets of sample space Ω is defined as:

$$\delta(P, Q) = \sup_{A \in \mathcal{F}} |P(A) - Q(A)|. \tag{3.2}$$

Terms $P(A)$, resp. $Q(A)$, refer to the probability restricted to subset A. If we use $\mathcal{F}$ as the set of all balls of radius σ, so A is one such ball, then $P(A)$ is the fraction of points of P falling in A. Hence $P(A)$ can be viewed as the $\textsc{kde}_{P,\sigma}(x)$ under the ball kernel, where x is the center of ball A. When Q is the coreset of P, then $Q(A)$ is the fraction of points of Q falling in A, so it can be viewed as the $\textsc{kde}_{Q,\sigma}(x)$ under the ball kernel. In this sense, the total variance distance is the L_∞, specifically $err(P, Q)$ where K is the ball kernel. The total variation distance can also be mapped to other unit kernels if $\mathcal{F}$ can admit weighted subsets, not just subsets.

However, normalized kernels are more useful in bandwidth selection. In this case, there is a finite value for $\sigma \in (0, \infty)$ which minimizes the L_1 or L_2 error between $\textsc{kde}_{P,\sigma}$ and $\textsc{kde}_{Q,\sigma}$, whereas for unit kernels, this is minimized for $\sigma \to 0$.

But recall unit and normalized kernels are only different in the scaling coefficient, so given one setting, it is simple to convert to the other without changing the bandwidth. Hence we use *both* types of kernels in different scenarios: unit kernels for choosing the coresets, and normalized kernel for problem (1) and problem (2).

3.1.2 Why σ Is Given?

Recall that problem (2) takes as given two point sets P and Q as well as a bandwidth σ associated with P, and then tries to find the best bandwidth ω for Q so that $\textsc{kde}_{P,\sigma}$ is close to $\textsc{kde}_{Q,\omega}$. This is different from how the "bandwidth selection problem" is typically posed [40, 126] : a single point set Q is given with no bandwidth, and it is assumed that Q

is drawn randomly from an unknown distribution.

We break from this formulation for two reasons. First, we often consider the point set Q chosen as a coreset from P, and this may not be randomly from P; in fact, more intricate techniques [103, 149] can obtain the same error with much smaller size sets Q. These non-random samples break most modeling assumptions essential to the existing techniques.

Second, the choice of bandwidth may vary largely within the same data set, and these varied choices may each highlight a different aspect of the data. As an extended example, consider temperature data (here we treat a reading of 50 degrees as 50 data points at that time) from a MesoWest weather station KSLC read every hour in all of 2012. This results in 8760 total readings, illustrated in Figure 3.1. For three bandwidth values of 3, 72, and 1440, KDEs are shown to represent daily, weekly, and yearly trends. All are useful representations of the data; there is no "one right bandwidth.". Section 3.4 shows a two-dimensional example of population densities where similarly there are several distinct reasonable choices of bandwidths.

3.1.3 Why L_∞ Error?

As mentioned, the most common error measures for comparing KDEs are the L_1 or L_2 error defined for $p = \{1, 2\}$ as

$$L_p(P, Q) = \|\text{KDE}_P - \text{KDE}_Q\|_p = \left(\int_{x \in \mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_Q(x)|^p \right)^{1/p}. \tag{3.3}$$

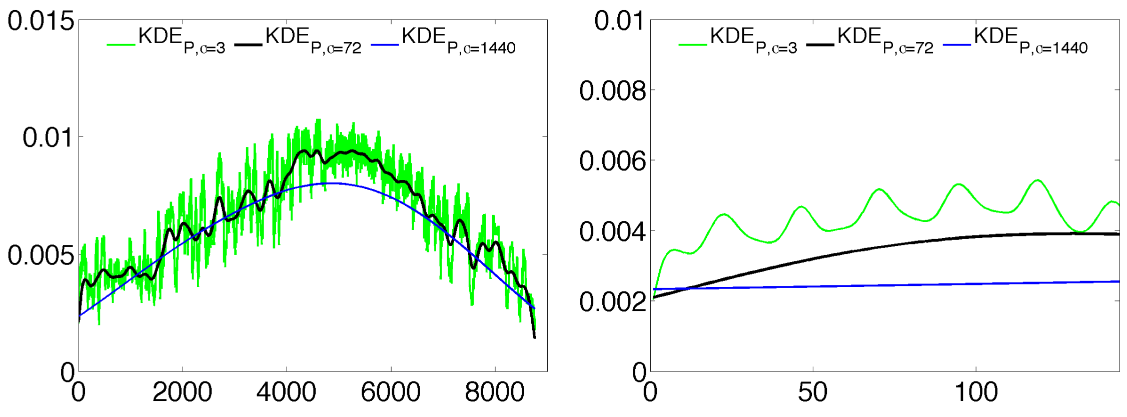


Figure 3.1. KDEs with different bandwidths showing daily, weekly, and yearly trends. Left shows a full year of data, and right shows one week of data.

Although this integral can be reasoned about, it is difficult to estimate precisely. Rather many techniques only evaluate at the points P and simply calculate

$$\Big(\frac{1}{|P|}\sum_{q\in P}|\text{KDE}_P(p) - \text{KDE}_Q(p)|^p\Big)^{1/p}. \tag{3.4}$$

These average over the domain or P; hence if $|\text{KDE}_P(x) - \text{KDE}_Q(x)| \leq \varepsilon$ *for all* x, then $L_p(P,Q)$ is also at most ε. That "for all" bound is precisely what is guaranteed by $L_\infty(P,Q)$, hence it is a stronger bound.

Another reason to study L_∞ error is that it preserves the worst case error. This is particularly important when $\text{KDE}_P(x)$ values above a threshold trigger an alarm. For instance in tracking densities of tweets, too much activity in one location may indicate some event worth investigating. L_1 or L_2 errors from a baseline may be small, but still have high error in one location either triggering a false alarm, or missing a real event.

3.1.4 Computing Coreset Q of P

In this chapter, we define the coreset as a point set Q the same as 1.9, such that for any query point x, evaluating $\text{KDE}_Q(x)$ is much faster than evaluating $\text{KDE}_P(x)$ and $\text{KDE}_Q(x)$ can approximate $\text{KDE}_P(x)$ very well based on L_2 [31] and L_∞ [149] [103].

In Chapter 2, we summarized the approaches to finding such a coreset in one and two dimensions. For a point set P with n points, in one dimension, random sampling method can give us a size $O((1/\varepsilon^2)\log(1/\delta)$ coreset with construction cost $O(n)$; sort-selection method can construct a coreset $|Q| = O(\varepsilon)$ with the cost is $O(n\log\frac{1}{\varepsilon})$ if using ε-approximation quantiles, or $O(n\log n)$ otherwise.

In two dimension, random sampling gives the same bound on the sample size and the cost as in one dimension; kernel herding takes $O(|Q|n)$ time to construct Q and claims they need $O(1/\varepsilon)$ steps; min-cost matching with merge reduce framework and random sampling as preprocessing step can be constructed in $O(n + \frac{1}{\varepsilon^4}\log^3\frac{1}{\varepsilon})$ time and $|Q| = O(\frac{1}{\varepsilon}\log^{1.5}\frac{1}{\varepsilon})$; both Grid matching and Z-order selection methods with merge reduce framework give us a coreset with $|Q| = O(\frac{1}{\varepsilon}\log^{2.5}\frac{1}{\varepsilon})$ in $O(n + \frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$ time by using random sampling as the preprocessing step.

In high dimensions, Phillips [103] states that one can construct Q in $O(n/\varepsilon^2)$ time of size $O((1/\varepsilon)^{2d/(d+2)}\log^{d/(d+2)}(1/\varepsilon))$ with bounded L_∞ error.

3.1.5 Related Work on Bandwidth Selection

There is a vast literature on bandwidth selection under the L_1 [40, 41] or L_2 [122, 126] metric. In these settings, Q is drawn, often at random, from a unknown continuous distribution μ (but μ can be evaluated at any single point x). Then the goal is to choose ω to minimize $\|\mu - \text{KDE}_{Q,\omega}\|_{\{1,2\}}$. This can be conceptualized in two steps as $\|\mu - \text{KDE}_{\mu,\omega}\|$ and $\|\text{KDE}_{\mu,\omega} - \text{KDE}_{Q,\omega}\|$. This first step is minimized as $\omega \to 0$ and the second step is minimized as $\omega \to \infty$. Together, there is a value $\omega_{\{1,2\}} \in (0, \infty)$ that minimizes the overall objective.

The most common error measure for ω under L_2 are Integrated Squared Errors (ISE) $ISE(\omega) = \int_{x\in\mathbb{R}^d}(\text{KDE}_{Q,\omega} - \mu)^2 dx$ and its expected value, the Mean Integrated Squared Error (MISE) $MISE(\omega) = E_{Q\sim\mu}[\int_{x\in\mathbb{R}^d}(\text{KDE}_{Q,\omega} - \mu)^2 dx]$. As MISE is not mathematically tractable, often approximations such as the Asymptotic Mean Squared Error (AMISE) or others [126, 130] are used. Cross-validation techniques [15, 45, 62, 117, 118, 123] are used to evaluate various parameters in these approximations. Alternatively, plug-in methods [76, 121, 141] recursively build approximations to μ using KDE_{Q,ω_i}, and then refine the estimate of ω_{i+1} using KDE_{Q,ω_i}. A series of Bayesian approaches [16,38,50,73,82,148] build on these models and select ω using MCMC approaches.

An alternative to these L_2 approaches is using an L_1 measure, like integrated absolute error (IAE) of $\text{KDE}_{Q,\omega}$ is $IAE(\omega) = \int_{x\in\mathbb{R}^d} |\text{KDE}_{Q,\omega} - \mu| dx$, which has simple interpretation of being the area between the two functions. Devroye and Györfi [40] describe several robustness advantages (better tail behavior, transformation invariance) to these approaches. Several of the approximation approaches from L_2 can be extended to L_1 [64].

Perplexingly, however, the bandwidths generated by these methods can vary quite drastically! In this chapter, we assume that some bandwidth is given to indicate the intended scale, and then we choose a bandwidth for a sparser point set. Hence the methods surveyed above are not directly comparable to our proposed approaches. But we still include the experiment results from some of the above methods to show that different approaches give quite different "optimal" bandwidth, which in another way shows us there are more than one correct bandwidth for some data set.

3.2 Computing $err(P, Q)$ Error

The goal of this section is to calculate

$$err(P, Q) = \max_{x \in \mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_Q(x)|. \tag{3.5}$$

For notational convenience

$$G(x) = |\text{KDE}_P(x) - \text{KDE}_Q(x)|. \tag{3.6}$$

We focus on the case where K is a unit Gaussian. Since even calculating $\max_{x \in \mathbb{R}^d} \text{KDE}_P(x)$ (which is a special case of $err(P, Q)$ where Q is empty) appears hard, and only constant factor approximations are known [1, 106], we will not calculate $err(P, Q)$ exactly. Unfortunately, these approximation techniques [1, 106] for $\max_{x \in \mathbb{R}^d} \text{KDE}_P(x)$ do not easily extend to estimating $err(P, Q)$. They can focus on dense areas of P, since the maximum must occur there, but in $err(P, Q)$, these dense areas may perfectly cancel out. These approaches are also much more involved than the strategies we will explore.

3.2.1 Approximation Strategy

Towards estimating $err(P, Q)$, which is optimized over all of $\mathbb{R}^d$, our strategy is to generate a finite set $X \subset \mathbb{R}^d$, and then return $err_X(P, Q) = \max_{x \in X} G(x)$. Our goal in the generation of X is so that in practice, our returned estimate $err_X(P, Q)$ is close to $err(P, Q)$, but also so that under this process as $|X| \to \infty$ then formally $err_X(P, Q) \to err(P, Q)$. We say such a process *converges*.

We formalize this in two steps. First we show that $G(x)$ is Lipschitz-continuous, hence a point $\hat{x} \in \mathbb{R}^d$ close to the point $x^* = \arg\max_{x \in \mathbb{R}^d} G(x)$ will also have error value close to x^*. Then given this fact, we will need to show that our strategy will, for any radius r, as $|X| \to \infty$ generate a point $\hat{x} \in X$ so that $\|x^* - \hat{x}\| \le r$. This will be aided by the following structural theorem on the location of x^*. (M is illustrated in Figure 3.2.)

Theorem 5. *For K_σ a unit Gaussian kernel, and two point sets $P, Q \in \mathbb{R}^d$, M is the Minkowski sum of a ball of radius σ and the convex hull of $P \cup Q$, then $x^* = \arg\max_{x \in \mathbb{R}^d} G(x)$ must be in M.*

We want to prove Theorem 5 in one and two dimensions. For simplicity, we assume $Q \subset P$ so $P = P \cup Q$.

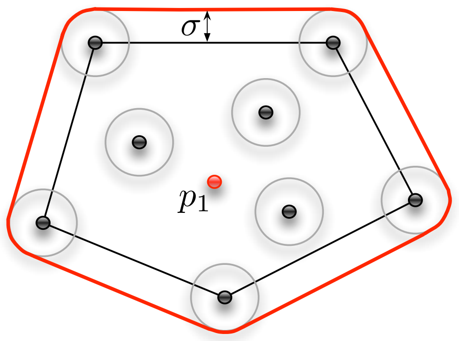


Figure 3.2. Illustration of the Minkowski sum of a ball of radius σ and convex hull of $P \cup Q$.

First, we work on the weighted one-dimensional data, and extend to two dimensions using that the cross section of a two-dimensional Gaussian is still a one-dimensional Gaussian. We focus on when P and Q use the same bandwidth σ, and a unit kernel K_σ. We start to examine two points in one dimension, and without loss of generality, we assume $p_1 = d$ and $p_2 = -d$ for $d \geq 0$, and that the coreset of P is $Q = \{p_2\}$. We assign the weight for p_1 as w_1 and the weight for p_2 as w_2. Plug in P, Q, and the weight for each point, $G(x) = |\text{KDE}_P(x) - \text{KDE}_Q(x)|$ is expanded as following:

$$G(x) = \left| \frac{1}{2} w_1 \exp\big(-\frac{(x-d)^2}{2\sigma^2}\big) - \frac{1}{2} w_2 \exp\big(-\frac{(x+d)^2}{2\sigma^2}\big) \right|.$$

We assume $w_1 \geq w_2$, the largest error point must be closer to p_1. So we only need to discuss when $x \geq 0$, then $\frac{1}{2} w_1 \exp\big(-\frac{(x-d)^2}{2\sigma^2}\big) \geq \frac{1}{2} w_2 \exp\big(-\frac{(x+d)^2}{2\sigma^2}\big)$, thus

$$G(x) = \frac{1}{2} w_1 \exp\big(-\frac{(x-d)^2}{2\sigma^2}\big) - \frac{1}{2} w_2 \exp\big(-\frac{(x+d)^2}{2\sigma^2}\big).$$

Lemma 5. *For K_σ a unit Gaussian kernel, $P = \{p_1, p_2\}$ and $Q = \{p_2\}$ where $p_1 = d$ and $p_2 = -d$, when $x \geq 0$, function $G(x)$ has only one local maximum, which is between d and $d + \sigma$ and $G(x)$ is decreasing when $x > d$.*

Proof. By taking the derivative of $G(x)$, we can get

$$\begin{aligned} \frac{\mathrm{d}G(x)}{\mathrm{d}x} = & \frac{1}{2} w_2 \exp\big(-\frac{(x+d)^2}{2\sigma^2}\big) \frac{x+d}{\sigma^2} \\ & - \frac{1}{2} w_1 \exp\big(-\frac{(x-d)^2}{2\sigma^2}\big) \frac{x-d}{\sigma^2}. \end{aligned}$$

When $0 \leq x < d$, both $\frac{1}{2} w_2 \exp\big(-\frac{(x+d)^2}{2\sigma^2}\big) \frac{x+d}{\sigma^2}$ and $-\frac{1}{2} w_1 \exp\big(-\frac{(x-d)^2}{2\sigma^2}\big) \frac{x-d}{\sigma^2} > 0$, thus $\frac{\mathrm{d}G(x)}{\mathrm{d}x} > 0$, so $G(x)$ is always increasing.

When $x = d$,

$$\frac{\mathrm{d}G(x)}{\mathrm{d}x} = \frac{1}{2} w_2 \exp\big(-\frac{2d^2}{\sigma^2}\big)\frac{2d}{\sigma^2} \geq 0.$$

To understand $x > d$ we examine the ratio function

$$r(x) = \frac{\frac{1}{2} w_2 \exp\big(-\frac{(x+d)^2}{2\sigma^2}\big)\frac{x+d}{\sigma^2}}{\frac{1}{2} w_1 \exp\big(-\frac{(x-d)^2}{2\sigma^2}\big)\frac{x-d}{\sigma^2}} = \frac{w_2}{w_1}\exp\big(-\frac{2xd}{\sigma^2}\big)\frac{x+d}{x-d}.$$

Since both $\exp\big(-\frac{2xd}{\sigma^2}\big)$ and $\frac{x+d}{x-d}$ are decreasing and positive, $r(x)$ and thus $\frac{\mathrm{d}G(x)}{\mathrm{d}x}$ is decreasing when $x > d$.

When $x = d + \sigma$, the ratio function is

$$r(d+\sigma) = \frac{w_2}{w_1}\exp(-\frac{2\sigma d + 2d^2}{\sigma^2})\frac{\sigma + 2d}{\sigma}.$$

We can view the above equation as a function of variable d.

$$r(d) = \frac{w_2}{w_1}\exp(-\frac{2\sigma d + 2d^2}{\sigma^2})\frac{\sigma + 2d}{\sigma},$$

and take the derivative of $r(d)$:

$$\frac{\mathrm{d}r(d)}{\mathrm{d}d} = -\frac{4d(d+\sigma)}{\sigma^3}\frac{w_2}{w_1}\exp(-\frac{2\sigma d + 2d^2}{\sigma^2}) \leq 0.$$

With $d \geq 0$ then $\frac{\mathrm{d}r(d)}{\mathrm{d}d} \leq 0$, and thus $r(d)$ is a decreasing function which attains maximum $\frac{w_2}{w_1} \leq 1$ when $d = 0$; thus $r(d) \leq 1$. So when $x = d + \sigma$, $\frac{\mathrm{d}G(x)}{\mathrm{d}x} \leq 0$. Due to the fact that $\frac{\mathrm{d}G(x)}{\mathrm{d}x} \geq 0$ when $x = d$ and $\frac{\mathrm{d}G(x)}{\mathrm{d}x}$ is decreasing when $x > d$, there is only one point between d and $d + \sigma$ making $\frac{\mathrm{d}G(x)}{\mathrm{d}x} = 0$. When $0 \leq x < d$, then $\frac{\mathrm{d}G(x)}{\mathrm{d}x} > 0$. There is only one maximum point of $G(x)$ between d and $d + \sigma$ when $x \geq 0$. □

From Lemma 5, we show that the evaluation point having largest error is between d and $d + \sigma$. Due to the symmetry of p_1 and p_2, when $w_1 \leq w_2$, $G(x)$ gets its largest error between $-d$ and $-d - \sigma$.

With the results on both sides, we now show the maximum value point of $G(x)$ can't be outside σ distance of $Conv(P)$.

Now we discuss the case for n points in one dimension.

Lemma 6. *For K_σ a unit Gaussian kernel, P has n points and $|Q| = |P|/2$, $\arg\max_{x\in\mathbb{R}^1} G(x)$ for one-dimensional data is within σ distance of $Conv(P)$.*

Proof. Suppose $n = 2k$, $P = \{p_1, p_2, p_3, p_4, ..., p_{2k-1}, p_{2k}\}$, choose any k points in Q. Then pair any point in Q with any point in P not in Q, so each point in P is in exactly one pair. For simplicity we set $Q = \{p_1, p_3, ..., p_{2k-1}\}$ and the pairs are $\{p_1, p_2\}, \{p_3, p_4\}, ..., \{p_{2k-1}, p_{2k}\}$.

Suppose $e_1 = \arg\max_{x \in \mathbb{R}^1} G(x)$ is not within σ distance of $Conv(P)$ and p_1 is the point closest to e_1. Based on Lemma 5, for P has only two points, function $G(x)$ is decreasing as a point outside σ moves away from p_1. So if we choose another point e_2 infinitesimally closer to p_1, and we set $P_1 = \{p_1, p_2\}$, $Q_1 = \{p_1\}$, $G_{P_1,Q_1}(e_2)$ has larger value than $G_{P_1,Q_1}(e_1)$. Since p_1 is the closest point in P, for any other set $P_2 = \{p_3, p_4\}$, $Q_2 = \{p_3\}$, e_2 is closer to P_2 than e_1 is to P_2, hence $G_{P_2,Q_2}(e_2)$ is also larger than $G_{P_2,Q_2}(e_1)$. The same result holds for all pairs $\{p_{2i-1}, p_{2i}\}$, where i is from 1 to k. So $G(e_2) > G(e_1)$, which contradicts the assumption that $e_1 = \arg\max_{x \in \mathbb{R}^1} G(x)$. So the largest error evaluation point should be within σ distance of $Conv(P)$. $\square$

In two dimensions we show a similar result. We illustrate the Minkowski sum M of a set of points P with a ball of radius σ in Figure 3.2.

Theorem 6. *For K_σ a unit Gaussian kernel, and two point sets $P, Q \in \mathbb{R}^2$, $|Q| = |P|/2$, $\arg\max_{x \in \mathbb{R}^2} G(x)$ should be within the Minkowski sum M of a ball of radius σ and $Conv(P)$.*

Proof. Suppose the largest error position $e_1 = \arg\max_{x \in \mathbb{R}^2} G(x) \notin M$, then for some direction v, no point in the convex hull of P is closer than σ to e_1 after both are projected onto v. Then since any cross section of a Gaussian is a one-dimensional Gaussian (with reduced weight), we can now invoke the one-dimensional result in Lemma 6 to show that e_1 is not the largest error position along the direction v, thus $e_1 \neq \arg\max_{x \in \mathbb{R}^2} G(x)$. So $\arg\max_{x \in \mathbb{R}^2} G(x)$ should be within the Minkowski sum M of a ball of radius σ and $Conv(P)$. $\square$

We will not focus on proving theoretical bounds on the rate of convergence of these processes since they are quite data dependent, but will thoroughly empirically explore this rate in Section 3.4. As $|X|$ grows, the max error value in X will consistently approach some error value (the same value for several provably converging approaches), and we can then have some confidence that as these processes plateau, they have successfully estimated $err(P, Q)$. Our best process WCen6 converges quickly (e.g., $|X| = 100$); it is likely that the maximum error is approximately achieved in many locations.

Now, as a basis for formalizing these results, we first show $G(x)$ is Lipschitz continuous. Recall a function $f : \mathbb{R}^d \to \mathbb{R}$ is Lipschitz continuous if there exists some constant β such that for *any* two points $x, y \in \mathbb{R}^d$ that $|f(x) - f(y)|/\|x - y\| \leq \beta$. This result follows from the Gaussian kernel (as well as all other kernels except the Ball kernel) also being Lipschitz continuous. Then, since the function $f(x) = \text{KDE}_P(x) - \text{KDE}_Q(x)$ is a finite weighted sum of Gaussian kernels, each of which is Lipschitz continuous, so is $f(x)$. Since taking absolute value does affect Lipschitz continuity, the claim holds.

3.2.2 Generation of Evaluation Points

We now consider strategies to generate a set of points X so that $err_X(P, Q)$ is close to $err(P, Q)$. Recall that M, the Minkowski sum of a ball of radius σ with the convex hull of $P \cup Q$, must contain the point x^* which results in $err(P, Q)$. In practice, it is typically easier to use $\mathcal{B}$, the smallest axis-aligned bounding box that contains M. And for discussion, we assume $Q \subset P$ so $P = P \cup Q$.

- **Rand:** *Choose each point at random from $\mathcal{B}$.*

Since $x^* \in M \subset \mathcal{B}$, eventually some point $x \in X$ will be close enough to x^*, and this process converges.

- **Orgp:** *Choose points randomly from P.*

This process does not converge since the maximum error point may not be in P. However, section 3.4 shows that this process converges to its limit very quickly. So many of the following proposed approaches will attempt to adapt this approach while still converging.

- **Orgp+N:** *Choose points randomly from the original point set P then add Gaussian noise with bandwidth σ, where σ is the bandwidth of K.*

Since the Gaussian has infinite support, points in X can be anywhere in $\mathbb{R}^d$, and will eventually become close enough to x^*. So this process converges.

- **Grid:** *Place a uniform grid on $\mathcal{B}$ (we assume each grid cell is a square) and choose one point in each grid.* For example, in two dimension, if four evaluation points are needed, the grid would be 2×2 and if nine points are needed, it would be 3×3. So with this method, the number of evaluation points is a non-prime integer.

Since $x^* \in \mathcal{B}$, and eventually the grid cell radius is arbitrarily small, then some point $x \in X$ is close enough to x^*. Thus, this process converges.

- **Cen{E[m]}:** *Randomly select one point p_1 from the original point set P and randomly choose m neighbor points of p_1 within the distance of 3σ. m is chosen through a Exponential process with rate $1/E[m]$. Then we use the centroid of the selected neighbor points as the evaluation point.* This method is inspired by [47], which demonstrates interesting maximums of KDEs at the centroids of the data points.

Since P is fixed, the centroid of any combination of points in P is also finite, and the set of these centroids may not include x^*. So, this process does not converge. We next modify it in a way so it does converge.

