Coresets for Kernel Regression

Yan Zheng
University of Utah
Salt Lake City, UT 84108
yanzheng@cs.utah.edu

Jeff M. Phillips
University of Utah
Salt Lake City, UT 84108
jeffp@cs.utah.edu

ABSTRACT

Kernel regression is an essential and ubiquitous tool for non-parametric data analysis, particularly popular among time series and spatial data. However, the central operation which is performed many times, evaluating a kernel on the data set, takes linear time. This is impractical for modern large data sets.

In this paper we describe coresets for kernel regression: compressed data sets which can be used as proxy for the original data and have provably bounded worst case error. The size of the coresets are 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.

ACM Reference format:
Yan Zheng and Jeff M. Phillips. 2017. Coresets for Kernel Regression. In *Proceedings of KDD '17, Halifax, NS, Canada, August 13-17, 2017,* 10 pages.
https://doi.org/10.1145/3097983.3098000

1 INTRODUCTION

Kernel regression [14, 25] 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 [27] to weather monitoring [21] to quantified self [23].

However, as these data sets have grown to enormous scale, these kernel regression techniques have hit computational bottlenecks. Just to evaluate the model at a single query point takes $O(|P|)$ time. This runtime is completely infeasible in comparison to parametric models where it takes $O(1)$ time, especially in a common situation where many (perhaps $O(|P|)$ queries) are made. This paper explores near-linear and sub-linear techniques to compress data for kernel regression, to bring these powerful approaches to large data.

Thanks to NSF CCF-1350888, IIS-1251019, ACI-1443046, CNS-1514520, CNS-1564287.

ACM ISBN 978-1-4503-4887-4/17/08…$15.00
https://doi.org/10.1145/3097983.3098000

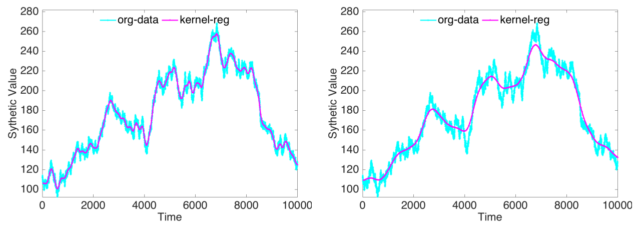

Figure 1: Kernel regression of synthetic data with bandwidth 50 (left) and 200 (right).

1.1 Basic Definitions

Kernel regression is based on a *kernel*, which is a bivariate function $K : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ which in this setting we can restrict to map to a range of $[0, 1]$ without loss of generality. As input points get closer, the kernel value should get larger. In this paper we will focus on Gaussian kernels so $K(p, q) = \exp(-\|p - q\|^2/\sigma^2)$ for a smoothing *bandwidth* parameter σ. These kernels have nice properties since they are smooth, but other kernels such as Laplace, Epanechnikov, or Triangle which have Lipschitz bounds should be perfectly suited for any of our results.

We consider an input data sets $P \subset \mathbb{R}^{d+1}$. 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 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).

A *kernel density estimate* (KDE) is a smooth function defined by convolving the data set with the kernel, for a query point $q \in \mathbb{R}^d$ as

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

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

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

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

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

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 visualizes, for $d = 1$, the kernel regression and its original data of a synthetic time series dataset (see Section 4) with bandwidth 50 and 200. The query q can be any time point between 0 and 10,000.

There are various other forms of kernel regression [19]. But in this paper we focus on the Nadaraya-Watson variety [14, 25] as it has an important long history and has been widely used in areas such as image processing [22] and economics [3]. Moreover, 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

The brute force solution of kernel regression is time consuming as each computation calculates KDE and WKDE, which takes $O(|P|)$ time. In this paper, 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 can for instance be a subset of P_x, 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 is 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 is 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 this data is a challenging problem, let alone analyzing it. 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 is 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 [20] central power database, power data serves as a way to monitor the cloud performance. It has scalar values and changes 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.
- The data is 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 [11], 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.

1.3 Our Approach

To formalize the meaning of $\textsc{kr}_S(q)$ is "close" to $\textsc{kr}_P(q)$, 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 non-existent trend or a non-existent 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

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

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 $\textsc{kr}_P(q)$ has in its denominator $\textsc{kde}_P(q)$, so when $\textsc{kde}_P(q)$ is very close to 0, then $\textsc{kr}_P(q)$ is very unstable. So we consider a domain $\mathcal{U}$ which is defined by a mild condition on $\textsc{kde}_P(q)$; in particular that $\textsc{kde}_P(q)$ is above some very small value ρ.