- **WCen{E[m]}:** *Randomly select one point p_1 from the original point set P and select the neighbor point $p_n \in P$ as candidate neighbor proportional to $\exp(-\frac{||p_n - p_1||^2}{2\sigma^2})$, where σ is the bandwidth for K. The smaller the distance between p_n and p_1, the higher probability it will be the chosen. Repeat to choose m total points including p_1, where again, m is from an Exponential process with rate $1/E[m]$. Now refine the m neighbor points so with probability 0.9, it remains the original point $p_n \in P$, with the remaining probability it is chosen randomly from a ball of radius σ centered at p_n. Next, we assign each point a random weight in $[0, 1]$ so that all weights add to 1. Then finally, the evaluation point is the weighted centroid of these points.*

This method retains much of the effectiveness of Cen, but does converge. Without the 0.1 probability rule of being in a ball of radius σ around each point, this method can generate any points within the convex hull of P. That 0.1 probability allows it to expand to M, the Minkowski sum of the convex hull of P with a ball of radius σ. And since $x^* \in M$, by Theorem 5 then this process converges.

- **Comb: Rand + Orgp:** *The combination of method Rand and Orgp, of which 20% points generated from $\mathcal{B}$ and 80% points generated from original points.*

The 20% of points from Rand guarantees convergence, but retain most empirical properties of Orgp. This was used before [149] with little discussion.

Section 3.4 describes extensive experiments on both synthetic and real data to evaluate these methods. The weighted centroid method WCen{E[m]} with large parameter (e.g., $E[m] = 6$) works very well for one and two dimensions, and also converges.

3.3 Bandwidth Selection

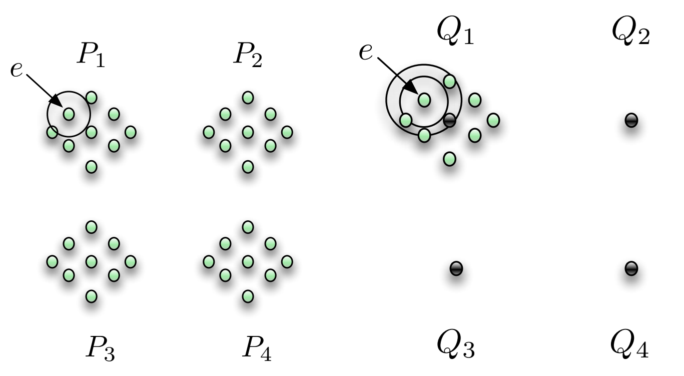
In this section, we consider being given two point sets $P, Q \subset \mathbb{R}^2$, a kernel K, and a bandwidth σ associated with P. We consider K as a normalized Gaussian kernel, and the case where Q is a coreset of P. The goal is to find another bandwidth ω to associate with Q so that $err(P, \sigma, Q, \omega)$ is small.

3.3.1 Refining the Bandwidth for Coresets

These coresets are constructed assuming that KDE_Q uses the same bandwidth σ as KDE_P. But can we improve this relationship by using a different bandwidth ω for Q? The theory for L_1 and L_2 error (assuming Q is a random sample from P) dictates that as $|Q|$ decreases, the bandwidth ω should increase. This intuition holds under any error measure since with fewer data points, the KDE should have less resolution. It also matches the L_∞ theoretical error bounds described above.

We first reinforce this with a simple two-dimensional example. Consider point set $P = \cup\{P_1, P_2, P_3, P_4\}$ in Figure 3.3(a), the radius of the circle represents the bandwidth σ for P. Figure 3.3(b) gives the coreset Q of P: $Q = \cup\{Q_1, Q_2, Q_3, Q_4\}$, each Q_i contains only one black point. Now suppose our evaluation point is point e. If we use the original bandwidth σ, $\text{KDE}_{Q,\sigma}(e) = 0$ with ball kernel, but if we use ω, which is the radius of a larger circle centered at e, then $\text{KDE}_{Q,\omega}(e) > 0$, so the error is decreased. But, we do not want ω too large, as it would reach the points in other Q_i, which is not the case for σ in P, so the error would be increased again. Thus, there seems to be a good choice for $\omega > \sigma$.

But the situation of finding the ω_{opt} that minimizes $h(\omega) = err(P, \sigma, Q, \omega)$ is more



(a) Original data set. (b) Coreset.

Figure 3.3. Example with need for larger ω for coreset Q.

complicated. For each ω it is itself a maximization over $x \in \mathbb{R}^d$. There may in fact be more than one local minimum for ω in $h(\omega)$.

However, equipped with the WCen6 procedure to evaluate $err(P, Q)$, we propose a relatively simple optimization algorithm. We can perform a golden section search over ω, using WCen6 to obtain a set X and evaluate $err_X(P, \sigma, Q, \omega)$. Such a search procedure requires a convex function for any sort of guarantees, and this property may not hold. However, we show next that $h(\omega)$ has some restricted Lipschitz property, so that with random restarts it should be able to find a reasonable local minimum. This is illustrated in Figure 3.4, where the curve that is Lipschitz either has a large, relatively convex region around the global minimum, or has shallow local minimums. The other curve without a Lipschitz property has a very small convex region around the global maximum, and any search procedure will have a hard time finding it.

3.3.2 Lipschitz Properties of h

In general, however, $h(\omega)$ is not Lipschitz in ω. But, we can show it is Lipschitz over a restricted domain, specifically when $\omega > \sigma$ and when $1/\sigma \leq A$ for some absolute constant A. Define $y(\omega, a) = \frac{1}{2\pi\omega^2} \exp(-a^2/(2\omega^2))$.

Lemma 7. *For any $\omega \geq \sigma$ and $1/\sigma \leq A$, $y(\omega, a)$ is β-Lipschitz with respect to ω, with $\beta = |a^2 - 1/\pi| A^3$.*

Proof. By taking the first derivative of $y(\omega)$, we have

$$\frac{\mathrm{d}y(\omega, a)}{\mathrm{d}\omega} = (a^2 - \frac{1}{\pi})\omega^{-3} \exp(-a^2/(2\omega^2)).$$

And thus

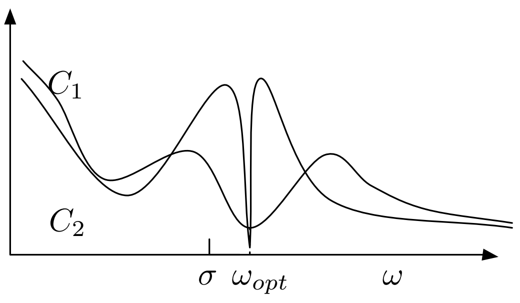


Figure 3.4. Two curves, one of which is Lipschitz.

$$\begin{aligned}\left|\frac{\mathrm{d}y(\omega,a)}{\mathrm{d}\omega}\right| &= |a^2 - 1/\pi|\omega^{-3}\exp(-a^2/(2\omega^2)) \\ &\leq |a^2 - 1/\pi|\sigma^{-3} \leq |a^2 - 1/\pi|A^3.\end{aligned}$$

So, the absolute value of largest slope of function $y(\omega, a)$ is $\beta = |a^2 - 1/\pi|A^3$, thus $y(\omega, a)$ is β-Lipschitz continuous on ω. □

Theorem 7. *For any $\omega \geq \sigma$ and $1/\sigma \leq A$, $h(\omega)$ is β-Lipschitz with respect to ω, for $\beta = \frac{1}{|Q|}\sum_{q\in Q}|(x^* - q)^2 - 1/\pi|A^3$ where $x^* = \arg\max_{x\in\mathbb{R}^2}|\text{KDE}_{P,\sigma}(x) - \text{KDE}_{Q,\omega}(x)|$.*

Proof. If $\text{KDE}_{P,\sigma}(x^*) \geq \text{KDE}_{Q,\omega}(x^*)$ then

$$\begin{aligned}h(\omega) &= |\text{KDE}_{P,\sigma}(x^*) - \text{KDE}_{Q,\omega}(x^*)| \\ &= \text{KDE}_{P,\sigma}(x^*) - \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{2\pi\omega^2}\exp\left(\frac{-(x^*-q)^2}{2\omega^2}\right) \\ &= \text{KDE}_{P,\sigma}(x^*) - \frac{1}{|Q|}\sum_{q\in Q} y(\omega, (x^* - q))).\end{aligned}$$

Since $h(\omega)$ is a linear combination of $|Q|$ functions of $y(\omega, a)$ plus a constant and $y(\omega, a)$ is Lipschitz continuous, based on the Lemma 7 $h(\omega)$ is Lipschitz continuous on ω. We can get the same result if $\text{KDE}_{P,\sigma}(x^*) \leq \text{KDE}_{Q,\omega}(x^*)$. In both directions, the first derivative of the function is bounded, so $h(\omega)$ is bounded. □

3.3.3 Random Golden Section Search

From the above properties, we design a search procedure that will be effective in finding the bandwidth ω minimizing $err(P, \sigma, Q, \omega)$. The *random golden section search* is based on the golden section search [79], a technique to find extremum in a strictly unimodal function. To find a minimum, it successively narrows a range $[\ell, r]$ with known function values $h(\ell), h(m_1), h(m_2)$, and $h(r)$ with $\ell < m_1 < m_2 < r$ and with both $h(m_1), h(m_2)$ less than $h(\ell)$ and $h(r)$. If $h(m_1) < h(m_2)$ the new search range is $[\ell, m_2]$, otherwise, it is $[m_1, r]$. In either case, a new fourth point is chosen according to the golden ratio in such a way that the interval shrinks by a constant factor on each step.

However, $h(\omega)$ in our case can be a multimodal function, thus golden section search is not guaranteed to work. We apply random restarts as follows. Starting with a range $[\ell = \sigma, r = 10\sigma]$, we choose one middle point at $m = \lambda\sigma$ for $\lambda \sim \mathsf{Unif}(1, 10)$. If $h(m) > h(r)$, we increase r by a factor 10 until it is (e.g., $r = 100\sigma$). Then the second middle point is chosen using the golden ratio, and the search is run deterministically. We repeat with several random values λ.

3.4 Experiments

Here we run an extensive set of experiments to validate our techniques. We compare KDE_P with kernel density estimate under smaller coreset KDE_Q for both synthetic and real data in one and two dimensions. To show our methods work well in large data sets, we use the large synthetic data set (0.5 million) and real data set (1 million) in two dimension.

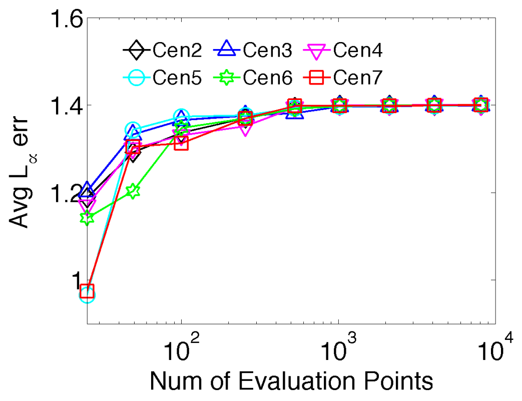
3.4.1 Data Sets

We consider data sets that have different features at various scales, so that as more data are present using a small bandwidth more fine-grain features are brought out, and a larger bandwidth only shows the coarse features. Our real data set in one dimension is the temperature data in Figure 3.1, with default $\sigma = 72$ (3 days). We use parameter $\varepsilon = 0.02$ to generate a coreset Q with the Sort-selection technique [149].

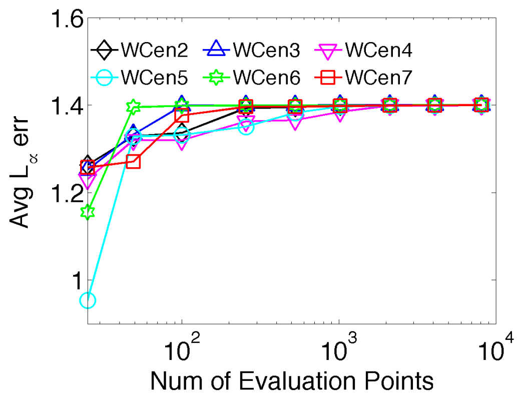
We can simulate such multiscale features more precisely. On a domain $[0, 1]$ we generate P recursively, starting with $p_1 = 0$ and $p_2 = 1$. Next we consider the interval between $[p_1, p_2]$ and insert two points at $p_3 = 2/5$ and $p_4 = 3/5$. There are now 3 intervals $[p_1, p_3]$, $[p_3, p_4]$, and $[p_4, p_2]$. For each interval $[p_i, p_j]$ we recursively insert 2 new points at $p_i + (2/5) \cdot (p_j - p_i)$ and at $p_i + (3/5) \cdot (p_j - p_i)$, until $|P| = 19684$. The KDE of this data set with $\sigma = 0.01$ is shown in Figure 3.5(d), along with that of a coreset Q of size $|Q| = 100$.

We construct the two-dimensional synthetic data set in a similar way. The data are in $[0, 1]^2$ starting with four points $p_1 = (0, 0), p_2 = (0, 1), p_3 = (1, 0), p_4 = (1, 1)$. We recurse on this rectangle by adding 4 new points in the middle m: the x-coordinates are either at the 0.5-quantile or 0.8-quantile of the x-coordinates, and it is the same for new y-coordinates. These 4 new points create 9 smaller empty rectangles. We further recurse on each of these rectangles until $|P| = 532900$. The KDE_P with $\sigma = 0.01$ is shown in Figure 3.6(a). We use Grid matching [149] to generate a coreset Q with $\varepsilon = 0.1$ and size $|Q| = 1040$. Under the original bandwidth σ, the KDE_Q is shown in Figure 3.6(b); due to a small bandwidth this KDE has many more modes than the original, which motivates the larger bandwidth KDE shown in Figure 3.6(c).

For real data with multiple scales in two dimension we consider OpenStreetMap data

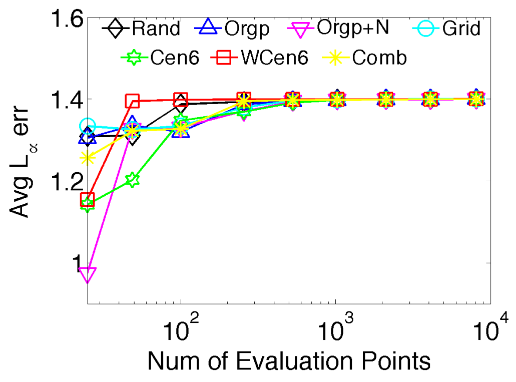


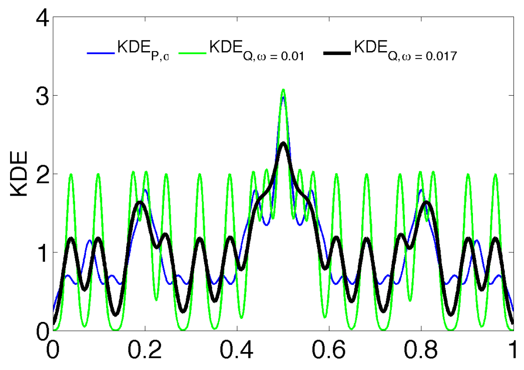
(a) Centroid method.



(b) Weighted centroid method.

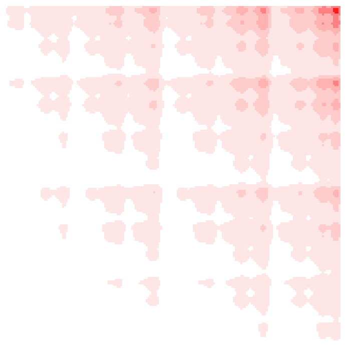
(c) Compare with all the other methods.



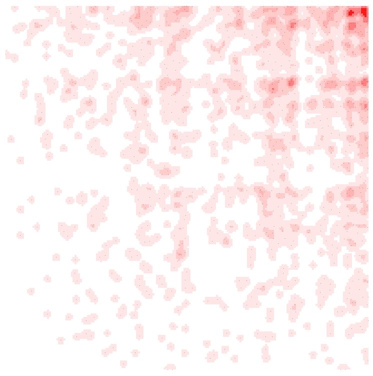


(d) $\text{KDE}_{P,\sigma}$ and $\text{KDE}_{Q,\omega}$ with original and best bandwidth.

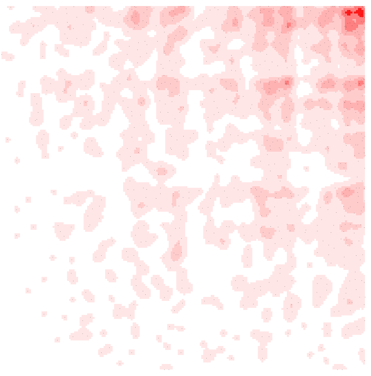
Figure 3.5. Results on one-dimensional synthetic data to choose best evaluating point generation techniques.



(a) $\text{KDE}_{P,\sigma=0.01}$.



(b) $\text{KDE}_{Q,\omega=0.01}$.



(c) $\text{KDE}_{Q,\omega=0.013}$.

Figure 3.6. Visualization of KDE_P and KDE_Q for two-dimensional synthetic data using different bandwidth.

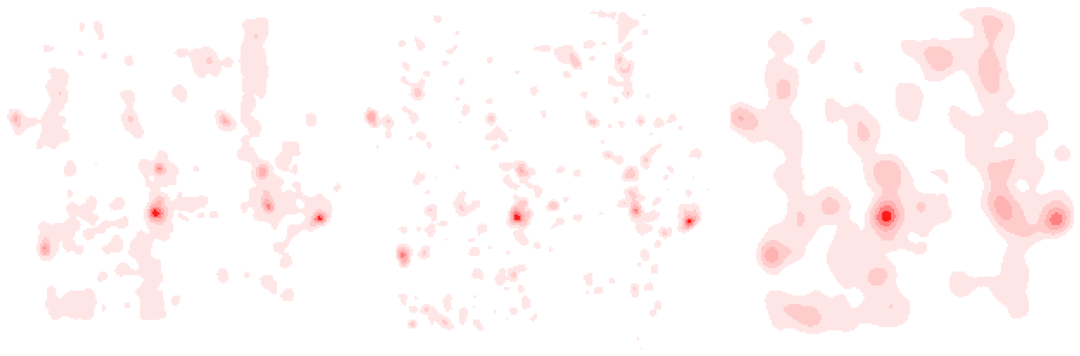
from the state of Iowa. Specifically, we use the longitude and latitude of all highway data points, then rescale so it lies in $[0,1] \times [0,1]$. It was recognized in the early 1900s [133] that agricultural populations, such as Iowa, exhibited population densities at several scales. For ease of experiments, we use the original data of size $|P| = 1155102$ with $\sigma = 0.01$, and Q as a smaller coreset with $\varepsilon = 0.1$ and $|Q| = 1128$. These are illustrated in Figure 3.7.

3.4.2 Evaluating Point Generation for err_X

To find the best evaluation point generation techniques, we compare the various ways to generate a set X to evaluate $err_X(P,Q)$. The larger numbers are better, so we want to find point sets X so that $err_X(P,Q)$ is maximized with $|X|$ small. As most of our methods are random, five evaluation point sets are generated for each method and the average $err_X(P,Q)$ is considered.

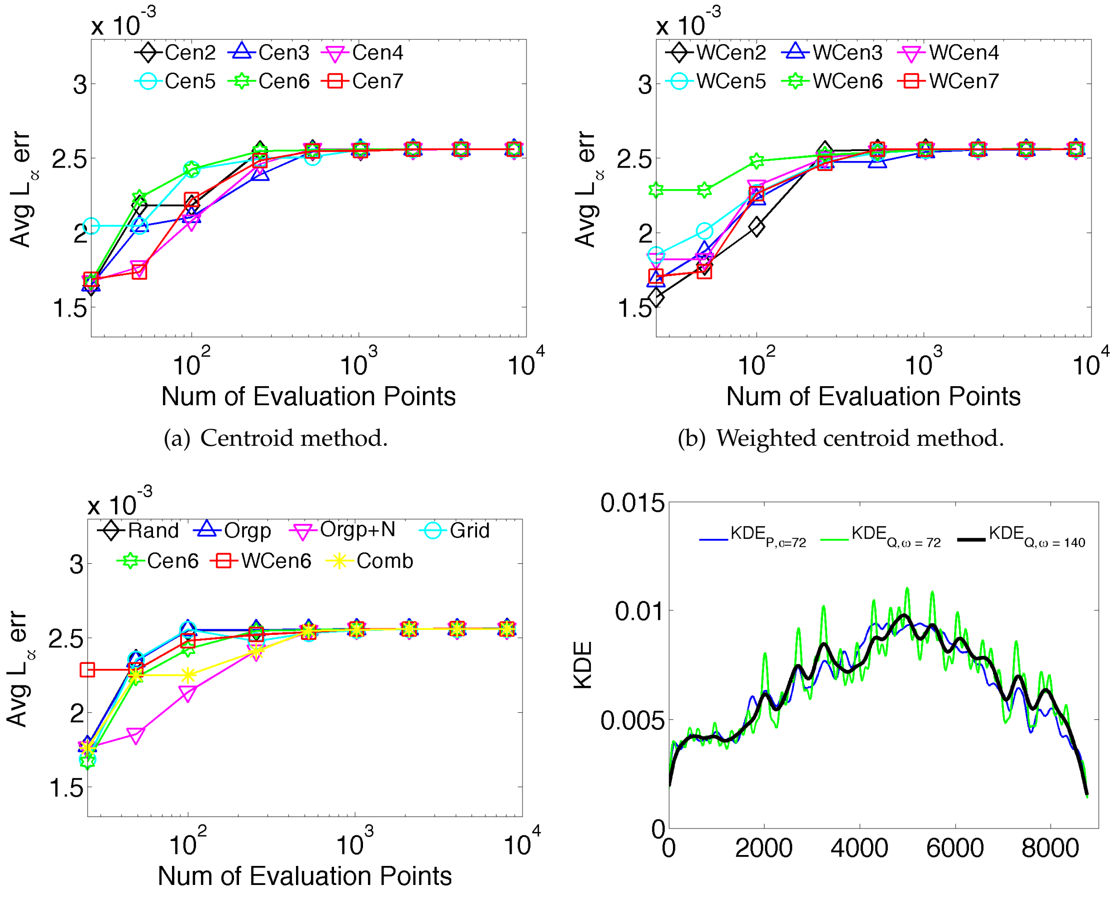
We start in one dimension, and investigate which parameter of the Cen and WCen methods work best. We will then compare the best in class against the remaining approaches. Recall the parameter $E[m]$ determines the expected number of points (under a Exponential process) chosen to take the centroid or weighted centroid of, respectively. We only show the test result with $E[m]$ from 2 to 7, since the results are similar when $E[m]$ is larger than 7, and the larger the parameter the slower (and less desirable) the process. The results are plotted in Figure 3.5 on the one-dimensional synthetic data. Specifically, Figure 3.5(a) shows the Cen method and Figure 3.5(b) the WCen method. Both methods plateau, for some parameter setting, after around $|X| = 100$, with WCen more robust to parameter choice. In particular, both WCen converges slightly faster but with not much pattern across the choice of parameter. We use Cen6 and WCen6 as representatives. We next compare these approaches directly against each other as well as Rand, Orgp, Orgp+N, Grid, and Comb in Figure 3.5(c). WCen6 appears the best in this experiment, but it has been selected as best WCen technique from random trials. The Rand and Grid techniques which also converge perform well, and are simpler to implement.

Similar results are seen on the real one-dimensional data in Figure 3.8. We can take best in class from Cen and WCen parameter choices, shown as Cen6 and WCen6 in Figure 3.8(a) and Figure 3.8(b). These perform well and similar to the simpler Rand, Grid, and Orgp in Figure 3.8(c). Since Rand and Grid also converge, in one dimension we would



(a) $\text{KDE}_{P,\sigma=0.01}$. (b) $\text{KDE}_{Q,\omega=0.01}$. (c) $\text{KDE}_{Q,\omega=0.032}$.

Figure 3.7. Visualization of KDE_P and KDE_Q for two-dimensional real data using different bandwidth.



(a) Centroid method. (b) Weighted centroid method.

(c) Compare with all the other methods. (d) $\text{KDE}_{P,\sigma}$ and $\text{KDE}_{Q,\omega}$ with original and best bandwidth.

Figure 3.8. Results on one-dimensional real data to choose the best evaluating point generation techniques.

recommend one of these simple methods.

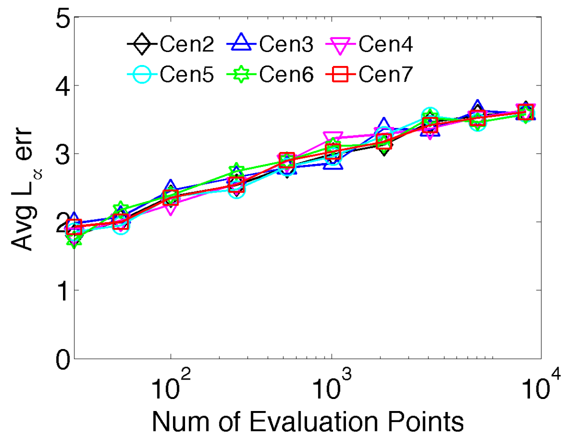
For two-dimensional data, the techniques perform a bit differently. We again start with the Cen and WCen methods as shown in Figure 3.9 on real and synthetic data. The convergence results are not as good as in one dimension, as expected, and it takes roughly $|X| = 1000$ points to converge. All methods perform roughly the same for various parameter settings, so we use Cen6 and WCen6 as representatives. Comparing against all techniques in Figure 3.9, most techniques perform roughly the same relative to each other, and again WCen6 appears to be a good choice to use. The notable exceptions is that Grid and Rand perform worse in two dimension than in one dimension; likely indicating that the data dependent approaches are more important in this setting.

3.4.3 Choosing New Bandwidth Evaluation

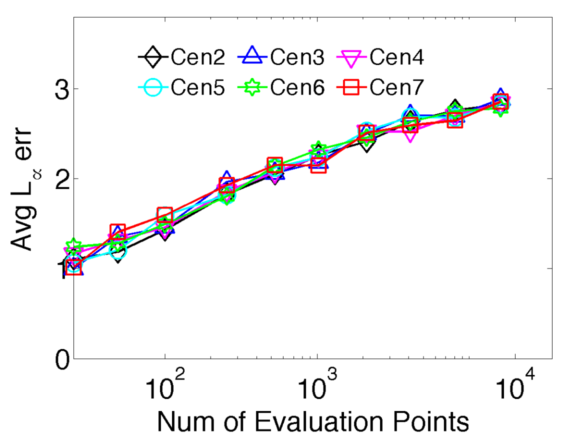
We now apply a random golden section search to find new bandwidth values for coresets on one-dimensional and two-dimensional synthetic and real data. In all random trials, we always find the same local minimum, and report this value. We will see that a value $\omega > \sigma$ can often give better error results, both visually and empirically, by smoothing out the noise from the smaller coresets.

Figures 3.10(a) and 3.10(b) show evaluation of $err_X(P, \sigma, Q, \omega)$ for various ω values chosen while running the random golden section search on synthetic and real one-dimensional data. In both cases, setting $\omega = \sigma$ (as $\omega = 0.01$ and $\omega = 72$, respectively) gives roughly twice as much error as using an omega roughly twice as large ($\omega = 0.017$ and $\omega = 142$, respectively).

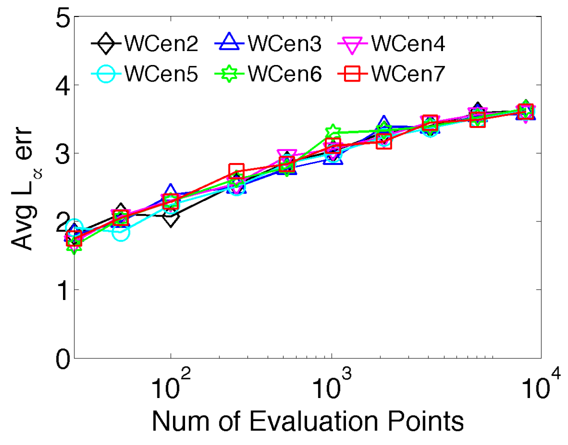
We can see even more dramatic results in the two-dimensional data in Figure 3.11. We observe in Figure 3.11(a) on synthetic data that with the original $\omega = \sigma = 0.01$ that the error is roughly 3.6, but by choosing $\omega = 0.013$ we can reduce the error to roughly 2.7. This is also shown visually in Figure 3.6 where a small coreset Q is chosen, and in Figure 3.6(b) the large-scale pattern in $\text{KDE}_{P,\sigma}$ is replaced by many isolated points; $\text{KDE}_{Q,\omega=0.013}$ increases the bandwidth and the desired visual pattern re-emerges. On real data, a similar pattern is seen in Figure 3.11(b). The original $\omega = \sigma = 0.01$ has error roughly 3.0, and an $\omega = 0.032$ (more than 3 times larger) gives error about 1.1. This extra smoothing is illustrated in Figure 3.7.



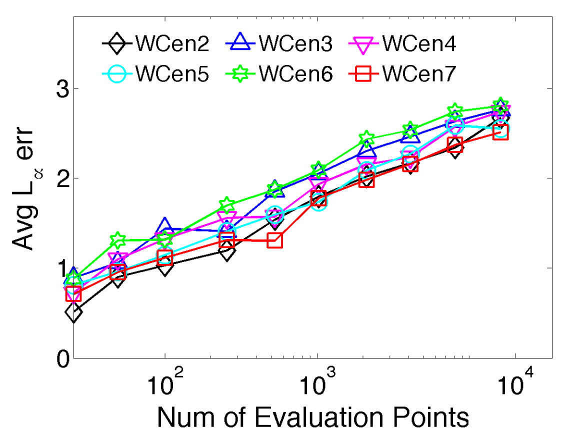
(a) Centroid methods.



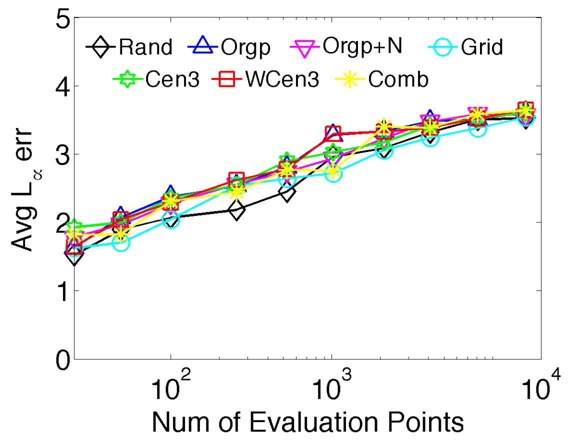
(b) Centroid method.



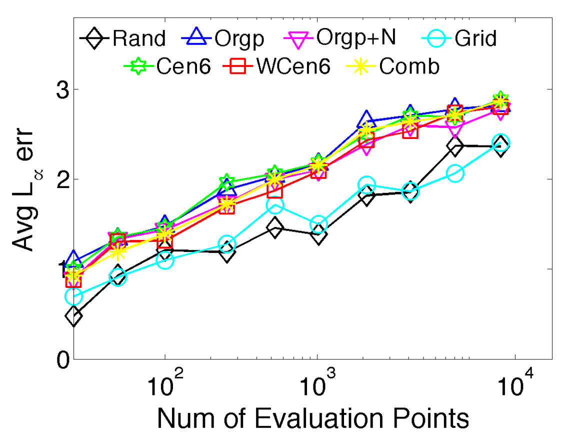
(c) Weighted centroid method.



(d) Weighted centroid method.



(e) All methods.



(f) All methods.

Figure 3.9. Choosing the best evaluation set X for on two-dimensional synthetic (left) and real (right) data.

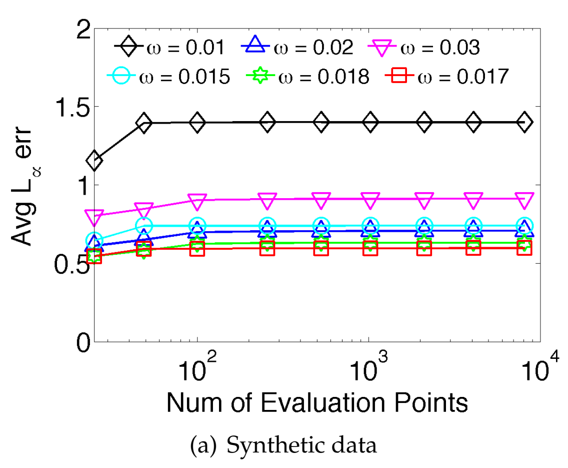


Figure 3.10. $\omega^* = \arg\min_\omega err_X(P, \sigma, Q, \omega)$ in $\mathbb{R}^1$.

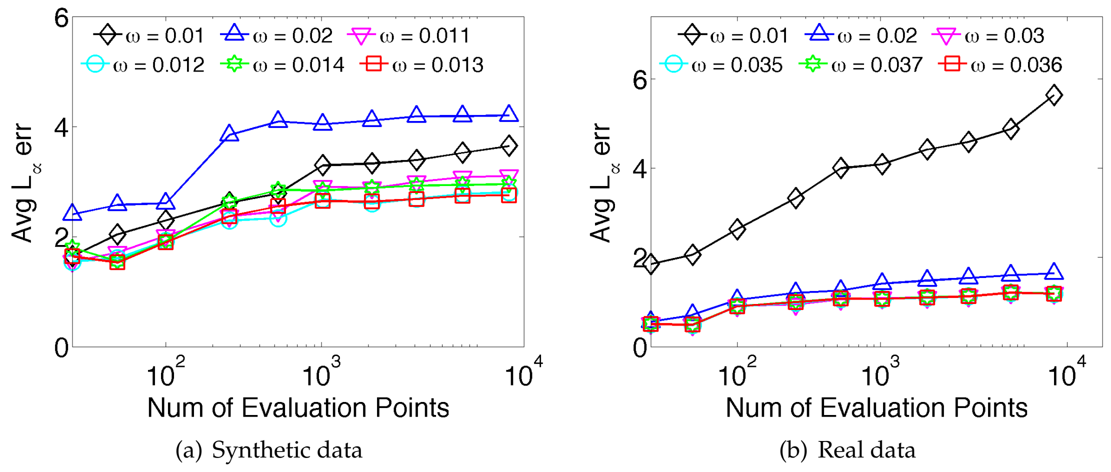


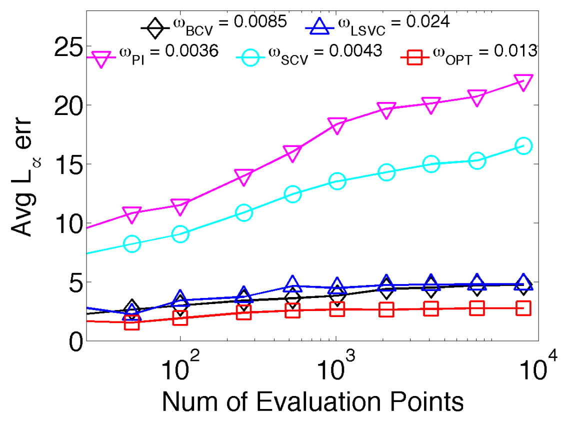
Figure 3.11. $\omega^* = \arg\min_\omega \exp_X(P, \sigma, Q, \omega)$ in $\mathbb{R}^2$.

Thus, we see that it is indeed useful to increase the bandwidth of kernel density estimates on a coreset, even though theoretical bounds already hold for using the same bandwidth. We show that doing so can decrease the error by a factor of 2 or more. Since we consider $\omega = \sigma$, and only decrease the error in the process, we can claim the same theoretical bounds for the new ω value. It is an open question of whether one can prove tighter coreset bounds by adapting the bandwidth value.

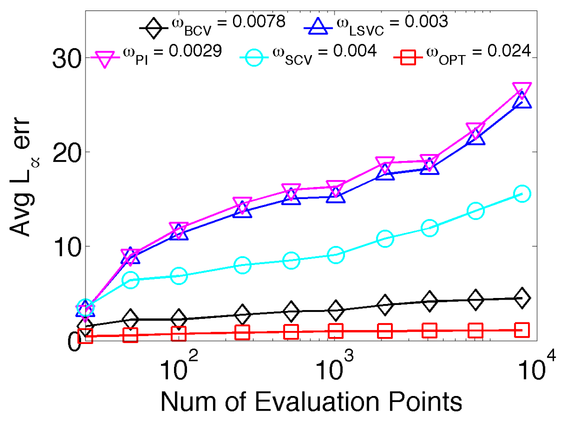
3.4.4 Compare with the Traditional Bandwidth Selection Method

In this section, we include some bandwidth selection results for the two-dimensional synthetic and real data using the traditional bandwidth selection method surveyed in Section 3.1.5, including biased cross-validation (BCV) bandwidth selection method, least-squares cross-validation (LSCV) bandwidth selection method, plug-in (PI) bandwidth selection method and smoothed cross-validation (SCV) bandwidth selection method.

We use the kernel smoothing R package(ks), which was originally introduced by Duong (2007) [44] and improved in 2014. In the experiment, our data set is normalized and we assume data in each dimension are independent and share the same bandwidth, so we use larger value from the main diagonal of bandwidth matrix computed from the R package. For the two-dimensional synthetic data set, we use the same coreset with $|Q| = 1040$, the bandwidth using the above four methods are $\omega_{BCV} = 0.0085, \omega_{LSCV} = 0.024, \omega_{PI} = 0.0036, \omega_{SCV} = 0.0043$. For the two-dimensional real data set, with the coreset $Q = 1128$, the bandwidth chosen from the four methods are $\omega_{BCV} = 0.0078, \omega_{LSCV} = 0.0003, \omega_{PI} = 0.0029, \omega_{SCV} = 0.004$. The corresponding error trends compared to our method for these two data sets are shown in Figure 3.12, where ω_{OPT} denotes the optimal bandwidth from our method. Both of these figures show our method can achieve the smallest error comparing to the baseline methods and all the baseline methods tend to give smaller bandwidth and not stable for different data sets.



(a) Synthetic data



(b) Real data

Figure 3.12. $\omega^* = \arg\min_\omega \exp_X(P, \sigma, Q, \omega)$ in $\mathbb{R}^2$.

3.5 Conclusion

This chapter considers evaluating kernel density estimates under L_∞ error, and how to use these criteria to select the bandwidth of a coreset. Since L_∞ error is stronger than the more traditional L_1 or L_2 errors, it provides approximation guarantees for *all* points in the domain, and aligns with recent theoretical results [103] of kernel range space, it is worth rigorously investigating.

We propose several methods to efficiently evaluate the L_∞ error between two kernel density estimates and provide a convergence guarantee. The method Grid works well, and is very simple to implement in $\mathbb{R}^1$. In $\mathbb{R}^2$, methods that adapt more to the data perform much better, and our technique WCen is shown to be accurate and efficient on real and synthetic data. We then use these technique to select a new bandwidth value for coresets that can improve the error by a factor of 2 to 3. We demonstrate this both visually and empirically on real and synthetic data sets.

CHAPTER 4

GENERALIZED KERNEL RANGE SPACE AND (ε, τ)-NET

This chapter considers traditional sample complexity problems but adapts to when the range space (or function space) smoothes out its boundary. This is important in various scenarios where either the data points or the measuring function are noisy. Similar problems have been considered in specific contexts of functions classes with a $[0, 1]$ range or kernel density estimates. We extend and generalize these results, motivated by scenarios such as the following.

(S1) Consider maintaining a random sample of noisy spatial data points (say twitter users with geo-coordinates), and we want this sample to include a witness to every large enough event. However, because the data coordinates are noisy, we use a kernel density estimate to represent the density. And moreover, we do not want to consider regions with a single or constant number of data points which only occur due to random variations. In this scenario, how many samples do we need to maintain?

(S2) Next, consider a large approximate (say high-dimensional image feature [2]) data set, where we want to build a linear classifier. Because the features are approximate (say due to feature hashing techniques), we model the classifier boundary to be randomly shifted using Gaussian noise. How many samples from this data set do we need to obtain a desired generalization bound?

(S3) Finally, consider one of these scenarios in which we are trying to create an informative subset of the enormous full data set, but have the opportunity to do so in ways more intelligent than randomly sampling. On such a reduced data set one may want to train several types of classifiers, or to estimate the density of various subsets. Can we generate a smaller data set compared to what would be required by random sampling?

The traditional way to study related sample complexity problems is through range spaces (a ground set X, and family of subsets $\mathcal{A}$) and their associated dimension (e.g., VC-dimension [135]). We focus on a smooth extension of range spaces defined on a geometric ground set. Specifically, consider the ground set P to be a subset of points in $\mathbb{R}^d$, and let $\mathcal{A}$ describe subsets defined by some geometric objects, for instance, a halfspace or a ball. Points $p \in \mathbb{R}^d$ that are inside the object (e.g., halfspace or ball) are typically assigned a value 1, and those outside a value 0. In our smoothed setting, points near the boundary are given a value between 0 and 1, instead of discretely switching from 0 to 1.

In learning theory, these smooth range spaces can be characterized by more general notions called P-dimension [108] (or Pseudo-dimension) or V-dimension [134] (or "fat" versions of these [3]) and can be used to learn real-valued functions for regression or density estimation, respectively.

In geometry and data structures, these smoothed range spaces are of interest in studying noisy data. Our work extends some recent work [78,103] which examines a special case of our setting that maps to kernel density estimates, and matches or improves on related bounds for non-smoothed versions.

We next summarize the main contributions in this chapter.

- We define a general class of *smoothed range spaces* (Section 4.2.1), with application to density estimation and noisy agnostic learning, and we show that these can inherit sample complexity results from *linked* non-smooth range spaces (Corollary 4.3.1).

- We define an (ε, τ)-net for a smoothed range space (Section 4.2.3). We show how this can inherit sampling complexity bounds from *linked* non-smooth range spaces (Theorem 9), and we relate this to non-agnostic density estimation and hitting set problems.

- We provide discrepancy-based bounds and constructions for ε-samples on smooth range spaces requiring significantly fewer points than uniform sampling approaches (Theorems 11 and 12), and also smaller than discrepancy-based bounds on the linked binary range spaces.

4.1 Definitions and Background

Recall that we will focus on geometric range spaces $(P, \mathcal{A})$ where the ground set $P \subset \mathbb{R}^d$ and the family of ranges $\mathcal{A}$ are defined by geometric objects. It is common to approximate a range space in one of two ways, as an ε-sample (aka ε-approximation) or an ε-net as we defined in Chapter 1. Given a range space $(P, \mathcal{A})$ where $|P| = m$, then $\pi_{\mathcal{A}}(m)$ describes the maximum number of possible distinct subsets of P defined by some $A \in \mathcal{A}$. If we can bound, $\pi_{\mathcal{A}}(m) \leq Cm^{\nu}$ for absolute constant C, then $(P, \mathcal{A})$ is said to have *shatter dimension* ν.

For instance, the shatter dimension of $\mathcal{H}$ halfspaces in $\mathbb{R}^d$ is d, and for $\mathcal{B}$ balls in $\mathbb{R}^d$ is $d+1$. For a range space with shatter dimension ν, a random sample of size $O((1/\varepsilon^2)(\nu + \log(1/\delta)))$ is an ε-sample with probability at least $1-\delta$ [85,135], and a random sample of size $O((\nu/\varepsilon)\log(1/\varepsilon\delta))$ is an ε-net with probability at least $1-\delta$ [69,98].

An ε-sample Q is sufficient for agnostic learning with generalization error ε, where the best classifier might misclassify some points. An ε-net Q is sufficient for non-agnostic learning with generalization error ε, where the best classifier is assumed to have no error on P.

The size bounds can be made deterministic and slightly improved for certain cases. An ε-sample Q can be made of size $O(1/\varepsilon^{2\nu/(\nu+1)})$ [91] and this bound can be no smaller [92] in the general case. For balls $\mathcal{B}$ in $\mathbb{R}^d$ which have shatter-dimension $\nu = d+1$, this can be improved to $O(1/\varepsilon^{2d/(d+1)} \log^{d/(d+1)}(1/\varepsilon))$ [10,92], and the best-known lower bound is $O(1/\varepsilon^{2d/(d+1)})$. For axis-aligned rectangles $\mathcal{R}$ in $\mathbb{R}^d$ which have shatter-dimension $\nu = 2d$, this can be improved to $O((1/\varepsilon)\log^{d+1/2}(1/\varepsilon))$ [84].

For ε-nets, the general bound of $O((\nu/\varepsilon)\log(1/\varepsilon))$ can also be made deterministic [91], and for halfspaces in $\mathbb{R}^4$ the size must be at least $\Omega((1/\varepsilon)\log(1/\varepsilon))$ [99]. But for halfspaces in $\mathbb{R}^3$ the size can be $O(1/\varepsilon)$ [67,93], which is tight. By a simple lifting, this also applies for balls in $\mathbb{R}^2$. For other range spaces, such as axis-aligned rectangles in $\mathbb{R}^2$, the size bound is $\Theta((1/\varepsilon)\log\log(1/\varepsilon))$ [5,99].

4.1.1 Kernels

A *kernel* is a bivariate similarity function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, which can be normalized so $K(x,x) = 1$ (which we assume through this chapter).

A *kernel range space* [78, 103] $(P, \mathcal{K})$ is an extension of the combinatorial concept of a range space $(P, \mathcal{A})$ (or to distinguish it we refer to the classic notion as a *binary range space*). It is defined by a point set $P \subset \mathbb{R}^d$ and a kernel K. An element K_x of $\mathcal{K}$ is a kernel $K(x, \cdot)$ applied at point $x \in \mathbb{R}^d$; it assigns a value in $[0, 1]$ to each point $p \in P$ as $K(x, p)$. If we use a ball kernel, then each value is exactly $\{0, 1\}$ and we recover exactly the notion of a binary range space for geometric ranges defined by balls.

A binary range space $(P, \mathcal{A})$ is *linked* to a kernel range space $(P, \mathcal{K})$ if the set $\{p \in P \mid K(x, p) \geq \tau\}$ is equal to $P \cap A$ for some $A \in \mathcal{A}$, for any threshold value τ. [78] showed that an ε-sample of a linked range space $(P, \mathcal{A})$ is also an ε-kernel sample of a corresponding kernel range space $(P, \mathcal{K})$. Since all range spaces defined by symmetric, shift-invariant kernels are linked to range spaces defined by balls, they inherit all ε-sample bounds, including that random samples of size $O((1/\varepsilon^2)(d + \log(1/\delta))$ provide an ε-kernel sample with probability at least $1 - \delta$. Then [103] showed that these bounds can be improved through discrepancy-based methods to $O(((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})^{2d/(d+2)})$, which is $O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})$ in $\mathbb{R}^2$.

A more general concept has been studied in learning theory on real-valued functions, where a function f as a member of a function class $\mathcal{F}$ describes a mapping from $\mathbb{R}^d$ to $[0, 1]$ (or more generally $\mathbb{R}$). A kernel range space where the linked binary range space has bounded shatter-dimension ν is said to have bounded V-dimension [134] (see [3]) of ν. Given a ground set X, then for $(X, \mathcal{F})$ this describes the largest subset Y of X which can be shattered in the following sense. Choose any value $s \in [0, 1]$ for all points $y \in Y$, and then for each subset of $Z \subset Y$, there exists a function $f \in \mathcal{F}$ so $f(y) > s$ if $y \in Z$ and $f(y) < s$ if $y \notin Z$. The best sample complexity bounds for ensuring Q is an ε-sample of P based on V-dimension are derived from a more general sort of dimension (called a P-dimension [108] where in the shattering definition, each y may have a distinct $s(y)$ value) requires $|Q| = O((1/\varepsilon^2)(\nu + \log(1/\delta)))$ [85]. As we will see, these V-dimension-based results are also general enough to apply to the to-be-defined smooth range spaces.

4.2 New Definitions

In this chapter, we extend the notion of kernel range spaces to other *smoothed range spaces* that are "linked" with common range spaces, e.g., halfspaces. These inherit the construction bounds through the linking result of [78], and we show cases where these bounds can also be improved. We also extend the notion of ε-nets to kernels and smoothed range spaces, and show linking results for these as well.

4.2.1 Smoothed Range Spaces

Here, we will define the primary smoothed combinatorial object that we will examine, starting with halfspaces, and then generalizing. Let $\mathcal{H}_w$ denote the family of *smoothed halfspaces with width parameter w*, and let $(P, \mathcal{H}_w)$ be the associated smoothed range space where $P \subset \mathbb{R}^d$. Given a point $p \in P$, then smoothed halfspace $h \in \mathcal{H}_w$ maps p to a value $v_h(p) \in [0,1]$ (rather than the traditional $\{0,1\}$ in a binary range space).

We first describe a specific mapping to the function value $v_h(p)$ that will be sufficient for the development of most of our techniques. Let F be the $(d-1)$-flat defining the boundary of halfspace h. Given a point $p \in \mathbb{R}^d$, let $p_F = \arg\min_{q \in F} \|p - q\|$ describe the point on F closest to p. Now we define

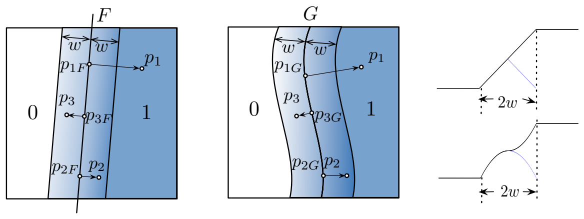
$$v_{h,w}(p) = \begin{cases} 1 & p \in h \text{ and } \|p - p_F\| \geq w \\ \frac{1}{2} + \frac{1}{2}\frac{\|p-p_F\|}{w} & p \in h \text{ and } \|p - p_F\| < w \\ \frac{1}{2} - \frac{1}{2}\frac{\|p-p_F\|}{w} & p \notin h \text{ and } \|p - p_F\| < w \\ 0 & p \notin h \text{ and } \|p - p_F\| \geq w. \end{cases}$$

These points within a slab of width $2w$ surrounding F can take on a value between 0 and 1, where points outside of this slab revert back to the binary values of either 0 or 1.

We can make this more general using a shift-invariant kernel $k(\|p - x\|) = K(p, x)$, where $k_w(\|p - x\|) = k(\|p - x\|/w)$ allows us to parameterize by w. Define $v_{h,w}(p)$ as follows.

$$v_{h,w}(p) = \begin{cases} 1 - \frac{1}{2}k_w(\|p - p_F\|) & p \in h \\ \frac{1}{2}k_w(\|p - p_F\|) & p \notin h. \end{cases}$$

For brevity, we will omit the w and just use $v_h(p)$ when clear. These definitions are equivalent when using the triangle kernel. But, for instance, we could also use a Epanechnikov kernel or Gaussian kernel. Although the Gaussian kernel does not satisfy the restriction that only points in the width $2w$ slab take non $\{0,1\}$ values, we can use techniques from [103] to extend to this case as well. This is illustrated in Figure 4.1. Another property held by this definition that we will exploit is the slope ς of these kernels is bounded by $\varsigma = O(1/w) = c/w$, for some constant c; the constant $c = 1/2$ for triangle and Gaussian, and $c = 1$ for Epanechnikov.