To further put this error bound in perspective, consider instead the relative error $\max_{q \in \mathcal{U}} \frac{\textsc{kr}_S(q)}{\textsc{kr}_P(q)}$. This is unstable whenever $\textsc{kr}_P(q)$ is close to 0. And, furthermore, the q where $\textsc{kr}_P(q)$ 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.

1.4 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 is indeed sampled, our bounds measure the error present before any analysis is applied. However, random sampling performs poor 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 [5]. We also propose some modifications which demonstrate sizable empirically improvements. Interestingly, effective coresets for KDEs [29], do not perform the best for kernel regression.

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 $S$
```

In particular, our recommended method **G-Aggregate** for $P_x \subset \mathbb{R}^1$, carefully aggregates data over a fixed size non-empty grid cells; it takes $O(|P|)$ after sorting the data. For $P_x \subset \mathbb{R}^2$, 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.

1.5 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 [17], including many for parametric regression variants like least-square regression [4] and l_p regression [7].

The only non-parametric regression coreset we are aware of is a form of kernel regression [26] 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 $O(1/\varepsilon)$ non-zero 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 [2, 12, 16, 29] 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.

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 [5]), 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" dataset. This is known to approximate kernel density estimates [2, 9, 12], 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 $(c, \text{KR}_P(c))$ in $S$
6: return $S$
```

k-Center (kCen): This method creates a k-center clustering of P_x using the greedy Gonzalez algorithms [8]; 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) [1] and also the initial step in (improved) fast Gauss transforms [28]. 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 a coreset for KDEs, 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 non-empty 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 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 non-empty 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 [6] and the illustrative toy example in Figure 2. We will see the improvement is especially significant for $P_x \subset \mathbb{R}^2$.

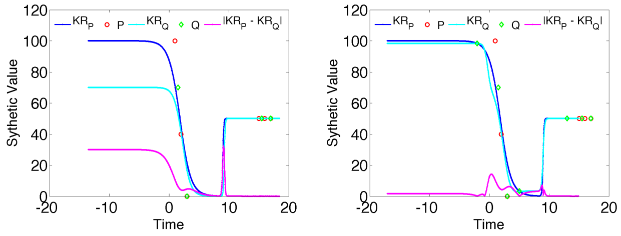

Figure 2: Example improvement of Aggregate-Neighbor (right) over G-Aggregate (left). Input $P = \{(1\ \ 100), (2\ \ 40), (3\ \ 0), (15\ \ 50), (16\ \ 50), (17\ \ 50)\}$. With G-Aggregate with $\gamma = 2$, the coreset $Q = \{(1.5\ 70), (3\ 0), (15.5\ 50), (17\ 50)\}$. The largest errors occur at $x < 0$ and around $x = 9$. If we add the extra points at the empty grid cells with non-empty neighbor grids, i.e., Aggregate-Neighbor, then L_∞ is significantly reduced. We add three points $(-2.06\ 98.3124)$, $(5\ 3.2559)$ and $(13\ 50)$.

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 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 $\text{width}(R_i) = a^{i-1}\text{width}(R_1)$ 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.

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 $\text{KDE}_P(q) > \rho$, then
$$|\text{KR}_P(q) - \text{KR}_S(q)| \le \varepsilon M,$$
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 2 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. A data set X and a family of subsets $\mathcal{R}$ define a *range space* $(X, \mathcal{R})$, and the range space's VC-dimension ν [24] (informally) describes the combinatorial complexity of the ranges; typically $\nu = \Theta(d)$. A kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ is *linked* to a range space if for every threshold τ and subset $Y_x \subset X$ defined $Y_x = \{y \in X \mid K(x, y) \ge \tau,\ x \in \mathbb{R}^d\}$ there exists a range $R \in \mathcal{R}$ such that $R \cap X = Y_x$. Importantly, all centrally symmetric kernels (including Gaussians) are linked to a range space where $\mathcal{R} = \mathcal{B}$, meaning all subsets are defined by inclusion in balls.