	$\|p - p_F\|$	$?(p \in h)$	$v_h(p)$
p_1	$3w/2$	TRUE	1
p_2	$3w/4$	TRUE	7/8
p_3	$w/2$	FALSE	1/4

Figure 4.1. Illustration of the smoothed halfspace, and smoothed polynomial surface, with function value of three points $\{p_1, p_2, p_3\}$ defined using a triangle kernel.

Finally, we can further generalize this by replacing the flat F at the boundary of h with a polynomial surface G. The point $p_G = \arg\min_{q \in G} \|p - q\|$ replaces p_F in the above definitions. Then the slab of width $2w$ is replaced with a curved volume in $\mathbb{R}^d$; see Figure 4.1. For instance, if G defines a circle in $\mathbb{R}^d$, then v_h defines a disc of value 1, then an annulus of width $2w$ where the function value decreases to 0. Alternatively, if G is a single point, then we essentially recover the kernel range space, except that the maximum height is $1/2$ instead of 1. We will prove the key structural results for polynomial curves in Section 4.4, but otherwise focus on halfspaces to keep the discussion cleaner. The most challenging elements of our results are all contained in the case with F as a $(d-1)$-flat.

4.2.2 ε-Sample in a Smoothed Range Space

It will be convenient to extend the notion of a kernel density estimate to these smoothed range space. A *smoothed density estimate* SDE_P is defined for any $h \in \mathcal{H}_w$ as

$$\text{SDE}_P(h) = \frac{1}{|P|} \sum_{p \in P} v_h(p). \tag{4.1}$$

An *ε-sample Q of a smoothed range space* $(P, \mathcal{H}_w)$ is a subset $Q \subset P$ such that

$$\max_{h \in \mathcal{H}_w} |\text{SDE}_P(h) - \text{SDE}_Q(h)| \leq \varepsilon. \tag{4.2}$$

Given such an ε-sample Q, we can then consider a subset $\bar{\mathcal{H}}_w$ of $\mathcal{H}_w$ with bounded integral (perhaps restricted to some domain like a unit cube that contains all of the data P). If we can learn the smooth range $\hat{h} = \arg\max_{h \in \bar{\mathcal{H}}_w} \text{SDE}_Q(h)$, then we know $\text{SDE}_P(h^*) - \text{SDE}_Q(\hat{h}) \leq \varepsilon$, where $h^* = \arg\max_{h \in \bar{\mathcal{H}}_w} \text{SDE}_P(h)$, since $\text{SDE}_Q(\hat{h}) \geq \text{SDE}_Q(h^*) \geq \text{SDE}_P(h^*) - \varepsilon$. Thus, such a set Q allows us to learn these more general density estimates with generalization error ε.

We can also learn smoothed classifiers, like scenario (S2) in the introduction, with generalization error ε, by giving points in the negative class a weight of -1; this requires separate $(\varepsilon/2)$-samples for the negative and positive classes.

4.2.3 (ε, τ)-Net in a Smoothed Range Space

We now generalize the definition of an ε-net. Recall that it is a subset $Q \subset P$ such that Q "hits" all large enough ranges ($|P \cap A|/|P| \geq \varepsilon$). However, the notion of "hitting" is now less well-defined since a point $q \in Q$ may be in a range but with value very close to 0; if a smoothed range space is defined with a Gaussian or other kernel with infinite support, any point q will have a nonzero value for *all* ranges! Hence, we need to introduce another parameter $\tau \in (0, \varepsilon)$, to make the notion of hitting more interesting in this case.

A subset $Q \subset P$ is an (ε, τ)*-net of smoothed range space* $(P, \mathcal{H}_w)$ if for any smoothed range, $h \in \mathcal{H}_w$ such that $\textsc{sde}_P(h) \geq \varepsilon$, then there exists a point $q \in Q$ such that $v_h(q) \geq \tau$.

The notion of ε-net is closely related to that of hitting sets. A *hitting set* of a binary range space $(P, \mathcal{A})$ is a subset $Q \subset P$ so every $A \in \mathcal{A}$ (not just the large enough ones) contains some $q \in Q$. To extend these notions to the smoothed setting, we again need an extra parameter $\tau \in (0, \varepsilon)$, and also need to only consider large enough smoothed ranges, since there are now an infinite number even if P is finite. A subset $Q \subset P$ is an (ε, τ)*-hitting set of smoothed range space* $(P, \mathcal{H}_w)$ if for any $h \in \mathcal{H}_w$ such that $\textsc{sde}_P(h) \geq \varepsilon$, then $\textsc{sde}_Q(h) \geq \tau$.

In the binary range space setting, an ε-net Q of a range space $(P, \mathcal{A})$ is sufficient to learn the best classifier on P with generalization error ε in the non-agnostic learning setting, that is assuming a perfect classifier exists on P from $\mathcal{A}$. In the density estimation setting, there is not a notion of a perfect classifier, but if we assume some other properties of the data, the (ε, τ)-net will be sufficient to recover them. For instance, consider (like scenario (S1) in the introduction) that P is a discrete distribution so for some "event" points $p \in P$, there is at least an ε-fraction of the probability distribution describing P at p (e.g., there are more than $\varepsilon|P|$ points very close to p). In this setting, we can recover the location of these points since they will have probability at least τ in the (ε, τ)-net Q.

4.3 Linking and Properties of (ε, τ)-Nets

First we establish some basic connections between ε-sample, (ε, τ)-net, and (ε, τ)-hitting set in smoothed range spaces. In binary range spaces, an ε-sample Q is also an ε-net, and a hitting set is also an ε-net; we show a similar result here up to the covering constant τ.

Lemma 8. *For a smoothed range space $(P, \mathcal{H}_w)$ and $0 < \tau < \varepsilon < 1$, an (ε, τ)-hitting set Q is also an (ε, τ)-net of $(P, \mathcal{H}_w)$.*

Proof. The (ε, τ)-hitting set property establishes for all $h \in \mathcal{H}_w$ with $\textsc{sde}_P(h) \geq \varepsilon$, then also $\textsc{sde}_Q(h) \geq \tau$. Since $\textsc{sde}_Q(h) = \frac{1}{|Q|} \sum_{q \in Q} v_h(q)$ is the average over all points $q \in Q$, then it implies that at least one point also satisfies $v_h(q) \geq \tau$. Thus Q is also an (ε, τ)-net. □

In the other direction, an (ε, τ)-net is not necessarily an (ε, τ)-hitting set since the (ε, τ)-net Q may satisfy a smoothed range $h \in \mathcal{H}_w$ with a single point $q \in Q$ such that $v_h(q) \geq \tau$, but all others $q' \in Q \setminus \{q\}$ having $v_h(q') \ll \tau$, and thus $\textsc{sde}_Q(h) < \tau$.

Theorem 8. *For $0 < \tau < \varepsilon < 1$, an $(\varepsilon - \tau)$-sample Q in smoothed range space $(P, \mathcal{H}_w)$ is an (ε, τ)-hitting set in $(P, \mathcal{H}_w)$, and thus also an (ε, τ)-net of $(P, \mathcal{H}_w)$.*

Proof. Since Q is the $(\varepsilon - \tau)$-sample in the smoothed range space, for any smoothed range $h \in \mathcal{H}_w$ we have $|\textsc{sde}_P(h) - \textsc{sde}_Q(h)| \leq \varepsilon - \tau$. We consider the upper and lower bound separately.

If $\textsc{sde}_P(h) \geq \varepsilon$, when $\textsc{sde}_P(h) \geq \textsc{sde}_Q(h)$, we have

$$\textsc{sde}_Q(h) \geq \textsc{sde}_P(h) - (\varepsilon - \tau) \geq \varepsilon - (\varepsilon - \tau) = \tau.$$

And more simply, when $\textsc{sde}_Q(h) \geq \textsc{sde}_P(h)$ and $\textsc{sde}_P(h) \geq \varepsilon \geq \tau$, then $\textsc{sde}_Q(h) \geq \tau$. Thus, in both situations, Q is an (ε, τ)-hitting set of $(P, \mathcal{H}_w)$. And then by Lemma 8, Q is also an (ε, τ)-net of $(P, \mathcal{H}_w)$. □

4.3.1 Relations between Smoothed Range Spaces and Linked Binary Range Spaces

Consider a smoothed range space $(P, \mathcal{H}_w)$, and for one smoothed range $h \in \mathcal{H}_w$, examine the range boundary F (e.g. a $(d-1)$-flat, or polynomial surface) along with a symmetric, shift invariant kernel K that describes v_h. The *superlevel* set $(v_h)^\tau$ is all points $x \in \mathbb{R}^d$ such that $v_h(x) \geq \tau$. Then recall a smoothed range space $(P, \mathcal{H}_w)$ is *linked* to a binary range space $(P, \mathcal{A})$ if every set $\{p \in P \mid v_h(p) \geq \tau\}$ for any $h \in \mathcal{H}_w$ and any $\tau > 0$, is exactly the same as some range $A \cap P$ for $A \in \mathcal{A}$. For smoothed range spaces defined by halfspaces, then the linked binary range space is also defined by halfspaces. For smoothed range spaces defined by points, mapping to kernel range spaces, then the linked binary range spaces are defined by balls.

Joshi *et.al.* [78] established that given a kernel range space $(P, \mathcal{K})$, a linked binary range space $(P, \mathcal{A})$, and an ε-sample Q of $(P, \mathcal{A})$, then Q is also an ε-kernel sample of $(P, \mathcal{K})$. An inspection of the proof reveals the same property holds directly for smoothed range spaces, as the only structural property needed is that all points $p \in P$, as well as all points $q \in Q$, can be sorted in decreasing function value $K(p, x)$, where x is the center of the kernel. For smoothed range space, this can be replaced with sorting by $v_h(p)$.

Corollary 4.3.1 ([78]). *Consider a smoothed range space $(P, \mathcal{H}_w)$, a linked binary range space $(P, \mathcal{A})$, and an ε-sample Q of $(P, \mathcal{A})$ with $\varepsilon \in (0, 1)$. Then Q is an ε-sample of $(P, \mathcal{H}_w)$.*

We now establish a similar relationship to (ε, τ)-nets of smoothed range spaces from $(\varepsilon - \tau)$-nets of linked binary range spaces.

Theorem 9. *Consider a smoothed range space $(P, \mathcal{H}_w)$, a linked binary range space $(P, \mathcal{A})$, and an $(\varepsilon - \tau)$-net Q of $(P, \mathcal{A})$ for $0 < \tau < \varepsilon < 1$. Then Q is an (ε, τ)-net of $(P, \mathcal{H}_w)$.*

Proof. Let $|P| = n$. Then, since Q is an $(\varepsilon - \tau)$-net of $(P, \mathcal{A})$, for any range $A \in \mathcal{A}$, if $|P \cap A| \geq (\varepsilon - \tau)n$, then $Q \cap A \neq \emptyset$.

Suppose $h \in \mathcal{H}_w$ has $\text{SDE}_P(h) \geq \varepsilon$ and we want to establish that $\text{SDE}_Q(h) \geq \tau$. Let $A \in \mathcal{A}$ be the range such that $(\varepsilon - \tau)n$ points with largest $v_h(p_i)$ values are exactly the points in A. We now partition P into three parts (1) let P_1 be the $(\varepsilon - \tau)n - 1$ points with largest v_h values, (2) let y be the point in P with $(\varepsilon - \tau)n$th largest v_h value, and (3) let P_2 be the remaining $n - n(\varepsilon - \tau)$ points. Thus, for every $p_1 \in P_1$ and every $p_2 \in P_2$ we have $v_h(p_2) \leq v_h(y) \leq v_h(p_1) \leq 1$.

Now using our assumption $n \cdot \text{SDE}_P(h) \geq n\varepsilon$ we can decompose the sum

$$n \cdot \text{SDE}_P(h) = \sum_{p_1 \in P_1} v_h(p_1) + v_h(y) + \sum_{p_2 \in P_2} v_h(p_2) \geq n\varepsilon,$$

and hence using upper bounds $v_h(p_1) \leq 1$ and $v_h(p_2) \leq v_h(y)$,

$$\begin{aligned} v_h(y) &\geq n\varepsilon - \sum_{p_1 \in P_1} v_h(p_1) - \sum_{p_2 \in P_2} v_h(p_2) \\ &\geq n\varepsilon - (n(\varepsilon - \tau) - 1) \cdot 1 - (n - n(\varepsilon - \tau)) v_h(y). \end{aligned}$$

Solving for $v_h(y)$ we obtain

$$v_h(y) \geq \frac{n\tau + 1}{n - n(\varepsilon - \tau) + 1} \geq \frac{n\tau}{n - n(\varepsilon - \tau)} \geq \frac{n\tau}{n} = \tau.$$

Since $(P, \mathcal{A})$ is linked to $(P, \mathcal{H}_w)$, there exists a range $A \in \mathcal{A}$ that includes precisely $P_1 \cup y$ (or more points with the same $v_h(y)$ value as y). Because Q is an $(\varepsilon - \tau)$-net of $(P, \mathcal{A})$, Q contains at least one of these points, lets call it q. Since all of these points have function value $v_h(p) \geq v_h(y) \geq \tau$, then $v_h(q) \geq \tau$. Hence, Q is also an (ε, τ)-net of $(P, \mathcal{H}_w)$, as desired. $\square$

This implies that if $\tau \leq c\varepsilon$ for any constant $c < 1$, then creating an (ε, τ)-net of a smoothed range space, with a known linked binary range space, reduces to computing an ε-net for the linked binary range space. For instance, any linked binary range space with shatter-dimension ν has an ε-net of size $O(\frac{\nu}{\varepsilon} \log \frac{1}{\varepsilon})$, including halfspaces in $\mathbb{R}^d$ with $\nu = d$ and balls in $\mathbb{R}^d$ with $\nu = d + 1$; hence there exists $(\varepsilon, \varepsilon/2)$-nets of the same size. For halfspaces in $\mathbb{R}^2$ or $\mathbb{R}^3$ (linked to smoothed halfspaces) and balls in $\mathbb{R}^2$ (linked to kernels), the size can be reduced to $O(1/\varepsilon)$ [67,93,111].

4.4 Min-Cost Matchings within Cubes

Before we proceed with our construction for smaller ε-samples for smoothed range spaces, we need to prepare some structural results about min-cost matchings. Following some basic ideas from [103], these matchings will be used for discrepancy bounds on smoothed range spaces in Section 4.5.

In particular, we analyze some properties of the interaction of a min-cost matching M and some basic shapes ([103] considered only balls). Let $P \subset \mathbb{R}^d$ be a set of $2n$ points. A *matching* $M(P)$ is a decomposition of P into n pairs $\{p_i, q_i\}$ where $p_i, q_i \in P$ and each p_i (and q_i) is in exactly one pair. A *min-cost matching* is the matching M that minimizes $cost_1(M, P) = \sum_{i=1}^{n} \|p_i - q_i\|$. The min-cost matching can be computed in $O(n^3)$ time by [48] (using an extension of the Hungarian algorithm from the bipartite case). In $\mathbb{R}^2$, it can be calculated in $O(n^{3/2} \log^5 n)$ time [136].

Following [103], again we will base our analysis on a result of [11] which says that if $P \subset [0,1]^d$ (a unit cube) then for d a constant, $cost_d(M,P) = \sum_{i=1}^{n} \|p_i - q_i\|^d = O(1)$, where M is the min-cost matching. We make no attempt to optimize constants, and assume d is constant.

One simple consequence, is that if P is contained in a d-dimensional cube of side length ℓ, then $cost_d(M,P) = \sum_{i=1}^{n} \|p_i - q_i\|^d = O(\ell^d)$.

We are now interested in interactions with a matching M for P in a d-dimensional cube of side length ℓ $C_{\ell,d}$ (call this shape an (ℓ,d)-*cube*), and more general objects; in particular C_w a (w,d)-cube and, S_w a slab of width $2w$, both restricted to be within $C_{\ell,d}$. Now for such an object O_w (which will either be C_w or S_w) and an edge $\{p,q\}$ where line segment $\overline{pq}$ intersects O_w define point p_B (resp. q_B) as the point on segment $\overline{pq}$ inside O_w closest to p (resp. q). Note if p (resp. q) is inside O then $p_B = p$ (resp. $q_B = q$), otherwise, it is on the boundary of O_w. For instance, see C_{20w} in Figure 4.2.

Define the *length* of a matching M restricted to an object $O_w \subset \mathbb{R}^d$ as

$$\rho(O_w, M) = \sum_{(q,p)\in M} \min\left\{(2w)^d, \|p_B - q_B\|^d\right\}.$$

Note this differs from a similar definition by [103] since that case did not need to consider when both p and q were both outside of O_w, and did not need the $\min\{(2w)^d, \ldots\}$ term because all objects had diameter 2.

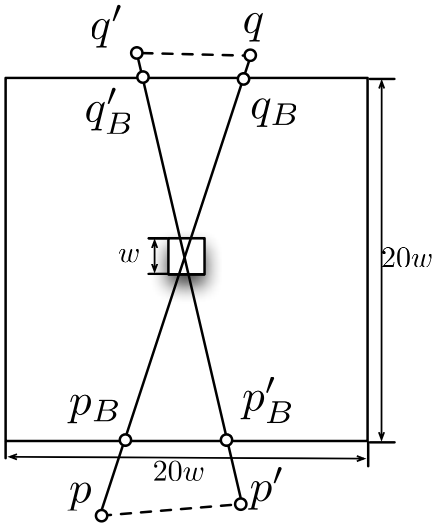


Figure 4.2. (T3) edges.

Lemma 9. *Let $P \subset C_{\ell,d}$, where d is constant, and M be its min-cost matching. For any (w,d)-cube $C_w \subset C_{\ell,d}$ we have $\rho(C_w, M) = O(w^d)$.*

Proof. We cannot simply apply the result of [11] since we do not restrict that $P \subset C_w$. We need to consider cases where either p or q or both are outside of C_w. As such, we have three types of edges we consider, based on a cube C_{20w} of side length $20w$, and with center the same as C_w.

(T1) Both endpoints are within C_{20w} of edge length at most $\sqrt{d}20w$.

(T2) One endpoint is in C_w, the other is outside C_{20w}.

(T3) Both endpoints are outside C_{20w}.

For all (T1) edges, the result of Bern and Eppstein can directly bound their contribution to $\rho(C_w, M)$ as $O(w^d)$ (scale to a unit cube, and rescale). For all (T2) edges, we can also bound their contribution to $\rho(C_w, M)$ as $O(w^d)$, by extending an analysis of [103] when both C_w and C_{20w} are similarly proportioned balls. This analysis shows there are $O(1)$ such edges.

We now consider the case of (T3) edges, restricting to those that also intersect C_w. We argue there can be at most $O(1)$ of them. In particular, consider two such edges $\{p,q\}$ and $\{p',q'\}$, and their mappings to the boundary of C_{20w} as p_B, q_B, p'_B, q'_B; see Figure 4.2. If $\|p_B - p'_B\| \leq 10w$ and $\|q_B - q'_B\| \leq 10w$, then we argue next that this cannot be part of a min-cost matching since $\|p - p'\| + \|q - q'\| < \|p - q\| + \|p' - q'\|$, and it would be better to swap the pairing. Then it follows from the straight-forward net argument below that there can be at most $O(1)$ such pairs.

We first observe that $\|p_B - p'_B\| + \|q_B - q'_B\| \leq 10w + 10w < 20w + 20w \leq \|p_B - q_B\| + \|p'_B - q'_B\|$. Now we can obtain our desired inequality using that $\|p - q\| = \|p - p_B\| + \|p_B - q_B\| + \|q_B - q\|$ (and similar for $\|p' - q'\|$) and that $\|p - p'\| \leq \|p - p_B\| + \|p_B - p'_B\| + \|p'_B - p'\|$ by triangle inequality (and similar for $\|q - q'\|$).

Next, we describe the net argument that there can be at most $O(d^2 \cdot 2^{2d}) = O(1)$ such pairs with $\|p_B - p'_B\| > 10w$ and $\|q_B - q'_B\| > 10w$. First place a $5w$-net $\mathcal{N}_f$ on each $(d-1)$-dimensional face f of C_{20w} so that any point $x \in f$ is within $5w$ of some point $\eta \in \mathcal{N}_f$. We can construct $\mathcal{N}_f$ of size $O(2^d)$ with a simple grid. Then let $\mathcal{N} = \bigcup_f \mathcal{N}_f$ as the union of the nets on each face; its size is $O(d \cdot 2^d)$. Now for any point $p \notin C_{20w}$ let $\eta(p) = \arg\min_{\eta \in \mathcal{N}} \|p_B - \eta\|$ be the closest point in $\mathcal{N}$ to p_B. If two points p and p' have $\eta(p) = \eta(p')$ then $\|p - p'\| \leq 10w$. Hence, there can be at most $O((d \cdot 2^d)^2)$ edges with $\{p, q\}$ mapping to unique $\eta(p)$ and $\eta(q)$ if no other edge $\{p', q'\}$ has $\|p_B - p'_B\| \leq 10w$ and $\|q_B - q'_B\| \leq 10w$.

Concluding, there can be at most $O(d^2 \cdot 2^{2d}) = O(1)$ edges in M of type (T3), and the sum of their contribution to $\rho(C_w, M)$ is at most $O(w^d)$, completing the proof. □

Lemma 10. *Let $P \subset \mathsf{C}_{\ell,d}$, where d is constant, and let M be its min-cost matching. For any width $2w$ slab S_w restricted to $\mathsf{C}_{\ell,d}$ we have $\rho(S_w, M) = O(\ell^{d-1} w)$.*

Proof. We can cover the slab S_w with $O((\ell/w)^{d-1})$ (w, d)-cubes. To make this concrete, we cover $\mathsf{C}_{\ell,d}$ with $\lceil \ell/w \rceil^d$ cubes on a regular grid. Then in at least one basis direction (the one closest to orthogonal to the normal of F), any column of cubes can intersect S_w in at most 4 cubes. Since there are $\lceil \ell/w \rceil^{d-1}$ such columns, the bound holds. Let $\mathcal{C}_w$ be the set of these cubes covering S_w.

Restricted to any one such cube C_w, the contribution of those edges to $\rho(S_w, M)$ is at most $O(w^d)$ by Lemma 9. Now we need to argue that we can just sum the effect of all covering cubes. The concern is that an edge goes through many cubes, only contributing a small amount to each $\rho(C_w, M)$ term, but when the total length is taken to the dth power it is much more. However, since each edge's contribution is capped at $(2w)^2$, we can say that if any edge goes through more than $O(1)$ cubes, its length must be at least w, and its contribution in one such cube is already $\Omega(w)$, so we can simply inflate the effect of each cube towards $\rho(S_w, M)$ by a constant.

In particular, consider any edge $\overline{pq}$ that has $p \in C_w$. Each cube has $3^d - 1$ neighboring cubes, including through vertex incidence. Thus, if edge $\overline{pq}$ passes through more than 3^d cubes, q must be in a cube that is not one of C'_w's neighbors. Thus, it must have length at least w; and hence its length in at least one cube C'_w must be at least $w/3^d$, with its contribution to $\rho(C'_w, M) > w^d/(3^{d^2})$. Thus, we can multiply the effect of each edge in $\rho(C_w, M)$ by $3^{d^2} 2^d = O(1)$ and be sure it is at least as large as the effect of that edge in $\rho(S_w, M)$. Hence

$$\begin{aligned}
\rho(S_w, M) &\leq 3^{d^2} 2^d \sum_{C_w \in \mathcal{C}_w} \rho(C_w, M) \\
&\leq O(1) \sum_{C_w \in \mathcal{C}_w} O(w^d) \\
&= O((\ell/w)^{d-1}) \cdot O(w^d) \\
&= O(\ell^{d-1} w).
\end{aligned}$$

□

We can apply the same decomposition as used to prove Lemma 10 to also prove a result for a w-expanded volume G_w around a degree g polynomial surface G. A degree g polynomial surface can intersect a line at most g times, so for some $\mathsf{C}_{\ell,d}$ the expanded surface $G_w \cap \mathsf{C}_{\ell,d}$ can be intersected by $O(g(\ell/w)^{d-1})$ (w,d)-cubes. Hence, we can achieve the following bound.

Corollary 4.4.1. *Let $P \subset \mathsf{C}_{\ell,d}$, where d is constant, and let M be its min-cost matching. For any volume G_w defined by a polynomial surface of degree g expanded by a width w, restricted to $\mathsf{C}_{\ell,d}$ we have $\rho(G_w, M) = O(g\ell^{d-1}w)$.*

4.5 Constructing ε-Samples for Smoothed Range Spaces

In this section, we build on the ideas from [103] and the new min-cost matching results in Section 4.4 to produce new discrepancy-based ε-sample bounds for smoothed range spaces. The basic construction is as follows. We create a min-cost matching M on P, then for each pair $(p,q) \in M$, we retain one of the two points at random, halving the point set. We repeat this until we reach our desired size. This should not be unfamiliar to readers familiar with discrepancy-based techniques for creating ε-samples of binary range spaces [28, 92]. In that literature similar methods exist for creating matchings "with low-crossing number". Each such matching formulation is specific to the particular combinatorial range space one is concerned with. However, in the case of smoothed range spaces, we show that the min-cost matching approach is a universal algorithm. It means that an ε-sample Q for one smoothed range space $(P, \mathcal{H}_w)$ is also an ε-sample for any other smoothed range space $(P, \mathcal{H}'_w)$, perhaps up to some constant factors. We also show how these bounds can sometimes improve upon ε-sample bounds derived from linked range spaces; herein the parameter w will play a critical role.