A *relative (ρ, ε)-approximation* of $(X, \mathcal{R})$ is a set Y
$$\max_{R \in \mathcal{R}} \left| \frac{|R \cap X|}{|X|} - \frac{|R \cap Y|}{|Y|} \right| \le \varepsilon \max\left\{ \frac{|R \cap X|}{|X|}, \rho \right\}.$$

Similarly, define a *relative (ρ, ε)-approximation* of (P, K) for kernel K as a set S such that
$$\max_{x \in \mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_S(x)| \le \varepsilon \max\{\text{KDE}_P(x), \rho\}.$$

Define a *(ρ, ε)-approximation for kernel regression of P* as a set S such that $\text{KDE}_P(q) \ge \rho$, then
$$|\text{KR}_S(q) - \text{KR}_P(q)| \le \varepsilon M,$$
where $M = \max_{p,p' \in P} |p_y - p'_y|$. Define a (non-relative) ε-approximation Y of a range space $(X, \mathcal{R})$, so
$$\max_{R \in \mathcal{R}} \left| \frac{|R \cap X|}{|X|} - \frac{|R \cap Y|}{|Y|} \right| \le \varepsilon.$$

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 $O((1/\rho\varepsilon^2)(\nu \log(1/\rho) + \log(1/\delta)))$ [10, 13]. Given an ε-approximation S of a range space linked to K, then it is known [12] that it is also a (non-relative) ε-approximation of (P, K). In this paper (in the Appendix A) we generalize this linking result (roughly following the structure of the proof in [12]) to relative (ρ, ε)-approximations.

Theorem 3.1. *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 3.2. *For error parameters $\alpha, \beta, \rho > 0$, with $\alpha \le 1/2$, 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*
$$|\mathit{KDE}_P(q) - \mathit{KDE}_S(q)| < \alpha \max\{\mathit{KDE}_P(q), \rho\}$$
$$|\mathit{WKDE}_P(q) - \mathit{WKDE}_S(q)| < \beta M.$$

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

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 $|\text{WKDE}_P(q) - \text{WKDE}_S(q)| \le \beta$, and also ensures $1 \le \text{KR}_P(q) \le 2$. 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 $0 \le \text{WKDE}_P(q) \le 2$. Thus we can consider relative error bounds, using $\textsc{kde}_P(q) > \rho$, and hence also $\textsc{wkde}_P(q) > \rho$.

$$\begin{aligned}\frac{\textsc{kr}_S(q)}{\textsc{kr}_P(q)} &= \frac{\textsc{wkde}_S(q)}{\textsc{wkde}_P(q)}\frac{\textsc{kde}_P(q)}{\textsc{kde}_S(q)}\\ &\geq \left(1-\frac{\beta}{\textsc{wkde}_P(q)}\right)\left(1-\frac{\alpha}{1+\alpha}\right)\\ &= 1-\frac{\beta}{\textsc{wkde}_P(q)}-\frac{\alpha}{1+\alpha}+\frac{\alpha\beta}{(1+\alpha)\textsc{wkde}_P(q)}\\ &\geq 1-\frac{\beta}{\rho}-\alpha\end{aligned}$$

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

$$\begin{aligned}\frac{\textsc{kr}_S(q)}{\textsc{kr}_P(q)} &= \frac{\textsc{wkde}_S(q)}{\textsc{wkde}_P(q)}\frac{\textsc{kde}_P(q)}{\textsc{kde}_S(q)}\\ &\leq \left(1+\frac{\beta}{\textsc{wkde}_P(q)}\right)\left(1+\frac{\alpha}{1-\alpha}\right)\\ &= 1+\frac{\beta}{\textsc{wkde}_P(q)}+\frac{\alpha}{1-\alpha}+\frac{\alpha\beta}{(1-\alpha)\textsc{wkde}_P(q)}\\ &\leq 1+\frac{\beta}{\rho}+\frac{\alpha}{1-\alpha}+\frac{\alpha\beta}{(1-\alpha)\rho}\\ &= 1+\frac{\beta}{(1-\alpha)\rho}+\frac{\alpha}{1-\alpha}\\ &\leq 1+\frac{2\beta}{\rho}+2\alpha\end{aligned}$$

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

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

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 3.3. *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. $\square$

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 3.4. *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 $(P, \mathcal{R})$ where $\mathcal{R}$ 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 $R \in \mathcal{R}$ 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 $(P, \mathcal{R})$, 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 $B \in \mathcal{B}$

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

This results generalizes to weighted kernel density estimates, for centrally-symmetric and non-increasing (as function of distance from center) kernels, following [12]. The only change is using the weighted bound in Lemma 3.4, 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 3.5. *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*

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

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

Now we are ready to show the main result.

Theorem 3.6. *Consider a point set $P \subset \mathbb{R}^{d+1}$ of arbitrary size, and parameters $\rho, \varepsilon \in (0, 1)$. Let S be a uniform sample from P of size $O(\frac{1}{\varepsilon^2 \rho^2}(d \log(1/\rho) + \log(2/\delta)))$, 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 [24] 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 $1 - \delta/2$ [10, 13]. Theorem 3.1 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 3.2 with $\alpha = 2\varepsilon'$.

Second we invoke Theorem 3.5 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 3.2 with $\beta = \varepsilon' \rho$.

Setting $\varepsilon' = \varepsilon/16$ invoking Lemma 3.2, then with probability at least $1 - (\delta/2 + \delta/2) = 1 - \delta$, for any $q \in \mathbb{R}^d$ that $|\text{KR}_S(q) - \text{KR}_P(q)| \leq 4(\varepsilon/8 + (\varepsilon\rho/16)/\rho)M \leq \varepsilon M$. $\square$

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 have the same asymptotic error bounds for d constant.

Theorem 3.7. ***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 3.2. 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 $\text{KDE}_P(q) \geq \rho$, then $|\text{KDE}_P(q) - \text{KDE}_S(q)| \leq \frac{\gamma\sqrt{d}}{\rho\sigma} \max\{\rho, \text{KDE}_P(q)\}$.

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 $|p_y - s_y| \leq M = \max_{p,p' \in P} |p_y - p'_y|$. Hence for all $q \in \mathbb{R}^d$, then $|\text{WKDE}_P(q) - \text{WKDE}_S(q)| \leq \gamma\sqrt{d}M/\sigma$.

Combining these two bounds together with Lemma 3.2 we obtain (for q with $\text{KDE}_P(q) \geq \rho$) that $|\text{KR}_P(q) - \text{KR}_S(q)| \leq 4(\frac{\gamma\sqrt{d}}{\rho\sigma} + \frac{\gamma\sqrt{d}}{\sigma}/\rho)M = 4(\frac{\varepsilon}{8} + \frac{\varepsilon\rho}{8}/\rho)M = \varepsilon M$. $\square$

By aggregating on each relevant grid cell, we bound coreset size with $\Delta = \max_{p,p' \in P} \|p_x - p'_x\|/\sigma$.

Corollary 3.8. *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 $\text{width}(R_1)$ 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 3.9. *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. For example, there exists simple examples with wildly varying density where the Z-order techniques approximates the density well, but not the function values. Hence, we either cannot bound the accuracy or the size for this method.

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.

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" dataset. CloudLab [20] 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 $i = 0, 1, \cdots$ and y_i is the corresponding y coordinates. It mimics a stock price so the next data depends on the previous one plus some

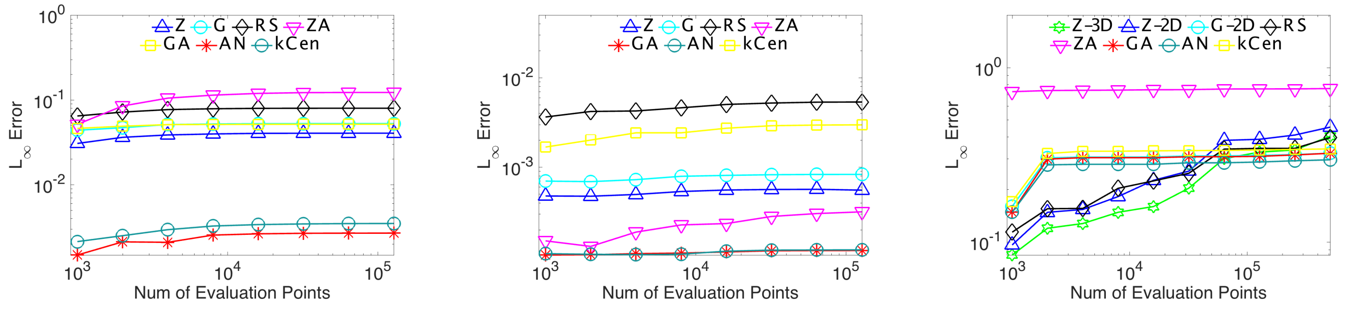

Figure 3: The maximum L_∞ error found based on the number of evaluation points on real (left) and synthetic data(middle) when $P_x \subset \mathbb{R}^1$, and real data(right) when $P_x \subset \mathbb{R}^2$.