4.5.1 Discrepancy for Smoothed Halfspaces

To simplify arguments, we first consider $P \subset \mathbb{R}^2$ extending to $\mathbb{R}^d$ in Section 4.5.4.

Let $\chi : P \to \{-1, +1\}$ be a coloring of P, and define the *discrepancy* of $(P, \mathcal{H}_w)$ with coloring χ as $disc_\chi(P, \mathcal{H}_w) = \max_{h \in \mathcal{H}_w} |\sum_{p \in P} \chi(p) v_h(p)|$. Restricted to one smoothed range $h \in \mathcal{H}_w$ this is $disc_\chi(P, h) = |\sum_{p \in P} \chi(p) v_h(p)|$. We construct a coloring χ using the min-cost matching M of P; for each $\{p_i, q_i\} \in M$ we randomly select one of p_i or q_i to have $\chi(p_i) = +1$, and the other $\chi(q_i) = -1$. We next establish bounds on the discrepancy of this coloring for a ς-bounded smoothed range space $(P, \mathcal{H}_w)$, i.e., where the gradient of v_h is bounded by $\varsigma \leq c_1/w$ for a constant c_1 (see Section 4.2.1).

For any smoothed range $h \in \mathcal{H}_w$, for each pair $\{p_j, q_j\}$ in the matching M, we can now define a random variable $X_j = \chi(p_j) v_h(p_j) + \chi(q_j) v_h(q_j)$. This allows us to rewrite $disc_\chi(P, h) = |\sum_j X_j|$. We can also define a variable $\Delta_j = 2|v_h(p_j) - v_h(q_j)|$ such that $X_j \in \{-\Delta_j/2, \Delta_j/2\}$. Now, following the key insight from [103], we can bound $\sum_j \Delta_j^2$ using results from Section 4.4, which shows up in the following Chernoff bound from [43]: Let $\{X_1, X_2, \ldots\}$ be independent random variables with $\mathbf{E}[X_j] = 0$ and $X_j = \{-\Delta_j/2, \Delta_j/2\}$ then

$$\Pr\left[disc_\chi(P, h) \geq \alpha\right] = \Pr\left[\Big|\sum_j X_j\Big| \geq \alpha\right] \leq 2\exp\left(\frac{-2\alpha^2}{\sum_j \Delta_j^2}\right). \tag{4.3}$$

Lemma 11. *Assume $P \subset \mathbb{R}^2$ is contained in some cube $C_{\ell,2}$, and with min-cost matching M defining χ, and consider a ς-bounded smoothed halfspace $h \in \mathcal{H}_w$ associated with slab S_w. Let $\rho(S_w, M) \leq c_2(\ell w)$ for constant c_2 (see definition of ρ in Section 4.4). Then* $\Pr\left[disc_\chi(P, h) > C\sqrt{\frac{\ell}{w}\log(2/\delta)}\right] \leq \delta$ *for any $\delta > 0$ and constant $C = c_1\sqrt{2c_2}$.*

Proof. Using the gradient of v_h is at most $\varsigma = c_1/w$ and $|v_h(p_j) - v_h(q_j)| \leq \varsigma \max\{2w, \|p_j - q_j\|\}$ we have

$$\sum_j \Delta_j^2 = \sum_j 4(v_h(p_j) - v_h(q_j))^2 \leq 4\varsigma^2 \rho(S_w, M) \leq 4c_1^2/w^2 \cdot c_2 \ell w = 4c_1^2 c_2 \ell / w,$$

where the second inequality follows by Lemma 10 which shows that $\rho(S_w, M) = \sum_j \max \{(2w)^2, \|p_j - q_j\|^2\} \leq c_2(\ell w)$.

We now study the random variable $disc_\chi(P, h) = |\sum_i X_i|$ for a single $h \in \mathcal{H}_w$. Invoking (4.3) we can bound $\Pr[disc_\chi(P, h) > \alpha] \leq 2\exp(-\alpha^2/(2c_1^2 c_2 \ell/w))$. Setting $C = c_1\sqrt{2c_2}$ and

$\alpha = C\sqrt{\frac{\ell}{w}\log(2/\delta)}$ reveals $\Pr\left[disc_\chi(P,h) > C\sqrt{\frac{\ell}{w}\log(2/\delta)}\right] \leq \delta$. □

4.5.2 From a Single Smoothed Halfspace to a Smoothed Range Space

The above theorems imply small discrepancy for a single smoothed halfspace $h \in \mathcal{H}_w$, but this does not yet imply small discrepancy $disc_\chi(P, \mathcal{H}_w)$, for all choices of smoothed halfspaces simultaneously. And in a smoothed range space, the family $\mathcal{H}_w$ is not finite, since even if the same set of points have $v_h(p) = 1$, $v_h(p) = 0$, or are in the slab S_w, infinitesimal changes of h will change $\text{SDE}_P(h)$. So in order to bound $disc_\chi(P, \mathcal{H}_w)$, we will show that there are polynomial in n number of smoothed halfspaces that need to be considered, and then apply a union bound across this set. The proof is deferred to the full version.

Theorem 10. *For $P \subset \mathbb{R}^2$ of size n, for $\mathcal{H}_w$, and value $\Psi(n,\delta) = O\big(\sqrt{\frac{\ell}{w}\log\frac{n}{\delta}}\big)$ for $\delta > 0$, we can choose a coloring χ such that $\Pr[disc_\chi(P,\mathcal{H}_w) > \Psi(n,\delta)] \leq \delta$.*

4.5.3 ε-Samples for Smoothed Halfspaces

To transform this discrepancy algorithm to ε-samples, let $f(n) = disc_\chi(P,\mathcal{H}_w)/n$ be the value of ε in the ε-samples generated by a single coloring of a set of size n. Solving for n in terms of ε, the sample size is $s(\varepsilon) = O(\frac{1}{\varepsilon}\sqrt{\frac{\ell}{w}\log\frac{\ell}{w\varepsilon\delta}})$. We can then apply the *MergeReduce* framework [29]; iteratively apply this random coloring in $O(\log n)$ rounds on disjoint subsets of size $O(s(\varepsilon))$. Using a generalized analysis (c.f., Theorem 3.1 in [102]), we have the same ε-sample size bound.

Theorem 11. *For $P \subset C_{\ell,2} \subset \mathbb{R}^2$, with probability at least $1-\delta$, we can construct an ε-sample of $(P,\mathcal{H}_w)$ of size $O(\frac{1}{\varepsilon}\sqrt{\frac{\ell}{w}\log\frac{\ell}{w\varepsilon\delta}})$.*

To see that these bounds make rough sense, consider a random point set P in a unit square. Then setting $w = 1/n$ will yield roughly $O(1)$ points in the slab (and should roughly revert to the non-smoothed setting); this leads to $disc_\chi(P,\mathcal{H}_w) = O(\sqrt{n}\sqrt{\log(n/\delta)})$ and an ε-sample of size $O((1/\varepsilon^2)\sqrt{\log(1/\varepsilon\delta)})$, basically the random sampling bound. But setting $w = \varepsilon$ so about εn points are in the slab (the same amount of error we allow in an ε-sample) yields $disc_\chi(P,\mathcal{H}_w) = O((1/\sqrt{\varepsilon n}) \cdot \sqrt{\log(n/\delta)})$ and the size of the ε-sample to be $O(\frac{1}{\varepsilon}\sqrt{\log(1/\varepsilon\delta)})$, which is a large improvement over $O(1/\varepsilon^{4/3})$, and the best bound known for non-smoothed range spaces [92].

4.5.4 Generalization to d Dimensions

We now extend from $\mathbb{R}^2$ to $\mathbb{R}^d$ for $d > 2$. Using results from Section 4.4 we implicitly get a bound on $\sum_j \Delta_j^d$, but the Chernoff bound we use requires a bound on $\sum_j \Delta_j^2$. As in [103], we can attain a weaker bound using Jensen's inequality over at most n terms

$$\left(\sum_j \frac{1}{n}\Delta_j^2\right)^{d/2} \leq \sum_j \frac{1}{n}\left(\Delta_j^2\right)^{d/2} \quad \text{so} \quad \sum_j \Delta_j^2 \leq n^{1-2/d}\left(\sum_j \Delta_j^d\right)^{2/d}. \tag{4.4}$$

Replacing this bound and using $\rho(S_w, M) \leq O(\ell^{d-1}w)$ in Lemma 11 and considering $\varsigma = c_1/w$ for some constant c_1 results in the next lemma. Its proof is deferred to the full version.

Lemma 12. *Assume $P \subset \mathbb{R}^d$ is contained in some cube $C_{\ell,d}$ and with min-cost matching M, and consider a ς-bounded smoothed halfspace $h \in \mathcal{H}_w$ associated with slab S_w. Let $\rho(S_w, M) \leq c_2(\ell^{d-1}w)$ for constant c_2. Then $\Pr\left[disc_\chi(P,h) > Cn^{1/2-1/d}(\ell/w)^{1-1/d}\sqrt{\log(2/\delta)}\right] \leq \delta$ for any $\delta > 0$ and $C = \sqrt{2}c_1(c_2)^{1/d}$.*

For all choices of smoothed halfspaces, applying the union bound, the discrepancy is increased by a $\sqrt{\log n}$ factor, with the following probabilistic guarantee,

$$\Pr[disc_\chi(P, \mathcal{H}_w) > Cn^{1/2-1/d}(\ell/w)^{1-1/d}\sqrt{\log(n/\delta)}] \leq \delta.$$

Ultimately, we can extend Theorem 11 to the following.

Theorem 12. *For $P \subset C_{\ell,d} \subset \mathbb{R}^d$, where d is constant, with probability at least $1-\delta$, we can construct an ε-sample of $(P, \mathcal{H}_w)$ of size $O\Big((\ell/w)^{2(d-1)/(d+2)} \cdot \Big(\frac{1}{\varepsilon}\sqrt{\log\frac{\ell}{w\varepsilon\delta}}\Big)^{2d/(d+2)}\Big)$.*

Note this result addresses scenario (S3) from the introduction where we want to find a small set (the ε-sample) so that it could be much smaller than the d/ε^2 random sampling bound, and allows generalization error $O(\varepsilon)$ for agnostic learning as described in Section 4.2.2. When ℓ/w is constant, the exponents on $1/\varepsilon$ are also better than those for binary ranges spaces (see Section 4.1).

CHAPTER 5

CORESETS FOR KERNEL REGRESSION

5.1 Basic Definitions

5.1.1 Coresets for Kernel Regression

The brute force solution of kernel regression is time consuming as each computation calculates KDE and WKDE, which takes $O(|P|)$ time. In this chapter, we show how to scalably apply kernel regression to massive scalar-valued data sets. The main idea is to approximate P with a *coreset* S where $S_x \subset P_x$, and in some cases, this can be relaxed, but each $s \in S$ can potentially be given a scalar value s_y different from the associated original point. In particular, the coreset S should act as a proxy for P, so that for any q the value $\textsc{kr}_S(q)$ should approximate $\textsc{kr}_P(q)$.

The coreset S should be substantially smaller than P, while also preserving the strong approximation guarantees. Any query to $\textsc{kr}_S$ takes time at most proportional to $|S|$ instead of $|P|$, so the size of S directly impacts the efficiency of interacting with $\textsc{kr}_S$. Moreover, if the construction of S is efficient (and ours is roughly as fast as reading the data, or sorting if needed), then the time to compute m values of the kernel regression (common for say visualization) is also reduced by $|P|/|S|$, after factoring the build time. Here are a list of scenarios where such coresets are essential.

- The data are too big to store. For example, Square Kilometer Array, the world's largest radio telescope, receives several terabytes of data per second. Most of the data are in scalar values, such as baseline-corrected power flux density, sensitivity, and receiver temperature, so kernel regression is a good way to track those scalar values over time. However storing all of these data is a challenging problem, let alone analyzing them. Instead of storing all of it, a coreset for kernel regression would keep relevant data that provably behaves like the original data, but needs much less space.

- The data are large and the older parts requires less accuracy. For instance, in analyzing trends in system log data, we want more accurate recent data, but allow more imprecision in historical data. For example, in the CloudLab [115] central power database, power data serve as a way to monitor the cloud performance. They have scalar values and change gradually overtime but may have noisy fluctuations. Kernel regression is a good way to track this, and older data can be kept with less precision.

- These data are for interactive analysis. To interact with very large data stored on disk one can first analyze a small coreset, and then refine to a larger coresets with more accuracy as more precision is needed; this is much more efficient than bringing all relevant data to disk for each query. Instead, we can maintain several layers of different sized coresets. For instance, in spatial data systems, such as Mesowest [72], temperature is connected with each geo-coordinate, to show temperature across the United States, a coarse level coreset is sufficient. But to zoom the map to see the temperature at the state or city level, then a more detailed coreset is required.

5.1.2 Our Approach

To formalize the meaning of $\text{KR}_S(q)$ is "close" to $\text{KR}_P(q)$, we focus on worst-case error guarantees (L_∞ as opposed to L_2 or L_1 more common to KDEs); this ensures we do not have any spurious regression values. This is essential for data analysis, since we want to be able to find important trends and detect outlier points, and also not be fooled into thinking we observe a nonexistent trend or a nonexistent outlier.

Beyond that, the error function should not be affected by either a shift or a scaling of a scalar value, since this is equivalent to changing the units (e.g., Celsius to Fahrenheit). As such a natural bound will be absolute error difference with the bound depending on some quantity that depends on the scaling. We will use $M = \max_{p,p' \in P} |p_y - p'_y|$, the maximum difference between scalar values, so as the scale of the units on p_y changes, M does at the same rate. In particular, we are interested in a coreset S of a data set P such that for some domain $\mathcal{U} \subset \mathbb{R}^d$ that

$$\max_{q \in \mathcal{U}} |\text{KR}_P(q) - \text{KR}_S(q)| \leq \varepsilon M.$$

Our coresets S have size depending linearly on $1/\varepsilon$ and sometimes $\Delta = \max_{p,p' \in P} \|p_x - p'_x\|/\sigma$.

It is worth noting that in setting $\mathcal{U} = \mathbb{R}^d$, such a result may not be possible. The kernel regression definition $\text{KR}_P(q)$ has in its denominator $\text{KDE}_P(q)$, so when $\text{KDE}_P(q)$ is very close to 0, then $\text{KR}_P(q)$ is very unstable. So, we consider a domain $\mathcal{U}$ which is defined by a mild condition on $\text{KDE}_P(q)$; in particular that $\text{KDE}_P(q)$ is above some very small value ρ.

To further put this error bound in perspective, consider the relative error $\max_{q \in \mathcal{U}} \frac{\text{KR}_S(q)}{\text{KR}_P(q)}$ instead. This is unstable whenever $\text{KR}_P(q)$ is close to 0. And, furthermore, the q where $\text{KR}_P(q)$ is small, depends entirely on the units chosen for the p_y values. For instance, we could have $p_y = 32^\circ$ Fahrenheit (not be close to 0) or $p_y = 0^\circ$ Celsius, which is exactly 0 and makes *any* relative error requirement imply no error at all. Since the change of units is meaningless, this error measure is not feasible.

5.1.3 Our Results

Our results focus mainly on $P_x \subset \mathbb{R}^1$ and $P_x \subset \mathbb{R}^2$ (so the x-coordinate(s) naturally represent time or spatial coordinates), although many aspects extend naturally to high dimensions.

We first bound the accuracy of a kernel regression coreset formed by random sampling; these are the first known bounds for the *sample complexity* of kernel regression. It is of particular interest since in many cases the data set provided on input is itself a random sample from some much larger data set or distribution we do not have access to (e.g., a 1% stream from Twitter). So if the input data are indeed sampled, our bounds measure the error present before any analysis is applied. However, random sampling performs poorly compared to most other methods we consider, so it makes sense to further compress them.

We analyze (theoretically and empirically) several straight-forward aggregation techniques to construct coresets. These are of particular interest since they mimic common online aggregation techniques [18]. We also propose some modifications which demonstrate sizable empirically improvements. Interestingly, effective coresets for KDEs [149], do not perform the best for kernel regression.

In particular, our recommended method **G-Aggregate** for $P_x \subset \mathbb{R}^1$, carefully aggregates data over a fixed size nonempty grid cells; it takes $O(|P|)$ after sorting the data. For

Algorithm 1: Z-order (Z)

```
1: Sort data $P_x$ in Z-order; set $h = |P|/|S|$
2: Choose a random number in $r = [0, h-1]$
3: for $i \leftarrow 1$ to $|S|$ do
4:     Put $P_{r+h\cdot(i-1)}$ into $S$
5: return $S$
```

Algorithm 2: Z-Aggregate (ZA)

```
1: Sort data $P_x$ in Z-order; set $h = |P|/|S|$
2: for $i \leftarrow 1$ to $|S|$ do
3:     $P_i = [P_{h\cdot(i-1)}, \cdots, P_{h\cdot i}]$
4:     Put average of all the points in $P_i$ into $S$
5: return $S$
```

$P_x \subset \mathbb{R}^2$, we recommend **Aggregate-Neighbor**, which carefully adds a few points to the coreset from **G-Aggregate**. For a data sets $P_x \subset \mathbb{R}^d$, these both produce a coreset S of size $O(\Delta/\varepsilon\rho)^d$, where $\Delta = \max_{p,p' \in P} \|p_x - p'_x\|/\sigma$, and guarantees for any $q \in \mathbb{R}^d$ with $\text{KDE}_{P_x}(q) > \rho$ that $|\text{KR}_P(q) - \text{KR}_S(q)| \leq \varepsilon M$, where $M = \max_{p,p' \in P} |p_y - p'_y|$. Moreover, these methods are simple to implement and work extremely well on real and synthetic data sets.

5.1.4 Related Work

This is the first work to address sample complexity and coreset size for Nadaraya-Watson kernel regression. There is an enormous body of work on other types of coresets, see the recent survey on coresets [104], including many for parametric regression variants like least-square regression [14] and l_p regression [37].

The only nonparametric regression coreset we are aware of is a form of kernel regression [143] related to the smallest enclosing ball. It predicts the value at a point $q \in \mathbb{R}^d$ as $f(q) = \beta + \sum_{p \in P} \alpha_p K(p_x, q)$ with loss function $\sum_{p \in P} \max\{0, |f(p_x) - p_y| - \bar{\varepsilon}\}$, for a parameter $\bar{\varepsilon}$. Then it finds a set of $O(1/\varepsilon)$ nonzero α_p parameters (corresponding with points in the coreset) so many points satisfy $|f(p_x) - p_y| \leq \bar{\varepsilon}(1+\varepsilon)$. No implementations were attempted.

Rather, we believe the most related work involves coresets for kernel density estimates [9, 78, 103, 149] as mentioned above. We extend some of these results and show others do not work well when translated to the regression variant of this problem.

5.2 Subset Selection Methods

We next describe several natural approaches to compress scalar-valued spatial data. Some of these are likely in use in existing data aggregation frameworks (e.g., RFF [18]), but as far as we know have not been analyzed in how they preserve kernel regression values.

- **Random sampling (RS):** This method simply draws a uniform random sample S from the data set P. This is probably the most common data aggregation method anywhere. In other cases, it is often assumed that even before aggregating data, the data is only a random sample of some unseen larger "true" data set. This is known to approximate kernel density estimates [9, 58, 78], and we will show extends to kernel regression.

Algorithm 3: G-Aggregate (GA)

```
1: Map $P_x$ into grid $G_\gamma$
2: for $g \in G_\gamma(P)$ do
3:     Put average of all the points in $P_g$ into $S$
4: return $S$
```

Algorithm 4: Aggregate-Neighbor (AN)

```
1: Map $P_x$ into grid $G_\gamma$
2: for $g \in G_\gamma(P)$ do
3:     Put average of all the points in $P_g$ into $S$
4: for $g \in G_\gamma(\bar{P})$ adjacent to $G_\gamma(P)$ do
5:     For center $c$ of $g$, put $(c, \textsc{kr}_P(c))$ in $S$
6: return $S$
```

- **k-Center (kCen):** This method creates a k-center clustering of P_x using the greedy Gonzalez algorithms [53]; that finds a set of k center points which (with a factor 2) minimizes the distance to the furthest data point. This is inspired by both a recent way to approximate the kernel mean (equivalent to the KDE) [36] and also the initial step in (improved) fast Gauss transforms [146]. It takes $O(kn)$ time to find the center set, and then data points can be aggregated to the closest center in as much time.

- **Sorting-based approaches:** For $P_x \subset \mathbb{R}^1$, these methods just sort the points, and choose S as evenly spaced points in the sorted order. Inspired by the KDE coreset algorithm in Chapter 1, we can extend to higher dimensions using the Z-order space-filling curve to implicitly define a single ordering over the data points which attempts to preserve spatial locality. Hence, we refer to it as **Z-order (Z)**. Also, inspired by this approach we take a random point from each block in the sorted order instead of the first or last of each block deterministically.

  As an extension, we propose **Z-Aggregate (ZA)** which is more careful on how it represents each interval. It again sorts the x-coordinate(s) of P by Z-order, and then for a set of consecutive points P_i of size h, $[h(i-1), hi)$ for $i = 1, 2, \ldots, k$, choose $s_x = \frac{1}{|P_i|}\sum_{p \in P_i} p_x$ and $s_y = \frac{1}{|P_i|}\sum_{p \in P_i} p_y$ as the ith point in S.

- **Grid-based approaches:** Define a grid G_γ into square grid cells (intervals for $P_x \subset \mathbb{R}$) of side length γ. It will be convenient to designate $G_\gamma(P)$ as the nonempty grid cells, and $G_\gamma(\bar{P})$ as the empty grid cells. For a grid g, define $P_g \subset P = \{p \in P \mid p_x \in g\}$, the points which fall in g. In the basic method **Grid (G)**, for each $g \in G_\gamma(P)$, randomly place one point from P_g into S, and give it a weight $|P_g|$.

  In an extension **G-Aggregate (GA)**, for each $g \in G_\gamma(P)$ we create a new point to place in S as (s_x, s_y) defined $s_x = \frac{1}{|P_g|}\sum_{p \in P_g} p_x$ and $s_y = \frac{1}{|P_g|}\sum_{p \in P_g} p_y$.

  The above algorithms can be subtly further improved by adding extra points in the empty grids with nonempty grids as neighbors, we call this **Aggregate-Neighbor (AN)**. Specifically, these empty but adjacent cells generate a point at the cell center c with value equal to the kernel regression value $\text{KR}_P(c)$. This takes a bit longer than just performing an aggregate, but these empty but adjacent cells are few so the time burden is negligible. This is inspired by the work [22] and the illustrative toy example in Figure 5.1. We will see the improvement is especially significant for $P_x \subset \mathbb{R}^2$.

5.2.1 Progressive Grid-Based Approaches

In many scenarios, $P_x \subset \mathbb{R}$ and this coordinate represents time. Let the current time $t_{\text{NOW}} : x = 0$, and so all other values are negative (say 5 hours ago is $x = -5$). In these settings, we might only examine windows of the data over $x \in [-T, 0]$, that is including now, and up to T time units into the past. Further, we can assume over any view we

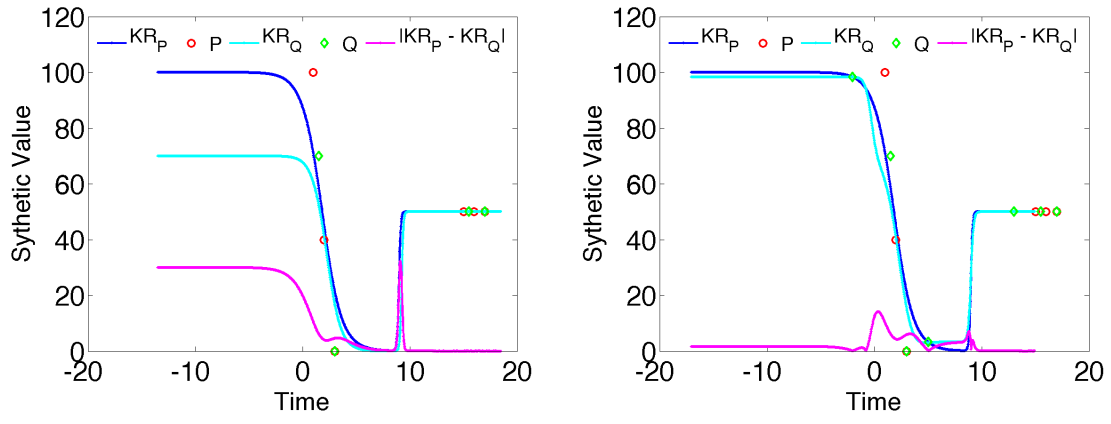


Figure 5.1. Example improvement of **Aggregate-Neighbor** (right) over **G-Aggregate** (left). Input $P = \{(1\ 100), (2\ 40), (3\ 0), (15\ 50), (16\ 50), (17\ 50)\}$. With **G-Aggregate** with $\gamma = 2$, the coreset $Q = \{(1.5\ 70), (3\ 0), (15.5\ 50), (17\ 50)\}$. The largest errors occur at $x < 0$ and around $x = 9$. If we add the extra points at the empty grid cells with nonempty neighbor grids, i.e., **Aggregate-Neighbor**, then L_∞ is significantly reduced. We add three points $(-1\ 98.3124), (5\ 3.2559)$, and $(13\ 50)$.

would set the bandwidth σ so that $\Delta = \max_{p,p' \in P} \|p_x - p'_x\|/\sigma = T/\sigma$ is upper bounded; otherwise, the smoothing is below the resolution of the what can fit in a view window (its too noisy).