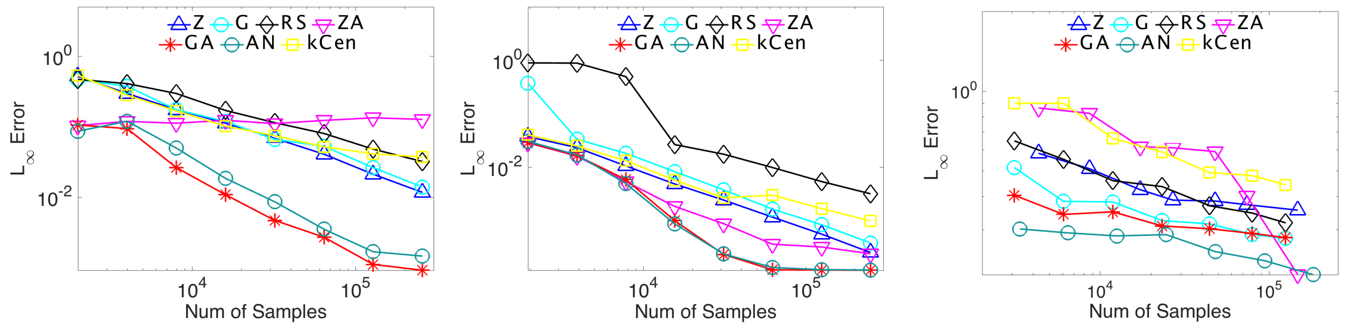

Figure 4: L_∞ error for coresets when also testing sparse regions on real data(left) and synthetic data(middle) when $P_x \subset \mathbb{R}$, and real data(right) when $P_x \subset \mathbb{R}^2$.

random noise. In the experiment, we set $c = 0, \phi = 1, y_0 = 10, \sigma = 1$ and generate 1 million points. The first 10,000 points and kernel regression with bandwidth 50 and 200 are shown in Figure 1.

For $P_x \subset \mathbb{R}^2$ real dataset, 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 dataset can give a good approximation of when the highway data point is added.

For high dimension experiment, we consider two datasets. One is house price dataset (CAD) in StatLib [15] and the other is Physicochemical Properties of Protein Tertiary Structure Dataset (CASP) from UCI machine learning repository. CAD dataset contains 20,640 observations on housing prices with 9 economic covariates and CASP dataset has 45,730 data points for 10 random variables. For both datasets, we use first 6 features to do the kernel regression.

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 is 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 3. 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 3), 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 dataset in $\mathbb{R}^1$, 50 for the synthetic dataset in $\mathbb{R}^1$, and 50 for real data in $\mathbb{R}^2$; other bandwidths have similar performance.

Figure 4 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 2. **G-Aggregate** and **Aggregate-Neighbor** (and sometimes **Z-Aggregate**) work significantly better compared to all the other methods in all datasets 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

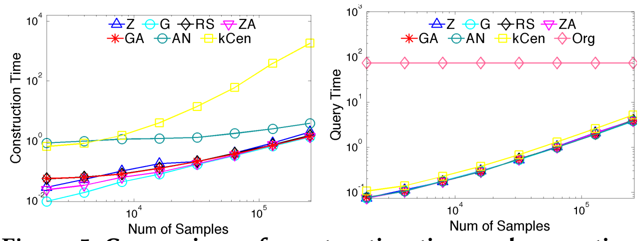

Figure 5: Comparison of construction time and query time for real dataset with $P_x \subset \mathbb{R}$.

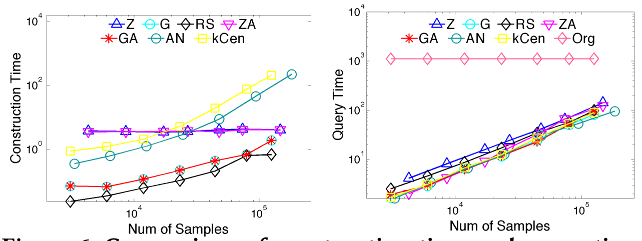

Figure 6: Comparison of construction time and query time for real dataset with $P_x \subset \mathbb{R}^2$.