For these scenarios, we design a progressive approach where we allow more errors for older data points. Extending the grid-based approaches, as data becomes older (new points arrive) we increase the grid resolution γ, and further compress the data. Specifically, we divide P into regions $R_1, \ldots, R_r$ so the resolution γ_i used in region R_i is $\gamma_i = a^{i-1}\gamma_1$, where a is a constant (we use $a = 1.5$ in our experiments). Setting the width of region $\mathsf{width}(R_i) = a^{i-1}\mathsf{width}(R_1)$ ensures that there are the same number of grid cells in each region. Then, for a fixed resolution in the first region, the size of the coreset will grow only logarithmically with time.

5.3 Analysis

We start by providing some structural lemmas that relate approximations of kernel density estimates and weighted kernel density estimates to kernel regression. Then we will use these results to bound the accuracy of specific techniques.

Our goal in each case is to show that the coreset S approximates the full data set P in the following sense for parameters $\rho, \varepsilon \in (0,1)$. For any $q \in \mathbb{R}^d$ such that $\textsc{kde}_P(q) > \rho$, then

$$|\text{KR}_P(q) - \text{KR}_S(q)| \leq \varepsilon M,$$

where $M = \max_{p,p' \in P} |p_y - p'_y|$. Then call S an (ρ, ε)-*coreset* of P.

We believe such strong worst case bounds should be surprising. If we revisit Figure 5.1 we can observe that removing one point can cause error in $\text{KR}_P(q) - \text{KR}_S(q)$ on the order of M (in this case $M/4$).

- **Structural results:** We need a few definitions and previous results before we can begin stating our new structural tools. Recall in Chapter 4, a data set X and a family of subsets $\mathcal{A}$ define a *range space* $(X, \mathcal{A})$, and the range space's VC-dimension (informally) describes the combinatorial complexity of the ranges.

  A *relative* (ρ, ε)*-approximation* of $(X, \mathcal{A})$ is a set Y

$$\max_{A \in \mathcal{A}} \left| \frac{|A \cap X|}{|X|} - \frac{|A \cap Y|}{|Y|} \right| \leq \varepsilon \max \left\{ \frac{|A \cap X|}{|X|}, \rho \right\}.$$

Similarly, define a *relative* (ρ, ε)*-approximation* of (P, K) for kernel K as a set S such that

$$\max_{x \in \mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_S(x)| \leq \varepsilon \max\{\text{KDE}_P(x), \rho\}.$$

Define a (ρ, ε)*-approximation for kernel regression of* P as a set S such that $\text{KDE}_P(q) \geq \rho$, then

$$|\text{KR}_S(q) - \text{KR}_P(q)| \leq \varepsilon M,$$

where $M = \max_{p,p' \in P} |p_y - p'_y|$. Define a (non-relative) ε-approximation Y of a range space $(X, \mathcal{A})$, so

$$\max_{A \in \mathcal{A}} \left| \frac{|A \cap X|}{|X|} - \frac{|A \cap Y|}{|Y|} \right| \leq \varepsilon.$$

It is know an ε-approximation can be constructed, with probability at least $1 - \delta$ via a random sample S of size $O((1/\varepsilon^2)(\nu + \log 1/\delta))$, and a relative (ρ, ε)-approximation with size $O((1/\rho\varepsilon^2)(\nu \log(1/\rho) + \log(1/\delta)))$ [66, 85]. Given an ε-approximation S of a range space linked to K, then it is known [78] that it is also a (non-relative) ε-approximation of (P, K). In the Appendix, we generalize this linking result (roughly following the structure of the proof in [78]) to relative (ρ, ε)-approximations.

Theorem 13. *For any kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ linked to a range space ($\mathbb{R}^d, \mathcal{A}$), a relative (ρ, ε)-approximation S of $(P, \mathcal{A})$ is a $(\rho K^+, 2\varepsilon)$-approximation of (P, K), where $K^+ = \max_{p,q \in P} K(p, q)$.*

Next we provide a sufficient condition for (ρ, ε)-approximation for kernel regression.

Lemma 13. *For error parameters $\alpha, \beta, \rho > 0$, with $\alpha \leq 1/2$, consider a point set $P \subset \mathbb{R}^{d+1}$. Let S be a coreset of P so that for any query point $q \in \mathbb{R}^d$, both*

$$|\text{KDE}_P(q) - \text{KDE}_S(q)| < \alpha \max\{\text{KDE}_P(q), \rho\}$$
$$|\text{WKDE}_P(q) - \text{WKDE}_S(q)| < \beta M.$$

Then for any $q \in \mathbb{R}^d$ such that $\text{KDE}_P(q) \geq \rho$, then $|\text{KR}_S(q) - \text{KR}_P(q)| \leq 4(\alpha + \beta/\rho)M$.

Proof. Change the units of p_y so all values lie between 1 and 2. The shifting of these values does not change the approximation factor βM, but the rescaling of the range changes the bound to $|\text{WKDE}_P(q) - \text{WKDE}_S(q)| \leq \beta$, and also ensures $1 \leq \text{KR}_P(q) \leq 2$. And recall, p_y values have no bearing on $\text{KDE}_P(q)$.

By using the Gaussian kernel we have $\text{KDE}_S(q) > 0$ and also $0 \leq \text{WKDE}_P(q) \leq 2$. Thus, we can consider relative error bounds, using $\text{KDE}_P(q) > \rho$, and hence also $\text{WKDE}_P(q) > \rho$.

$$\begin{aligned}
\frac{\text{KR}_S(q)}{\text{KR}_P(q)} &= \frac{\text{WKDE}_S(q)}{\text{WKDE}_P(q)} \frac{\text{KDE}_P(q)}{\text{KDE}_S(q)} \\
&\geq \left(1 - \frac{\beta}{\text{WKDE}_P(q)}\right)\left(1 - \frac{\alpha}{1+\alpha}\right) \\
&= 1 - \frac{\beta}{\text{WKDE}_P(q)} - \frac{\alpha}{1+\alpha} + \frac{\alpha\beta}{(1+\alpha)\text{WKDE}_P(q)} \\
&\geq 1 - \frac{\beta}{\rho} - \alpha
\end{aligned}$$

Next, we see the relative error bound is slightly different in the other direction.

$$\begin{aligned}
\frac{\text{KR}_S(q)}{\text{KR}_P(q)} &= \frac{\text{WKDE}_S(q)}{\text{WKDE}_P(q)} \frac{\text{KDE}_P(q)}{\text{KDE}_S(q)} \\
&\leq \left(1 + \frac{\beta}{\text{WKDE}_P(q)}\right)\left(1 + \frac{\alpha}{1-\alpha}\right) \\
&= 1 + \frac{\beta}{\text{WKDE}_P(q)} + \frac{\alpha}{1-\alpha} + \frac{\alpha\beta}{(1-\alpha)\text{WKDE}_P(q)} \\
&\leq 1 + \frac{\beta}{\rho} + \frac{\alpha}{1-\alpha} + \frac{\alpha\beta}{(1-\alpha)\rho} \\
&= 1 + \frac{\beta}{(1-\alpha)\rho} + \frac{\alpha}{1-\alpha} \\
&\leq 1 + \frac{2\beta}{\rho} + 2\alpha
\end{aligned}$$

Together these imply $\frac{\text{KR}_S(q)}{\text{KR}_P(q)} \in [1 - \beta/\rho - \alpha, 1 + 2\beta/\rho + 2\alpha]$. This translates to the following additive error

$$\begin{aligned}|\text{KR}_P(q) - \text{KR}_S(q)| &\leq 2(\beta/\rho + \alpha)\text{KR}_P(q)\\ &\leq 2(\beta/\rho + \alpha)2M = 4(\alpha + \beta/\rho)M.\end{aligned}$$

□

We also need another property about the slope of the Gaussian kernel. This is the only bound specific to the Gaussian kernel, so for any other kernels with a similar bound (e.g., Triangle, Epanechnikov) the remaining analysis and algorithms can apply.

Lemma 14. *A unit Gaussian kernel $K(x) = \exp(-x^2/2\sigma^2)$ is $1/\sigma$-Lipschitz.*

Proof. By taking the first derivative of K with respect to x, we have $\frac{dK(x)}{dx} = \exp(-\frac{x^2}{2\sigma^2})(-\frac{x}{\sigma^2})$. Take the second derivative $\frac{d^2K(x)}{dx^2} = \exp(-\frac{x^2}{2\sigma^2})(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})$ and set $\frac{d^2K(x)}{dx^2} = 0$. We get $x = \pm\sigma$, and thus $|\frac{dK(x)}{dx}|$ has the maximum values on $x = \pm\sigma$, equals to $\exp(-\frac{1}{2})(\frac{1}{\sigma}) \leq 1/\sigma$. So a unit Gaussian kernel is $1/\sigma$-Lipschitz. □

5.3.1 Accuracy of Random Sampling

We start by analyzing how kernel regression is preserved under random sampling. In many cases the "input" data to a problem should actually be modeled as a random sample of some much larger set, or it may be done as a first pass on data to reduce its complexity.

The key structural result will be on sampling weighted sets.

Lemma 15. *For a weighted point set (P, w) with $P \subset \mathbb{R}^d$ of size n, then a random sample of points $Q \subset P$ of size $s = O((1/\varepsilon^2)(d + \log(1/\delta)))$, with probability at least $1 - \delta$, satisfies for any $B \in \mathcal{B}$*

$$\left|\frac{1}{n}\sum_{p \in P \cap B} w(p) - \frac{1}{s}\sum_{p \in Q \cap B} w(p)\right| \leq \varepsilon M,$$

where $M = \max_{p \in X} w(p) - \min_{p \in X} w(p)$.

Proof. First assume $\max_{p \in X} w(p) = 1$ and that $\min_{p \in X} w(p) = 0$; then $M = 1$. Otherwise, we can simply "change the units" by uniformly shifting and scaling all w values to reach this scenario.

We first consider (X, w) as a point set $P \subset \mathbb{R}^{d+1}$, where the y-coordinate is $w(p)$. Then, we consider the range space $(P, \mathcal{A})$ where $\mathcal{A}$ defines the set of subsets induced by ranges which are balls in the first d coordinates, and an interval in the y-coordinate; we refer to them as hypercylinders. The range space has VC-dimension $O(d)$. For a given query $B \in \mathcal{B}$ on X (B is the ball in $\mathbb{R}^d$ on the x-coordinates), we are interested hypercylinders $A \in \mathcal{A}$ so that the x-coordinates are restricted to those in our query choice of B.

In fact, we can break the hypercylinder R, which implicitly has a y-interval of $[0,1]$, up into $\eta = c/\varepsilon$ disjoint hypercylinders (design constant c so that c/ε is an integer), each with the same ball B in x-coordinates and a y width of ε/c. Let P_i be the set P restricted to ith such y interval. We can round all values within the interval to a value $v_i = i \cdot (c/\varepsilon)$, incurring at most ε/c error. Then if each ith piece's sample Q_i is off in count by α_i and $|\sum_i \alpha_i| \leq \varepsilon n/2$, then we can say the total error is at most $|P|\varepsilon/c + n\varepsilon/2$. Setting $c \geq 2$, ensures the total as is at most εn as desired.

However, *individually* bounding each α_i to be small is hard. If there are few points in one of the levels, then we get a poor estimate on the count in Q_i using standard techniques. Instead we can bound $\sum_i \alpha_i$ *in aggregate*. By the definition of ε-samples, if Q is an $\varepsilon/2$-sample of $(P, \mathcal{A})$, then $|\sum_{r=i}^{j} \alpha_i| \leq \varepsilon/2 \cdot n$ for all $i, j \in [1, \eta]$. And this holds by our random sample with probability at least $1 - \delta$.

Now we can write the total error from (P, w) to (Q, w) in an ball $B \in \mathcal{B}$ as

$$\begin{aligned}
\frac{1}{n}\sum_{p \in P \cap I} w(p) &= \frac{1}{n}\sum_{i=1}^{\eta}\sum_{p \in P_i \cap B} w(p) \\
&\leq \frac{1}{n}\sum_{i=1}^{\eta}\sum_{p \in P_i \cap B} (v_i + \varepsilon/c) \\
&= \frac{\varepsilon}{c} + \frac{1}{n}\sum_{i=1}^{\eta} v_i |P_i \cap B| \\
&\leq \frac{\varepsilon}{c} + \frac{1}{n}\sum_{i=1}^{\eta} v_i \left(\alpha_i + \frac{n}{s}|Q_i \cap B|\right) \\
&= \frac{\varepsilon}{c} + \frac{1}{n}\sum_{i=1}^{\eta} v_i \alpha_i + \frac{1}{s}\sum_{1=1}^{\eta}\sum_{p \in Q_i \cap B} v_i \\
&\leq \frac{\varepsilon}{c} + \frac{1}{n}\sum_{i=1}^{\eta} \alpha_i + \frac{1}{s}\sum_{i=1}^{\eta}\sum_{p \in Q_i \cap B} (w(p) + \varepsilon/c) \\
&\leq \frac{2\varepsilon}{c} + \frac{\varepsilon}{2} + \frac{1}{s}\sum_{p \in Q \cap B} w(p).
\end{aligned}$$

Setting $c \geq 4$, and repeating the argument symmetrically to show the lower bound, we obtain that for any $B \in \mathcal{B}$

$$\left| \frac{1}{n} \sum_{p \in P \cap B} w(p) - \frac{1}{s} \sum_{p \in Q \cap B} w(p) \right| \leq \varepsilon.$$

□

These results generalize to weighted kernel density estimates, for centrally-symmetric and non-increasing (as function of distance from center) kernels, following [78]. The only change is using the weighted bound in Lemma 15, in place of where Joshi *et.al.* used the unweighted bound in the definition of a ball-range space linked with the aforementioned kernels.

Theorem 14. *Consider any kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ linked to $(\mathbb{R}^d, \mathcal{B})$. For a weighted point set (X, w) with $X \subset \mathbb{R}^d$, then a random sample $Q \subset X$ of size $s = O((1/\varepsilon^2)(d + \log(1/\delta)))$, with probability at least $1 - \delta$, for any $x \in \mathbb{R}^d$ satisfies*

$$|\text{WKDE}_{X,w}(x) - \text{WKDE}_{Q,w}(x)| \leq \varepsilon M,$$

where $M = \max_{p \in X} w(p) - \min_{p \in X} w(p)$.

Now we are ready to show the main result.

Theorem 15. *Consider a point set $P \subset \mathbb{R}^{d+1}$ of arbitrary size, and parameters $\rho, \varepsilon \in (0, 1)$. If S is a uniform sample from P of size $O(\frac{1}{\varepsilon^2 \rho^2}(d \log(1/\rho) + \log(2/\delta)))$, then with probability at least $1 - \delta$, the set S is a (ρ, ε)-approximation for kernel regression on P.*

Proof. For a binary range space (such as $(P, \mathcal{B})$) with constant VC-dimension [135] ν, a random sample S of size $k = O(\frac{1}{(\varepsilon')^2 \rho}(\nu \log(1/\rho) + \log(2/\delta)))$ provides an (ρ, ε')-sample with probability at least $1 - \delta/2$ [66, 85]. Theorem 13 gives a linking result for kernel density estimate, implying that this is also a relative $(\rho, 2\varepsilon')$-coreset for a kernel where $K(x, x) = 1$. This satisfies the first condition of Lemma 13 with $\alpha = 2\varepsilon'$.

Second, we invoke Theorem 14 so that we have with probability at least $1 - \delta/2$ that $|\text{WKDE}_P(q) - \text{WKDE}_S(q)| \leq (\varepsilon' \rho) M$, hence satisfying the second condition of Lemma 13 with $\beta = \varepsilon' \rho$.

Setting $\varepsilon' = \varepsilon/16$ invoking Lemma 13, then with probability at least $1 - (\delta/2 + \delta/2) = 1 - \delta$, for any $q \in \mathbb{R}^d$ that $|\text{KR}_S(q) - \text{KR}_P(q)| \le 4(\varepsilon/8 + (\varepsilon\rho/16)/\rho)M \le \varepsilon M$. $\square$

5.3.2 Accuracy of Grid-Based Approaches

We first bound the error in **Grid**. This implies other related algorithms (**G-Aggregate**, **Aggregate-Neighbor**) will have the same asymptotic error bounds for d constant.

Theorem 16. *Grid with $\gamma = \frac{\varepsilon\sigma\rho}{8\sqrt{d}}$ produces S, a (ρ,ε)-coreset for the kernel regression of $P \subset \mathbb{R}^{d+1}$.*

Proof. We will prove bounds on both error in KDE_P and WKDE_P separately, then combine them with Lemma 13. This algorithm maps all points P_g for a grid cell $g \in G_\gamma$ to a single point, and by reweighting, changes each points location by at most $\gamma\sqrt{d}$. Using that K is $(1/\sigma)$-Lipschitz, this changes KDE_P by at most $\gamma\sqrt{d}/\sigma$ in KDE_S. Only considering $\text{KDE}_P(q) \ge \rho$, then $|\text{KDE}_P(q) - \text{KDE}_S(q)| \le \frac{\gamma\sqrt{d}}{\rho\sigma}\max\{\rho, \text{KDE}_P(q)\}$.

For WKDE_P the analysis is similar, but we may also replace p_y for a point $p \in P_g$ with a different s_y. We can bound $|p_y - s_y| \le M = \max_{p,p' \in P} |p_y - p'_y|$. Hence for all $q \in \mathbb{R}^d$, then $|\text{WKDE}_P(q) - \text{WKDE}_S(q)| \le \gamma\sqrt{d}M/\sigma$.

Combining these two bounds together with Lemma 13 we obtain (for q with $\text{KDE}_P(q) \ge \rho$) that $|\text{KR}_P(q) - \text{KR}_S(q)| \le 4(\frac{\gamma\sqrt{d}}{\rho\sigma} + \frac{\gamma\sqrt{d}}{\sigma}/\rho)M = 4(\frac{\varepsilon}{8} + \frac{\varepsilon\rho}{8}/\rho)M = \varepsilon M$. $\square$

We bound coreset size with $\Delta = \max_{p,p' \in P} \|p_x - p'_x\|/\sigma$.

Corollary 5.3.1. *For $P \subset \mathbb{R}^{d+1}$ for constant d, methods **Grid**, **G-Aggregate**, and **Aggregate-Neighbor**, run in $O(|P|)$ time, and return S a (ρ,ε)-coreset for kernel regression of P of size at most $O((\Delta/\varepsilon\rho)^d)$.*

- **Accuracy of progressive grid-based methods** If the size $\mathsf{width}(R_1)$ of the first region in the progressive methods is a constant, there are at most $O(\log\Delta)$ regions. Set $\gamma = \varepsilon\sigma\rho/8 \cdot a^{i-1}$, so each region has a grid with $O(1/\varepsilon\rho)$ cells.

Corollary 5.3.2. *For $P \subset \mathbb{R}^2$, under any allowable view window of size T and scaling so T/σ is fixed, then the progressive Grid approach achieves an (ρ,ε)-coreset for kernel regression of P of size at most $O((1/\varepsilon\rho)\log\Delta)$.*

- **Accuracy bounds for other methods** Despite bounds for $|\text{KDE}_P(q) - \text{KDE}_S(q)|$ for other methods (e.g., **Z-order**) we are not able to show these for WKDE_P and, hence KR_P.

5.4 Experiments

Here we run an extensive set of experiments to validate our methods. We compare KR_P where $P_x \subset \mathbb{R}^1$, $P_x \subset \mathbb{R}^2$, and $P_x \subset \mathbb{R}^6$ with kernel regression under smaller coreset KR_S for both synthetic and real data. To show our methods work well in large data sets, we use large real data set ($n = 2$ million and 24 million) and synthetic data ($n = 1$ million) for $P_x \subset \mathbb{R}^1$, and real data set ($n = 1$ million) for $P_x \subset \mathbb{R}^2$. Our algorithms scale well beyond these sizes, but evaluating error was prohibitive.

5.4.1 Data Sets

For real data, we consider "Individual Household Electric Power Consumption" data set on UCI Machine Learning Repository. The number of instances is 2,075,259, we use the first three attributes to do kernel regression. Date, time (together for x-value), and global active power (for y-value): household global minute-averaged active power (in kilowatt). This data set has gaps on the x-axis, and kernel regression does a nice job of interpolating those gaps.

To demonstrate the effectiveness of progressive grids, we use a "CloudLab" data set. CloudLab [115] is cloud computing platform, and we have obtained a trace of power usage from the Utah site with 400 million values. We use the most recent 10-month window which has size 24,351,363.

The time series synthetic data $P_x \subset \mathbb{R}^1$ is generated using formula: $y_i = c + \phi y_{i-1} + N(0, \sigma)$, where the x-coordinates are $i = 1, 2, \cdots$ and y_i is the corresponding y coordinates. It mimics a stock price so the next data depends on the previous one plus some random noise. In the experiment, we set $c = 0, \phi = 1, y_0 = 10, \sigma = 1$ and generate 1 million points. The original data and regression based on the first 10,000 points with bandwidth 50 and 200 is shown on Figure 1.2.

For $P_x \subset \mathbb{R}^2$ real data set, we consider OpenStreetMap data from the state of Iowa. Specifically, we use the longitude and latitude of all highway data points as P_x and time stamp as P_y. Kernel regression on this data set can give a good approximation of when the highway data point is added.

For the high-dimensional experiment, we consider two data sets. One is house price data set (CAD) in StatLib [97] and the other is Physicochemical Properties of Protein Tertiary Structure Data set (CASP) from UCI machine learning repository. CAD data set contains 20,640 observations on housing prices with 9 economic covariates and CASP data set has 45,730 data points for 10 random variables. For both data sets, we use the first 6 features to do the kernel regression.

5.4.2 Effectiveness of Coresets

Coresets guarantee that kernel regression error is bounded for *all* values of $q \in \mathbb{R}$ (as long as the data are not too sparse). But evaluating at all of these points, is by definition, impossible. As a result, we evaluate over a very fine covering of evaluation points (in our case 128,000 for $P_x \subset \mathbb{R}^1$ and 512,000 for $P_x \subset \mathbb{R}^2$). We have plotted error as the number of evaluation points increase and observed that all methods clearly converge well before this many samples.

In more detail, we randomly generate a evaluation point q in the domain $\mathbb{R}$ for $P_x \subset \mathbb{R}$ and q in the domain $\mathbb{R}^2$ for $P_x \subset \mathbb{R}^2$, without the restriction $\text{KDE}_P(q) > \rho$. With fixed coreset size 64,000, we experiment on the number of evaluation points from 1,000 to 128,000 for $P_x \subset \mathbb{R}$ and 1,000 to 512,000 for $P_x \subset \mathbb{R}^2$ in Figure 5.2. As the number of evaluation points increases, the value of maximum error in the domain will consistently approach some error value and we can then have some confidence that we have the correct worst case error as this processes plateaus. Under all the subset selection methods (Figure 5.2), the errors are steady at size 128,000 for $P_x \subset \mathbb{R}$ and 512,000 for $P_x \subset \mathbb{R}^2$, so we use evaluation points of size 128,000 for $P_x \subset \mathbb{R}$ and 512,000 for $P_x \subset \mathbb{R}^2$ in the following experiments.

Since all the methods are randomized algorithms, we run all the subset selection methods ten times and use the average errors as the final results. The bandwidth is set to 400 for the real data set in $\mathbb{R}^1$, 50 for the synthetic data set in $\mathbb{R}^1$, and 50 for real data in $\mathbb{R}^2$; other bandwidths have similar performance.

Figure 5.3 shows all the methods converge as the size of the coreset increases. The exception is **Z-Aggregate** on real data in $\mathbb{R}$; on inspection, the problem occurs in sparse regions, similar to Figure 5.1. **G-Aggregate** and **Aggregate-Neighbor** (and sometimes **Z-Aggregate**) work significantly better compared to all the other methods in all data sets

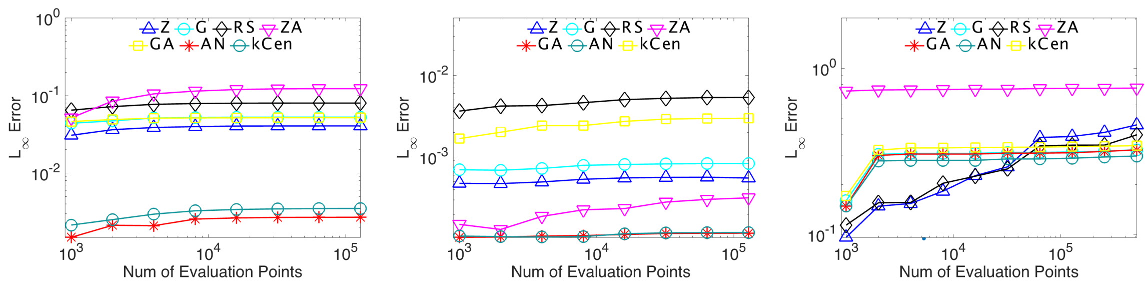


Figure 5.2. The maximum L_∞ error found based on the number of evaluation points on real (left) and synthetic data (middle) when $P_x \subset \mathbb{R}^1$, and real data (right) when $P_x \subset \mathbb{R}^2$.

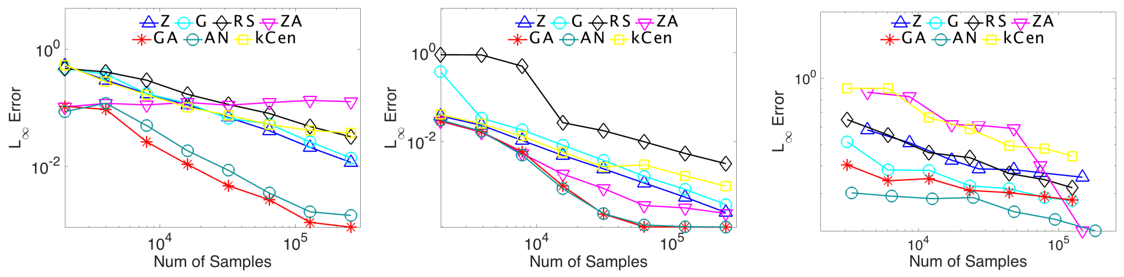


Figure 5.3. L_∞ error for coresets when also testing sparse regions on real data(left) and synthetic data(middle) when $P_x \subset \mathbb{R}$, and real data(right) when $P_x \subset \mathbb{R}^2$.

with $P_x \subset \mathbb{R}$. They consistently decrease, at certain sizes have one or two orders of magnitude less error, and obtain virtually no error at size about 50,000. Even when the size of the coreset is small, **G-Aggregate** and **Aggregate-Neighbor** have very small errors and converge very fast when the size increases. For $P_x \subset \mathbb{R}^2$, **Aggregate-Neighbor** achieve noticeably smaller error, but **G-Aggregate** (and **Grid**) perform well and are simpler.

In particular, **G-Aggregate** and **Aggregate-Neighbor** stay *at least* one order of magnitude smaller in error than **Random Sampling**. This indicates it is much better to aggregate based on x-value than just randomly sample. It also justifies further thinning the data with these methods if the data should be modeled as a random sample since the additional error introduced would be negligible compared to what was already present due to the sampling.

The grid-based methods also consistently outperform the (z-order) sorting-based methods, so it is better to compress based on the x-coordinate change, rather than on the number of points.

Filling in a few neighbor values (the **-Neighbor** method) can also result in significant gains in accuracy for $P_x \subset \mathbb{R}^2$, but for $P_x \subset \mathbb{R}$, it does not show much improvement, and sometimes performs worse. The larger error per points (Figure 5.3, real data) is mainly due to the extra points added without much error reduction. It seems just aggregating does a good enough job for $P_x \subset \mathbb{R}$, but for $P_x \subset \mathbb{R}^2$ more complicated situations arise where this extra step is helpful.

5.4.3 Efficiency of Coresets

To show the efficiency of our methods, we compare the construction time and query time based on the coreset compared to the original data set (denoted Org) for the data set $P_x \subset \mathbb{R}$. For both comparisons, inspired by the Improved Fast Gauss Transform [146] and other fast kernel evaluation methods, for each query point, only the neighbor points within ten bandwidth are queried to calculate the kernel regression values. The construction time includes building the tree structure for the local data query, plus the time to generate the coreset. The query time are based on 128,000 evaluation points.

From Figure 5.4 for $P_x \subset \mathbb{R}$, **Grid**, **G-Aggregate**, **Random Sample**, **Z-order**, and **Z-Aggregate** have the most efficient construction times, roughly as fast as just reading the data. Note that **k-Center** becomes quite slow for large coreset size. Similarly, **Grid**, **G-Aggregate**, and **Random Sample** are very efficient for $P_x \subset \mathbb{R}^2$ (Figure 5.5), but **Z-order** and **Z-Aggregate** have noticeable overhead compared to the grid-based methods (and have no accuracy or analysis advantage). In both settings, there is also considerable time overhead to running **Aggregate-Neighbor**, which has a slight accuracy advantage for $P_x \subset \mathbb{R}^2$ – thus, it is probably only worth it if preprocessing time on these scales are not of much importance but accuracy for $P_x \subset \mathbb{R}^2$ is.

For the query time, all the methods improve at least 2 orders of magnitude over using the original data. Their query times are all about the same; this is as expected since they all produce a coreset of the same size, which can be used as proxy for the full data set in precisely the same way.

- **Main take-away** In conclusion, **G-Aggregate** is the best algorithm in terms of effectiveness and efficiency for $P_x \subset \mathbb{R}$, with **Aggregate-Neighbor** has better accuracy in $P_x \subset \mathbb{R}^2$, but has some increased overhead in construction time. They are orders of magnitude faster than using the original data (and best among all proposed methods)

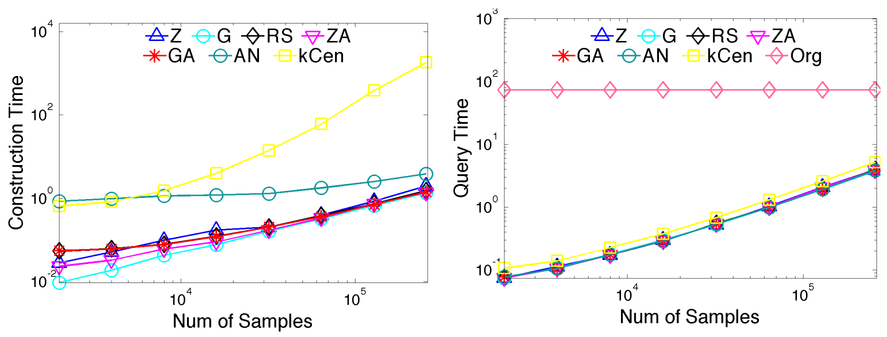


Figure 5.4. Comparison of construction time and query time for real data set with $P_x \subset \mathbb{R}$.

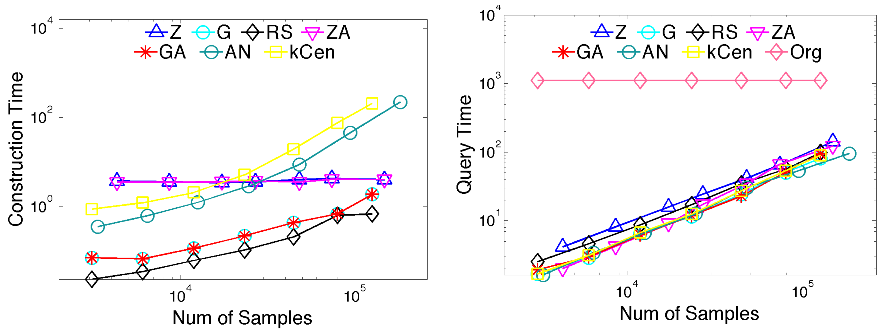


Figure 5.5. Comparison of construction time and query time for real data set with $P_x \subset \mathbb{R}^2$.

and have extremely small error even for small coreset sizes (again the best among all proposed methods). For data sets of size 1 or 2 million, they achieve very small error using only 10,000 points and almost no error around 100,000 points. They (especially **G-Aggregate**) are very simple to implement, and about as fast to construct as reading the data. As we have seen in Section 5.3, we are also able to prove very strong error guarantees for these methods.

5.4.4 Consistency with Bandwidth

In Figure 5.6, we test the consistency of the algorithms by varying the bandwidth. We fix the number of evaluation point as 128,000 and the coreset size 62,499. By varying the bandwidth from 40 to 800 for real data (left) of $P_x \subset \mathbb{R}$, 10 to 160 for synthetic data (middle) for $P_x \subset \mathbb{R}$ and real data (right) for $P_x \subset \mathbb{R}^2$, the errors are decreasing for all the methods. This matches with our analysis in Section 5.3 and aligns with the notion that the more we smooth the data, the more stable it is, and the fewer data points we

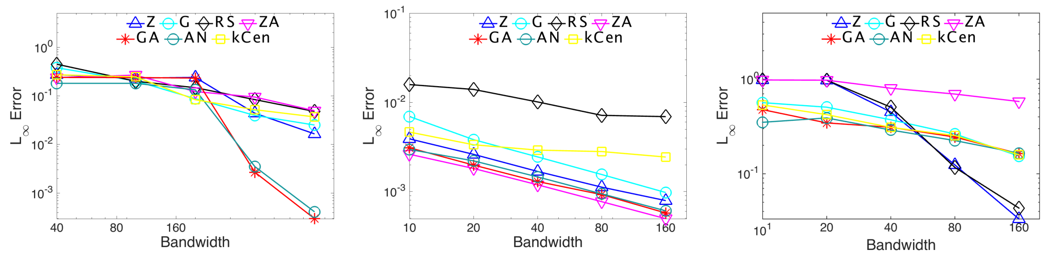


Figure 5.6. The relation between L_∞ error and bandwidth for real data (left), synthetic data (middle) for $P_x \subset \mathbb{R}$ and real data (right) for $P_x \subset \mathbb{R}^2$.

actually need. Again, **G-Aggregate** consistently performs the best or among the best of our methods. The exception is the real data in $\mathbb{R}^2$ (right), where for very large bandwidths, the simple methods **Z-order** and **Random Sample** dominate. In this setting, these data are so smoothed that these methods exactly or roughly amount to a random sample, and work better than trying to fit gridded data to circularly smoothed estimates. Note, we do not attempt to automatically choose the bandwidth, as this should be a choice of the user to determine the scale they examine the data [150].

5.4.5 Progressive Grid-Based Approaches

We evaluate the progressive grid-based approach on the CloudLab data. The total coreset size is 316,485, using **G-Aggregate** in each region. And we use 256,000 evaluation points. We evaluate the algorithm at four smoothing choices $\sigma = \{10, 15, 30, 45\}$. For each σ, we gradually increase the window size T, starting at 1 day (86,400), up to 10 months ($2.5 \cdot 10^7$), as shown in Figure 5.7. We see that as a new region is reached, and the grid size enlarges, then so does the error. Also, as long as T/σ is bounded by $4 \cdot 10^4$, the error stays under 0.01.

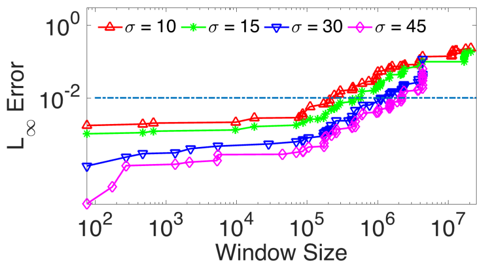


Figure 5.7. The relation between window size and L_∞ error for progressive G-Aggregate method.

5.4.6 High Dimension

For simplicity, we only compare **G-Aggregate** and **Random Sampling**. When increasing the grid size, the number of empty grid increases as well, so we observe that the size of coreset does not increase exponentially. By increasing the number of grids cells from 10 to 20, that is a factor 2 in each dimension, the average number of nonempty grids for CAD data set are $\{902, 1367, 1863, 2538, 3150, 3791\}$ and for CASP data set are $\{1034, 1554, 2122, 2742, 3543, 4342\}$. The relationship of L_∞ error and coreset size is shown in the Figure 5.8(left), using bandwidths 3 and 3.8, respectively. The error decreases when the size of coreset increases for both methods. For the same coreset size, **Random Sampling** perform better than **G-Aggregate**, and its running time (right figure in Figure 5.8) is much less. This aligns with our theoretical bounds. For example, for grid size 20^6, the **G-Aggregate** method takes about 250s, while the **Random Sampling** takes only around 3s. So we recommend the simple and fast method **Random Sampling** to generate coreset for kernel regression for high dimension data sets.

5.5 Conclusion

We describe several algorithms for coresets for kernel regression. Many (random sampling, order-based thinning, and grid-based thinning) are common heuristics. As we demonstrate on data sets with millions of points, those based on grids work much better, and that small modification of aggregating and sometimes filling in sparse-neighborhood boundaries can make large difference in error reduction. With our best methods, massive data sets can be drastically reduced in size and have negligible error.

Find our code and data at: `http://www.cs.utah.edu/~yanzheng/kernel-reg/`.

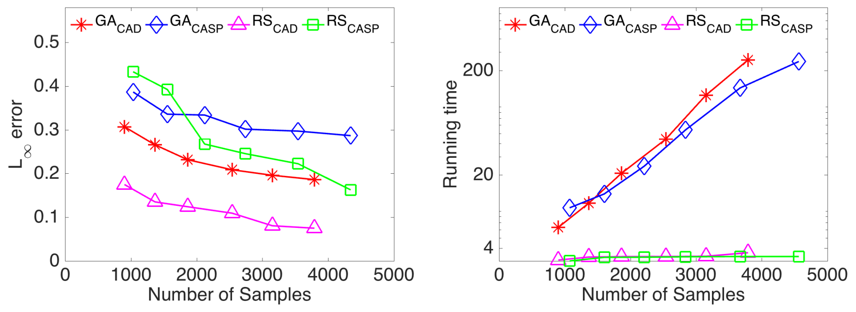


Figure 5.8. Left: L_∞ error for coresets of high dimensional data sets. Right: Running time to generate the coresets.

CHAPTER 6

CONCLUSION

The rate at which scientists and businesses are producing data is increasing at an unstoppable rate. The advancement of science and industry becomes heavily dependent on understanding these data sets. Kernel smoothing provides a simple way of finding structures in data sets without the imposition of a parametric model, in this dissertation, we consider kernel regression and kernel density estimates. However, the enormity of the data size precludes brute force approaches of analyzing it. Thus, data summarization is an important tool for dealing with massive data. Coresets enable accurate query answering while requiring much lower resources, and can be much faster. In Chapter 2 and Chapter 5, we have demonstrated that the effectiveness of coresets in data size reduction, and thus reduce the computation time of kernel density estimates and kernel regression significantly.

As we have seen an example of applying KDE coreset in topological data analysis, there are numerous other challenges to apply KDE coreset. For example, in [107], we propose a new definition of ε-net under kernels, called (ε, τ)-*net* of kernel range space. It tries to answer questions of what is the sample size we need to maintain to include a significant witness for every large enough event of noisy spatial data points, for example, twitter users with geo-coordinates. Putting this result to visualization purpose, we get a very quick way to visualize the large twitter data without looking at the whole data set: we only need one witness point for each large enough event, and do not need to care about the small events or outlier. By tuning the parameter ε and τ, we can control how much outlier we want to filter out. This method gives users a freedom to decide how much outlier we want to filter out and only keep visually curious regions.

Another challenge is applying kernel smoothing technique into other applications. For instance, spatial scan statistics [83], as a effective anomaly detection method, computes the maximum discrepancy region obtained by scanning the spatial region under study with a set of circular regions of various radii. The discrepancy score for each region is based on a likelihood ratio test statistic constructed to detect significant over density under the Poisson or Bernoulli model. Putting kernel in the classic spatial scan statistics setting, the circular regions or other hard boundary geometric regions turn to be smooth ranges, the value of each point in each range relates how close it is to the kernel center, and thus statistically robust to spatial noise.

As discussed in [36], kernel density estimates is a kernel mean that estimates the density of the data. The kernel mean embedding (KME) is another form of kernel mean that maps the probability distribution into a reproducing kernel Hilbert space. Since some of the kernels discussed in this dissertation, for example, Gaussian kernel and Laplacian kernel, are also symmetric positive definite kernels, our methods can apply to both kernel means, which have many applications in machine learning. For example, anomaly detection [39] and mean-shift clustering [32] for KDE, kernel two sample test [59] and support measure machines [94] for KME.

In summary, this dissertation provides a contribution to the problem of creating coresets for kernel smoothing with approximation guarantees. There is still plenty of work left to be done in this area in order to be able to understand the kernel smoothing and how to use it in various applications. This is becoming a central problem to large noisy data analysis, and the techniques presented herein will hopefully be used as building blocks for future progress in this direction.

APPENDIX

PROOFS FOR CORESET ANALYSIS

A.1 Linking and (ρ, ε)-Approximations for Kernel Regression

Theorem 17. *For any kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ linked to a range space ($\mathbb{R}^d, \mathcal{A}$), a (ρ, ε)-approximation S of $(P, \mathcal{A})$ for $S \subset \mathbb{R}^d$ is a $(\rho K^+, 2\varepsilon)$-approximation of (P, K), where $K^+ = \max_{p,q \in P} K(p, q)$.*

Proof. We first give the definition of κ. For two point sets P, S, define a *similarity* between the two point sets as

$$\kappa(P, S) = \frac{1}{|P|}\frac{1}{|S|}\sum_{p \in P}\sum_{s \in S} K(p, s),$$

and when the pointset S only contains one single point s and a subset $P' \subset P$, we have $\kappa_P(P', s) = (1/|P|)\sum_{p \in P'} K(p, s)$.

Then we follow the same technique in proof of Theorem 5.1 in [78], suppose q is any query point, we can sort all $p_i \in P$ in similarity to q so that $p_i < p_j$ (and by notation $i < j$) if $K(p_i, q) > K(p_j, q)$. Thus any super-level set containing p_j also contains p_i for $i < j$. We can now consider the one-dimensional problem on this sorted order from q.

We now count the deviation $D(P, S, q) = \text{KDE}_P(q) - \text{KDE}_S(q)$ from p_1 to p_n using a charging scheme. That is each element $s_j \in S$ is charged to $g = |P|/|S|$ points in P. For simplicity we will assume that g is an integer, otherwise we can allow fractional charges. We now construct a partition of P slightly differently, for positive and negative $D(P, S, q)$ values, corresponding to undercounts and overcounts, respectively.

- **Undercount of $\text{KDE}_S(q)$:** For undercounts, we partition P into $2|S|$ sets $\{P'_1, P_1, P'_2, P_2, ..., P'_{|S|}, P_{|S|}\}$ of consecutive points by the sorted order from q. Starting with p_1, we place points in set P'_j or P_j following their sorted order. Recursively on j and i, starting at $j = 1$ and $i = 1$, we place each p_i in P'_j as long as $K(p_i, q) > K(s_j, q)$(this may be empty). Then we place the next g points p_i into P_j. After g points are placed in P_j, we begin with P'_{j+1}, until all of P has been placed in some set. Let $t \leq |S|$ be the index of the last set P_j

such that $|P_j| = g$. Note that for all $p_i \in P_j$(for $j \leq t$) we have $K(s_j, q) \geq K(p_i, q)$, thus $\kappa_S(\{s_j\}, q) \geq \kappa_P(P_j, q)$. We can now bound the undercount as

$$\begin{aligned} D(P,S,q) &= \sum_{j=1}^{|S|} \left(\kappa_P(P_j,q)\right) - \kappa_S(\{s_j\},q) + \sum_{j=1}^{|S|} \kappa_P(P'_j,q) \\ &\leq \sum_{j=1}^{t+1} \kappa_P(P'_j,q) \end{aligned}$$

since the first term is at most 0 and $|P'_j| = 0$ for $j > t+1$. Now consider a super-level set $H \in \mathcal{A}$ containing all points before s_{t+1}; H is the smallest range that contains every non-empty P'_j. Because (for $j \leq t$) each set P_j can be charged to s_j, then $\sum_{j=1}^{t} |P_j \cap H| = g|S \cap H|$. And because S is an (ρ, ε)-approximation of $(P, \mathcal{A})$, then

$$\frac{|P \cap H|}{|P|} - \frac{|S \cap H|}{|S|} \leq \varepsilon \max\left\{\frac{|P \cap H|}{|P|}, \rho\right\}$$

$$\begin{aligned} \frac{1}{|P|}\sum_{j=1}^{t+1} |P'_j| &= \frac{1}{|P|}\sum_{j=1}^{t+1} |P'_j \cap H| \\ &= \frac{1}{|P|}\Big(\sum_{j=1}^{t+1} |P'_j \cap H| + \sum_{j=1}^{t} |P_j \cap H| - g|S \cap H|\Big) \\ &= \frac{|P \cap H|}{|P|} - \frac{|S \cap H|}{|S|} \leq \varepsilon \max\left\{\frac{|P \cap H|}{|P|}, \rho\right\} \end{aligned}$$

We can now bound

$$D(P,S,q) \leq \sum_{j=1}^{t+1} \kappa_P(P'_j,q) = \sum_{j=1}^{t+1} \sum_{p \in P'_j} \frac{K(p,q)}{|P|}$$

When $\frac{|P \cap H|}{|P|} \geq \rho$, and $\rho \geq \sum_{p \in P'_1} \frac{K(p,q)}{|P|} \Big/ \varepsilon$,

$$\begin{aligned} D(P,S,q) &\leq \sum_{j=1}^{t+1} \sum_{p \in P'_j} \frac{K(p,q)}{|P|} \\ &\leq \frac{\varepsilon}{|P|} \sum_{p \in P} K(p,q) + \sum_{p \in P'_1} \frac{K(p,q)}{|P|} \\ &\leq \varepsilon \text{KDE}_P(q) + \varepsilon\rho \leq 2\varepsilon \max\{\text{KDE}_P(q), \rho\}. \end{aligned}$$

The second inequality is because, all the points in P_j has larger $K(\cdot, q)$ values than the points in P'_{j+1} and $\frac{1}{|P|}\sum_{j=1}^{t+1} |P'_j| \leq \varepsilon \frac{|P \cap H|}{|P|}$.

When $\frac{|P \cap H|}{|P|} \leq \rho$,

$$\begin{aligned} D(P,S,q) &\leq \sum_{j=1}^{t+1} \sum_{p \in P'_j} \frac{K(p,q)}{|P|} \\ &\leq \frac{1}{|P|} \sum_{j=1}^{t+1} |P'_j| K^+ \leq \varepsilon \rho K^+. \end{aligned}$$

- **Overcount of $\text{KDE}_S(q)$:** The analysis for overcounts is similar to undercounts, but we partition the data in a reverse way: we partition P into $2|S|$ sets $\{P_1, P'_1, P_2, P'_2, ..., P_{|S|}, P'_{|S|}\}$ of consecutive points by the sorted order from q (some of the sets may be empty). Starting with p_n (the furthest point from q) we place points in sets P'_j or P_j following their reverse-sorted order. Recursively on j and i, starting at $j = |S|$ and $i = n$, we place each p_i in P'_j as long as $K(p_i, q) < K(s_j, q)$ (this may be empty). Then we place the next g points p_i into P_j. After g points are placed in P_j, we begin with P'_{j-1}, until all of P has been placed in some set. Let $t \leq |S|$ be the index of the last set P_j such that $|P_j| = g$ (the smallest such j). Note that for all $p_i \in P_j$ (for $j \geq t$) we have $K(s_j, q) \leq K(p_i, q)$, thus $\kappa_S(\{s_j\}, q) \leq \kappa_P(P_j, q)$. We can now bound the (negative) overcount as

$$\begin{aligned} D(P,S,q) = &\sum_{j=|S|}^{t} \left(\kappa_P(P_j,q)\right) - \kappa_S(\{s_j\},q) \\ &+ \sum_{j=t-1}^{1} \left(\kappa_P(P_j,q)\right) - \kappa_S(\{s_j\},q) + \sum_{j=1}^{|S|} \kappa_P(P'_j,q) \\ &\geq \kappa_P(P_{t-1},q) - \sum_{j=t-1}^{1} \kappa_S(\{s_j\},q) \end{aligned}$$

since the first full term is at least 0, as is each $\kappa_P(P_j, q)$ and $\kappa_P(P'_j, q)$ term in the second and third terms. We will need the one term $\kappa_P(P_{t-1}, q)$ related to P.

Now using that S is an (ρ, ε)-sample of $(P, \mathcal{A})$, we will derive a bound on t. We consider the maximal super-level set $H \in \mathcal{A}$ such that no points $H \in P$ are in P'_j for any j. This is the largest set where each point $p \in P$ can be charged to a point $s \in S$ such that $K(p,q) > K(s,q)$, and thus presents the smallest (negative) overcount. In this case, $H \cap P = \cap_{j=1}^{w} P_j$ for some w and $H \cap S = \cap_{j=1}^{w} \{s_j\}$. Since $t \leq w$, then $|H \cap P| = (w - t + 1)g + |P_{t-1}| = (w - t + 1)|P|/|S| + |P_{t-1}|$ and $|H \cap S| = w$. With the definition of (ρ, ε)-sample,

$$\frac{|S \cap H|}{|S|} - \frac{|P \cap H|}{|P|} \leq \varepsilon \max\left\{\frac{|P \cap H|}{|P|}, \rho\right\}$$

$$\begin{aligned}
&\frac{|S \cap H|}{|S|} - \frac{|P \cap H|}{|P|} \\
&= \frac{w}{|S|} - \frac{(w-t+1)|P|/|S|}{|P|} - \frac{|P_{t-1}|}{|P|} \\
&\geq \frac{t-1}{|S|} - \frac{|P_{t-1}|}{|P|}
\end{aligned}$$

If $\frac{|P \cap H|}{|P|} \geq \rho$, $\frac{t-1}{|S|} - \frac{|P_{t-1}|}{|P|} \leq \varepsilon \frac{|P \cap H|}{|P|}$, the same as $\frac{|P_{t-1}|}{|P|} - \frac{t-1}{|S|} \geq -\varepsilon \frac{|P \cap H|}{|P|}$, then

$$\begin{aligned}
D(P, S, q) &\geq \kappa_P(P_{t-1}, q) - \sum_{j=t-1}^{1} \kappa_S(\{s_j\}, q) \\
&= \frac{\kappa(P_{t-1}, q)}{|P|} - \frac{\sum_{j=t-1}^{1} K(s_j, q)}{|S|} \\
&\geq -\varepsilon \frac{\sum_{j=w}^{1} \kappa(P_j, q)}{|P|} \geq -\varepsilon \mathrm{KDE}_P(q)
\end{aligned}$$

The second inequality is because $\frac{|P_{t-1}|}{|P|} - \frac{t-1}{|S|} \geq -\varepsilon \frac{|P \cap H|}{|P|}$ and for each s_j with $j \leq w$, for any $p \in P_j$, $K(s_j, q) \leq K(p, q)$.