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 4, 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.

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 dataset (denoted Org) for the dataset $P_x \subset \mathbb{R}$. For both comparisons, inspired by the Improved Fast Gauss Transform [28] 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 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 6), 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 $P_x \subset \mathbb{R}^2$ – this it is probably only worth it if pre-processing 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) 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 achieves 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 seen in Section 3 we prove very strong error guarantees for these methods.

4.4 Consistency with Bandwidth

In Figure 7, we test the consistency of the algorithms by varying the bandwidth. We fix the number of evaluation point as 128,000 and the median coreset size as 62,500. 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 3 and aligns with the notion that the more we smooth the data, the more stable it is, and the fewer data points we 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, the data is so smoothed these methods, which in this setting exactly or roughly amount to a random sample, work better than trying to fit gridded data to circularly smoothed estimates. 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 [18].

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 8. 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.

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 doesn't 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 non-empty grids for CAD dataset are $\{902, 1367, 1863, 2538, 3150, 3791\}$ and for CASP dataset are $\{1034, 1554, 2122, 2742, 3543, 4342\}$. The relationship of L_∞ error and coreset size is shown in the left Figure 9, using bandwidths 3 and 3.8 respectively. The error decreases when

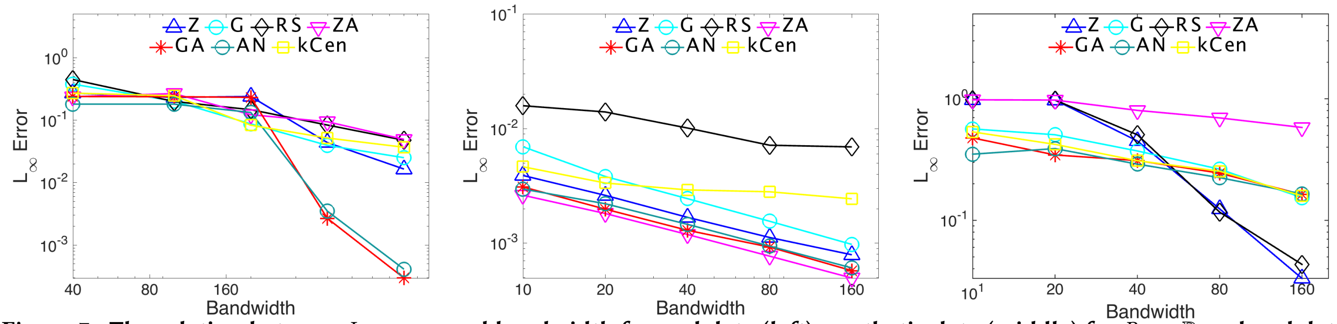

Figure 7: The relation between L_∞ error and bandwidth for real data (left), synthetic data (middle) for $P_x \subset \mathbb{R}$ and real data (right) for $P_x \subset \mathbb{R}^2$.

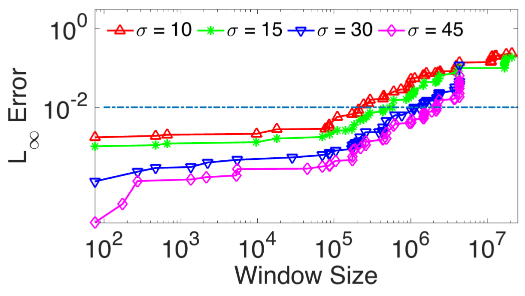

Figure 8: The relation between window size and L_∞ error for progressive G-Aggregate method.

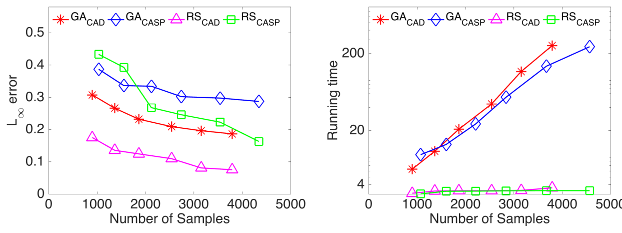

Figure 9: Left: L_∞ error for coresets of high dimensional datasets. Right: Running time to generate the coresets.

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 9) 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 datasets.

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/~jeffp/students/kernel-reg/.

REFERENCES

[1] T. Arnold. Sparse density representations for simultaneous inference on large spatial datasets. *arXiv preprint arXiv:1510.00755*, 2015.
[2] S. Balakrishnan, B. T. Fasy, F. Lecci, A. Rinaldo, A. Singh, and L. Wasserman. Statistical inference for persistent homology. *Annals of Statistics*, 2014.
[3] R. Blundell and A. Duncan. Kernel regression in empirical microeconomics. *Journal of Human Resources*, pages 62–87, 1998.
[4] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal coresets for least-squares regression. *IEEE Trans. Information Theory*, 59(10), 2013.
[5] J. D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *System Administration Conference (LISA)*. USENIX, 2000.
[6] S. Chan, I. Diakonikolas, R. A. Servedio, and X. Sun. Efficient density estimation via piecewise polynomial approximation. In *STOC*, 2014.
[7] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SICOMP*, 38:2060–2078, 2009.
[8] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
[9] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *JMLR*, 13:723–773, 2012.
[10] S. Har-Peled. *Geometric approximation algorithms*. AMS, 2011.
[11] 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(2):211–225, 2002.
[12] S. Joshi, R. V. Kommaraju, J. M. Phillips, and S. Venkatasubramanian. Comparing distributions and shapes using the kernel distance. *SoCG*, 2011.
[13] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the samples complexity of learning. *J. Comp. and Sys. Sci.*, 62:516–527, 2001.
[14] E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–142, 1964.
[15] R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.
[16] J. M. Phillips. eps-samples for kernels. *SODA*, 2013.
[17] J. M. Phillips. Coresets and sketches. In *Handbook on Discrete and Computational Geometry*, chapter 47. CRC Press, 3rd edition, 2016.
[18] J. M. Phillips and Y. Zheng. L_infty error and bandwith selection for kernel density estimates of large data. In *KDD*, 2015.
[19] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[20] R. Ricci, E. Eide, and The CloudLab Team. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *USENIX ;login:*, 39(6), Dec. 2014.
[21] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy. Predicting solar generation from weather forecasts using machine learning. In *SmartGridComm*, 2011.
[22] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Trans Image Processing*, 16:349–366, 2007.
[23] F. Tom. Mining the quantified self: Personal knowledge discovery as a challenge for data science. *Big Data*, 3:249–266, 2016.
[24] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Th. of Prob. Applic.*, 16:264–280, 1971.
[25] G. S. Watson. Smooth regression analysis. *Indian Journal of Statistics, Series A*, 26:359–372, 1964.
[26] 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*, pages 741–752, 2008.
[27] J. R. Wolberg. *Expert Trading Systems: Modeling Financial Markets with Kernel Regression*. Wiley, 2000.
[28] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *ICCV*, 2003.
[29] Y. Zheng, J. Jestes, J. M. Phillips, and F. Li. Quality and efficiency in kernel density estimates for large data. In *SIGMOD*, 2012.

A LINKING AND (ρ, ε)-APPROXIMATIONS FOR KERNEL REGRESSION

Theorem A.1. *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 point set 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 [12], 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 $t \le |S|$ be the index of the last set P_j such that $|P_j| = g$. Note that for all $p_i \in P_j$ (for $j \le t$) we have $K(s_j, q) \ge K(p_i, q)$, thus $\kappa_S(\{s_j\}, q) \ge \kappa_P(P_j, q)$. We can now bound the undercount as

$$D(P, S, q) = \sum_{j=1}^{|S|}\Big(\kappa_P(P_j, q)\Big) - \kappa_S(\{s_j\}, q) + \sum_{j=1}^{|S|}\kappa_P(P'_j, q) \le \sum_{j=1}^{t+1}\kappa_P(P'_j, q)$$

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 $j \le t$) 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

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

Hence

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

We can now bound

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

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

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

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 $\frac{1}{|P|}\sum_{j=1}^{t+1}|P'_j| \le \varepsilon\frac{|P \cap H|}{|P|}$.

Finally, when $\frac{|P \cap H|}{|P|} \le \rho$,

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

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 $t \le |S|$ 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 $j \ge t$) we have $K(s_j, q) \le K(p_i, q)$, thus $\kappa_S(\{s_j\}, q) \le \kappa_P(P_j, q)$. We can now bound the (negative) overcount as

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

and using a similar approach as with the undercount, we can show

$$D(P, S, q) \ge -\varepsilon\rho K^+.$$

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