If $\frac{|P \cap H|}{|P|} \leq \rho$, $\frac{t-1}{|S|} - \frac{|P_{t-1}|}{|P|} \leq \varepsilon\rho$, the same as $t - 2 \leq \varepsilon\rho|S| + \frac{|S||P_{t-1}|}{|P|} - 1$. Letting $p_i = \min_{i' \in P_{t-1}} K(p'_i, q)$

$$\begin{aligned}
&D(P, S, q) \\
&\geq \kappa_P(P_{t-1}, q) - \kappa_S(\{s_{t-1}\}, q) + \sum_{j=t-2}^{1} \kappa_S(\{s_j\}, q) \\
&= \frac{\kappa(P_{t-1}, q)}{|P|} - \frac{K(s_{t-1}, q)}{|S|} - (\varepsilon\rho|S| + \frac{|S||P_{t-1}|}{|P|} - 1)\frac{K^+}{|S|} \\
&\geq -\varepsilon\rho K^+ + K^+\Big(\frac{g - |P_{t-1}|}{|P|}\Big) - \frac{g \cdot K(s_{t-1}, q) - \kappa(P_{t-1}, q)}{|P|} \\
&\geq -\varepsilon\rho K^+ + K^+\Big(\frac{g - |P_{t-1}|}{|P|}\Big) - K(p_i, q)\Big(\frac{g - |P_{t-1}|}{|P|}\Big) \\
&\geq -\varepsilon\rho K^+
\end{aligned}$$

So when $\frac{|P \cap H|}{|P|} \geq \rho$, S is an $(\rho, 2\varepsilon)$- approximation of $(P, \mathcal{K})$, and when $\frac{|P \cap H|}{|P|} \leq \rho$, it is a $(\varepsilon\rho K^+)$-approximation of $(P, \mathcal{K})$. □

REFERENCES

[1] P. K. Agarwal, S. Har-Peled, H. Kaplan, and M. Sharir, *Union of random minkowski sums and network vulnerability analysis*, in Proceedings of 29th Symposium on Computational Geometry, 2013.

[2] A. F. Alessandro Bergamo, Lorenzo Torresani, *Picodes: Learning a compact code for novel-category recognition*, in NIPS, 2011.

[3] N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler, *Scale-sensitive dimensions, uniform convergence, and learnability*, Journal of ACM, 44 (1997), pp. 615–631.

[4] A. W. Appel, *An efficient program for many-body simulation*, SIAM J.Scientific and Statistical Computing, 6 (1985), pp. 85–103.

[5] B. Aronov, E. Ezra, and M. Sharir, *Small size ε-nets for axis-parallel rectangles and boxes*, Siam Journal of Computing, 39 (2010), pp. 3248–3282.

[6] N. Aronszajn, *Theory of reproducing kernels*, Transactions of the American Mathematical Society, 68 (1950), pp. 337–404.

[7] F. Bach, S. Lacoste-Julien, and G. Obozinski, *On the equivalence between herding and conditional gradient algorithms*, arXiv preprint arXiv:1203.4523, (2012).

[8] A. Bagchi, A. Chaudhary, D. Eppstein, and M. T. Goodrich, *Deterministic sampling and range counting in geometric data streams*, ACM Transactions on Algorithms, 3 (2007), p. 16.

[9] S. Balakrishnan, B. T. Fasy, F. Lecci, A. Rinaldo, A. Singh, and L. Wasserman, *Statistical inference for persistent homology*, tech. rep., ArXiv:1303.7117, March 2013.

[10] J. Beck, *Irregularities of distribution I*, Acta Mathematica, 159 (1987), pp. 1–49.

[11] M. Bern and D. Eppstein, *Worst-case bounds for subadditive geometric graphs*, Proceedings 9th Symposium on Computational Geometry, (1993).

[12] R. Blundell and A. Duncan, *Kernel regression in empirical microeconomics*, Journal of Human Resources, (1998), pp. 62–87.

[13] T. Bouezmarni and J. V. Rombouts, *Nonparametric density estimation for multivariate bounded data*, J. Statistical Planning and Inference, 140 (2010), pp. 139–152.

[14] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, *Near-optimal coresets for least-squares regression*, IEEE Trans. Information Theory, 59 (2013).

[15] A. W. Bowman, *An alternative method of cross-validation for the smoothing of density estimates*, Biometrika, 71 (1984), pp. 353–360.

[16] M. J. Brewer, *A bayesian model for local smoothing in kernel density estimation*, Statistics and Computing, 10 (2000), pp. 299–309.

[17] T. Brox, B. Rosenhahn, D. Cremers, and H.-P. Seidel, *Nonparametric density estimation with adaptive, anisotropic kernels for human motion tracking*, in Human Motion–Understanding, Modeling, Capture and Animation, Springer, 2007, pp. 152–165.

[18] J. D. Brutlag, *Aberrant behavior detection in time series for network monitoring*, in System Administration Conference (LISA), USENIX, 2000.

[19] P. B. Callahan and S. R. Kosaraju, *Algorithms for dynamic closest-pair and n-body potential fields*, in SODA, 1995.

[20] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, *Hierarchical density estimates for data clustering, visualization, and outlier detection*, ACM Transactions on Knowledge Discovery from Data (TKDD), 10 (2015), p. 5.

[21] Y. Cao, H. He, H. Man, and X. Shen, *Integration of self-organizing map (SOM) and kernel density estimation (KDE) for network intrusion detection*, in SPIE Europe Security+ Defence, 2009.

[22] S. Chan, I. Diakonikolas, R. A. Servedio, and X. Sun, *Efficient density estimation via piecewise polynomial approximation*, in STOC, 2014.

[23] F. Chazal, D. Cohen-Steiner, and A. Lieutier, *Normal cone approximation and offset shape isotopy*, Computational Geometry: Theory and Applications, 42 (2009), pp. 566–581.

[24] F. Chazal, D. Cohen-Steiner, and A. Lieutier, *A sampling theory for compact sets in Euclidean space*, Discrete Computational Geometry, 41 (2009), pp. 461–479.

[25] F. Chazal, D. Cohen-Steiner, and Q. Mérigot, *Geometric inference for probability measures*, Foundations of Computational Mathematics, 11 (2011), pp. 733–751.

[26] F. Chazal and A. Lieutier, *Weak feature size and persistent homology: computing homology of solids in rn from noisy data samples*, Proceedings 21st Annual Symposium on Computational Geometry, (2005), pp. 255–262.

[27] F. Chazal and A. Lieutier, *Topology guaranteeing manifold reconstruction using distance function to noisy data*, Proceedings of the 22nd Annual Symposium on Computational Geometry, (2006), pp. 112–118.

[28] B. Chazelle, *The Discrepancy Method*, Cambridge, 2000.

[29] B. Chazelle and J. Matousek, *On linear-time deterministic algorithms for optimization problems in fixed dimensions*, J. Algorithms, 21 (1996), pp. 579–597.

[30] Y. Chen, M. Welling, and A. Smola, *Super-samples from kernel hearding*, in Conference on Uncertainty in Artificial Intellegence, 2010.

[31] Y. Chen, M. Welling, and A. Smola, *Super-samples from kernel herding*, arXiv preprint arXiv:1203.3472, (2012).

[32] Y. Cheng, *Mean shift, mode seeking, and clustering*, IEEE transactions on pattern analysis and machine intelligence, 17 (1995), pp. 790–799.

[33] K. L. Clarkson, *Coresets, sparse greedy approximation, and the frank-wolfe algorithm*, ACM Transactions on Algorithms (TALG), 6 (2010), p. 63.

[34] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava., *Space- and time-efficient deterministic algorithms for biased quantiles over data streams*, in PODS, 2006.

[35] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang, *Optimal sampling from distributed streams*, in PODS, 2010.

[36] E. C. Cortés and C. Scott, *Sparse approximation of a kernel mean*, IEEE Transactions on Signal Processing, (2016).

[37] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney, *Sampling algorithms and coresets for ℓ_p regression*, SICOMP, 38 (2009), pp. 2060–2078.

[38] M. S. de Lima and G. S. Atuncar, *A bayesian method to estimate the optimal bandwidth for multivariate kernel estimator*, Journal of Nonparametric Statistics, 23 (2011), pp. 137–148.

[39] M. Desforges, P. Jacob, and J. Cooper, *Applications of probability density estimation to the detection of abnormal conditions in engineering*, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 212 (1998), pp. 687–703.

[40] L. Devroye and L. Györfi, *Nonparametric Density Estimation: The L_1 View*, Wiley, 1984.

[41] L. Devroye and G. Lugosi, *Combinatorial Methods in Density Estimation*, Springer-Verlag, 2001.

[42] R. Duan, S. Pettie, and H.-H. Su, *Scaling algorithms for approximate and exact maximum weight matching*, tech. rep., arXiv:1112.0790, 2011.

[43] D. P. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*, Cambridge, 2009.

[44] T. Duong et al., *ks: Kernel density estimation and kernel discriminant analysis for multivariate data in r*, Journal of Statistical Software, 21 (2007), pp. 1–16.

[45] T. Duong and M. L. Hazelton, *Cross-validation bandwidth matrices for multivariate kernel density estimation*, Scandinavian J. of Stat., 32 (2005), pp. 485–506.

[46] H. Edelsbrunner, M. Facello, P. Fu, and J. Liang, *Measuring proteins and voids in proteins*, in Proceedings 28th Annual Hawaii International Conference on Systems Science, 1995.

[47] H. Edelsbrunner, B. T. Fasy, and G. Rote, *Add isotropic Gaussian kernels at own risk: More and more resiliant modes in higher dimensions*, Proceedings 28th Annual Symposium on Computational Geometry, (2012), pp. 91–100.

[48] J. Edmonds, *Paths, trees, and flowers*, Canadian Journal of Mathematics, 17 (1965), pp. 449–467.

[49] B. T. Fasy, J. Kim, F. Lecci, and C. Maria, *Introduction to the R package TDA*, tech. rep., arXiV:1411.1830, 2014.

[50] A. Gangopadhyay and K. Cheung, *Bayesian approach to choice of smoothing parameter in kernel density estimation*, J. of Nonparam. Stat., 14 (2002), pp. 655–664.

[51] M. Gao, C. Chen, S. Zhang, Z. Qian, D. Metaxas, and L. Axel, *Segmenting the papillary muscles and the trabeculae from high resolution cardiac CT through restoration of topological handles*, in Proceedings of the International Conference on Information Processing in Medical Imaging, 2013.

[52] J. Glaunès, *Transport par difféomorphismes de points, de mesures et de courants pour la comparaison de formes et l'anatomie numérique.*, PhD thesis, Université Paris 13, 2005.

[53] T. F. Gonzalez, *Clustering to minimize the maximum intercluster distance*, Theoretical Computer Science, 38 (1985), pp. 293–306.

[54] S. Govindarajan, P. K. Agarwal, and L. Arge, *CRB-tree: An efficient indexing scheme for range-aggregate queries*, in ICDT, 2003, pp. 143–157.

[55] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, *Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total*, in ICDE, 1996.

[56] L. Greengard and J. Strain, *The fast Gauss transform*, Journal Scientific and Statistical Computing, 12 (1991).

[57] M. Greenwald and S. Khanna, *Space-efficient online computation of quantile summaries*, in SIGMOD, 2001.

[58] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, *A kernel two-sample test*, Journal of Machine Learning Research, 13 (2012), pp. 723–773.

[59] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, *A kernel two-sample test*, Journal of Machine Learning Research, 13 (2012), pp. 723–773.

[60] S. Guha, N. Koudas, and K. Shim, *Approximation and streaming algorithms for histogram construction problems*, ACM TODS, 31 (2006), pp. 396–438.

[61] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi, *Approximating multi-dimensional aggregate range queries over real attributes*, in SIGMOD, 2000, pp. 463–474.

[62] J. Habbema, J. Hermans, and K. van den Broek, *A stepwise discrimination analysis program using density estimation*, Proceedings in Computational Statistics, (1974).

[63] P. Hall, J. Marron, and B. U. Park, *Smoothed cross-validation*, Prob. The. and Rel. Fields, 92 (1992), pp. 1–20.

[64] P. Hall and M. P. Wand, *Minimizing L_1 distance in nonparametric density estimation*, Journal of Multivariate Analysis, 26 (1988), pp. 59–88.

[65] S. Har-Peled, *Geometric Approximation Algorithms*, Chapter 2, American Mathematical Society, 2011.

[66] S. Har-Peled, *Geometric approximation algorithms*, vol. 173, American mathematical society Boston, 2011.

[67] S. Har-Peled, H. Kaplan, M. Sharir, and S. Smorodinksy, *ε-nets for halfspaces revisited*, tech. rep., arXiv:1410.3154, 2014.

[68] N. Harvey and S. Samadi, *Near-optimal herding.*, in COLT, 2014, pp. 1165–1182.

[69] D. Haussler and E. Welzl, *epsilon-nets and simplex range queries.*, Disc. & Comp. Geom., 2 (1987), pp. 127–151.

[70] M. Hein and O. Bousquet, *Hilbertian metrics and positive definite kernels on probability measures*, in Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, 2005.

[71] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant, *Range queries in OLAP data cubes*, in SIGMOD, 1997.

[72] J. Horel, M. Splitt, L. Dunn, J. Pechmann, B. White, C. Ciliberti, S. Lazarus, J. Slemmer, D. Zaff, and J. Burks, *Mesowest: Cooperative mesonets in the western united states*, Bulletin of the American Meteorological Society, 83 (2002), pp. 211–225.

[73] S. Hu, D. S. Poskitt, and X. Zhang, *Bayesian adaptive bandwidth kernel density estimation of irregular multivariate distributions*, CS&DA, 56 (2012), pp. 732–740.

[74] Z. Huang, L. Wang, K. Yi, and Y. Liu, *Sampling based algorithms for quantile computation in sensor networks*, in SIGMOD, 2011, pp. 745–756.

[75] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel, *Optimal histograms with quality guarantees*, in VLDB, 1998.

[76] M. Jones and S. Sheather, *Using non-stochastic terms to advantage in kernel-based estimation of integrated squared density derivatives*, Statistics & Probability Letters, 11 (1991), pp. 511–514.

[77] M. C. Jones, J. S. Marron, and S. J. Sheather, *A brief survey of bandwidth selection for density esimation*, American Statistical Association, 91 (1996), pp. 401–407.

[78] S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian, *Comparing distributions and shapes using the kernel distance*, in Proceedings of the twenty-seventh annual symposium on Computational geometry, ACM, 2011, pp. 47–56.

[79] J. Kiefer, *Sequential minimax search for a maximum*, Proc. Am. Mathematical Society, 4 (1953), pp. 502–506.

[80] V. Kolmogorov, *BLOSSOM V: A new implementation of a minimum cost perfect matching algorithm*, Mathematical Programming, 1 (2009).

[81] N. Koudas, S. Muthukrishnan, and D. Srivastava, *Optimal histograms for hierarchical range queries*, in PODS, 2000.

[82] K. Kulasekera and W. Padgett, *Bayes bandwidth selection in kernel density estimation with censored data*, Nonparametric Statistics, 18 (2006), pp. 129–143.

[83] M. Kulldorff, *A spatial scan statistic*, Communications in Statistics-Theory and Methods, 26 (1997), pp. 1481–1496.

[84] K. G. Larsen, *On range searching in the group model and combinatorial discrepancy*, in Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science, 2011.

[85] Y. Li, P. M. Long, and A. Srinivasan, *Improved bounds on the sample complexity of learning*, Journal of Computer and System Sciences, 62 (2001), pp. 516–527.

[86] J. Liang, H. Edelsbrunner, P. Fu, P. V. Sudharkar, and S. Subramanian, *Analytic shape computation of macromolecues: I. molecular area and volume through alpha shape*, Proteins: Structure, Function, and Genetics, 33 (1998), pp. 1–17.

[87] X. Lin, J. Xu, Q. Zhang, H. Lu, J. X. Yu, X. Zhou, and Y. Yuan, *Approximate processing of massive continuous quantile queries over high-speed data streams*, TKDE, 18 (2006), pp. 683–698.

[88] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, *Approximate medians and other quantiles in one pass and with limited memory*, in SIGMOD, 1998, pp. 426–435.

[89] J. Marron and A. Tsybakov, *Visual error criteria for qualitative smoothing*, Journal of the American Statistical Association, 90 (1995), pp. 499–507.

[90] J. Matoušek, *Approximations and optimal geometric divide-and-conquer*, in STOC, 1991.

[91] J. Matoušek, *Tight upper bounds for the discrepancy of halfspaces.*, Discrete & Computational Geometry, 13 (1995), pp. 593–601.

[92] J. Matoušek, *Geometric Discrepancy*, Springer, 1999.

[93] J. Matoušek, R. Seidel, and E. Welzl, *How to net a lot with little: Small ε-nets for disks and halfspaces*, in Proceedings of the 6th Annual Symposium on Computational Geometry, 1990.

[94] K. Muandet, K. Fukumizu, F. Dinuzzo, and B. Schölkopf, *Learning from distributions via support measure machines*, in Proceedings of Advances in Neural Information Processing Systems, 2012, pp. 10–18.

[95] A. Müller, *Integral probability metrics and their generating classes of functions*, Advances in Applied Probability, 29 (1997), pp. 429–443.

[96] E. A. Nadaraya, *On estimating regression*, Theory of Probability and its Applications, 9 (1964), pp. 141–142.

[97] R. K. Pace and R. Barry, *Sparse spatial autoregressions*, Statistics & Probability Letters, 33 (1997), pp. 291–297.

[98] J. Pach and P. K. Agarwal, *Combinatorial geometry.*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, 1995.

[99] J. Pach and G. Tardos, *Tight lower bounds for the size of epsilon-nets*, Journal of American Mathematical Society, 26 (2013), pp. 645–658.

[100] E. Parzen, *On estimation of a probability density function and mode*, Annals of Mathematical Statistics, 33 (1962), pp. 1065–1076.

[101] E. Parzen, *Probability density functionals and reproducing kernel hilbert spaces*, in Proceedings of the Symposium on Time Series Analysis, vol. 196, Wiley, New York, 1963, pp. 155–169.

[102] J. M. Phillips, *Algorithms for ε-approximations of terrains*, in Automata, Languages and Programming, Springer, 2008, pp. 447–458.

[103] J. M. Phillips, *ε-samples for kernels*, in Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, 2013, pp. 1622–1632.

[104] J. M. Phillips, *Coresets and Sketches*, CRC Press, 3rd ed., 2016.

[105] J. M. Phillips and S. Venkatasubramanian, *A gentle introduction to the kernel distance*. arXiv:1103.1625, March 2011.

[106] J. M. Phillips, B. Wang, and Y. Zheng, *Geometric inference on kernel density estimates*, arXiv preprint arXiv:1307.7760 (SoCG 2015), (2013).

[107] J. M. Phillips and Y. Zheng, *Subsampling in smoothed range spaces*, in Proceedings of the 26th International Conference, on Algorithmic Learning Theory, ALT 2015, 2015, pp. 224–238.

[108] D. Pollard, *Emperical Processes: Theory and Applications*, NSF-CBMS REgional Confernece Series in Probability and Statistics, 1990.

[109] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita, *Improved histograms for selectivity estimation of range predicates*, in SIGMOD, 1996, pp. 294–305.

[110] C. A. Price, O. Symonova, Y. Mileyko, T. Hilley, and J. W. Weitz, *Leaf gui: Segmenting and analyzing the structure of leaf veins and areoles*, Plant Physiology, 155 (2011), pp. 236–245.

[111] E. Pyrga and S. Ray, *New existence proofs ε-nets*, in Proceedings 24th Annual Symposium on Computational Geometry, 2008.

[112] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[113] V. C. Raykar and R. Duraiswami, *Fast optimal bandwidth selection for kernel density estimation*, in SDM, 2006.

[114] V. C. Raykar, R. Duraiswami, and L. H. Zhao, *Fast computation of kernel estimators*, J. of Computational and Graphical Statistics, 19 (2010), pp. 205–220.

[115] R. Ricci, E. Eide, and C. Team, *Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications*, ; login:: the magazine of USENIX & SAGE, 39 (2014), pp. 36–38.

[116] M. Rosenblatt et al., *Remarks on some nonparametric estimates of a density function*, The Annals of Mathematical Statistics, 27 (1956), pp. 832–837.

[117] M. Rudemo, *Empirical choice of histograms and kernel density estimators*, Scandinavian Journal of Statistics, (1982), pp. 65–78.

[118] S. R. Sain, K. A. Baggerly, and D. W. Scott, *Cross-validation of multivariate densities*, Journal of the American Statistical Association, 89 (1994), pp. 807–817.

[119] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.

[120] E. Schubert, A. Zimek, and H.-P. Kriegel, *Generalized outlier detection with flexible kernel density estimates*, in Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA, 2014, pp. 542–550.

[121] D. Scott, R. Tapia, and J. Thompson, *Kernel density estimation revisited,*, Nonlinear Analysis, Theory, Methods and Appplication, 1 (1977), pp. 339–372.

[122] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*, Wiley, 1992.

[123] D. W. Scott and G. R. Terrell, *Biased and unbiased cross-validation in density estimation*, J. ASA, 82 (1987), pp. 1131–1146.

[124] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, *Predicting solar generation from weather forecasts using machine learning*, in SmartGridComm, 2011.

[125] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[126] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman & Hall/CRC, 1986.

[127] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. G. Lanckriet, *Hilbert space embeddings and metrics on probability measures*, Journal of Machine Learning Research, 11 (2010), pp. 1517–1561.

[128] S. Suri, C. D. Tóth, and Y. Zhou, *Range counting over multidimensional data streams*, Discrete and Computational Geometry, 36 (2006), pp. 633–655.

[129] H. Takeda, S. Farsiu, and P. Milanfar, *Kernel regression for image processing and reconstruction*, Image Processing, IEEE Transactions on, 16 (2007), pp. 349–366.

[130] G. R. Terrell, *The maximal smoothing principle in density estimation*, Journal of the American Statistical Association, 85 (1990), pp. 470–477.

[131] F. Tom, *Mining the quantified self: Personal knowledge discovery as a challenge for data science*, Big Data, 3 (2016), pp. 249–266.

[132] B. A. Turlach, *Bandwidth selection in kernel density estimation: A review*. Discussion paper 9317, Istitut de Statistique, UCL, Louvain-la-Neuve, Belgium.

[133] E. Ullman, *A theory of location for cities*, American Journal of Sociology, (1941), pp. 853–864.

[134] V. Vapnik, *Inductive principles of the search for empirical dependencies*, in Proceedings 2nd Annual Workshop on a Computational Learning Theory, 1989.

[135] V. Vapnik and A. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability and its Applications, 16 (1971), pp. 264–280.

[136] K. R. Varadarajan, *A divide-and-conquer algorithm for min-cost perfect matching in the plane*, in Proceedings of the 39th IEEE Symposium on Foundations of Computer Science, 1998.

[137] C. Villani, *Topics in Optimal Transportation*, American Mathematical Society, 2003.

[138] J. S. Vitter, *Random sampling with a reservoir*, ACM Transactions on Mathematical Software (TOMS), 11 (1985), pp. 37–57.

[139] J. S. Vitter, M. Wang, and B. Iyer, *Data cube approximation and histograms via wavelets*, in CIKM, 1998.

[140] G. Wahba, *Support vector machines, reproducing kernel Hilbert spaces, and randomization GACV*, in Advances in Kernel Methods – Support Vector Learning, Bernhard Schölkopf and Alezander J. Smola and Christopher J. C. Burges and Rosanna Soentpiet, 1999, pp. 69–88.

[141] M. P. Wand and M. C. Jones, *Multivariate plug-in bandwidth selection*, Computational Statistics, 9 (1994), pp. 97–116.

[142] G. S. Watson, *Smooth regression analysis*, Indian Journal of Statistics, Series A, 26 (1964), pp. 359–372.

[143] X. Wei and Y. Li, *Theoretical analysis of a rigid coreset minimum enclosing ball algorithm for kernel regression estimation*, in International Symposium on Neural Networks, 2008, pp. 741–752.

[144] J. R. Wolberg, *Expert Trading Systems: Modeling Financial Markets with Kernel Regression*, Wiley, 2000.

[145] C. Yang, R. Duraiswami, and L. S. Davis, *Efficient kernel machines using the improved fast gauss transform*, in NIPS, 2004.

[146] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, *Improved fast Gauss transform and efficient kernel density estimation*, in Proceedings of the 9th International Conference on Computer Vision, 2003, pp. 664–671.

[147] T. Zhang, R. Ramakrishnan, and M. Livny, *Fast density estimation using cf-kernel for very large databases*, in KDD, 1999, pp. 312–316.

[148] X. Zhang, M. L. King, and R. J. Hyndman, *A bayesian approach to bandwidth selection for multivariate kernel density estimation*, CS&DA, 50 (2006), pp. 3009–3031.

[149] Y. Zheng, J. Jestes, J. M. Phillips, and F. Li, *Quality and efficiency in kernel density estimates for large data*, in Proceedings of the ACM Conference on the Management of Data (SIGMOD), 2012.

[150] Y. Zheng and J. M. Phillips, *L_infty error and bandwith selection for kernel density estimates of large data*, in Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015.

[151] Y. Zheng and J. M. Phillips, *Coresets for kernel regression*, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, New York, NY, USA, 2017, ACM, pp. 645–654.