Coreset for Rational Functions

David Denisov

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE MASTER'S DEGREE

University of Haifa
Faculty of Social Sciences
Department of Computer Science

November, 2022

Coreset for Rational Functions

By: David Denisov

Supervised by: Prof. Dan Feldman

University of Haifa
Faculty of Social Sciences
Department of Computer Science

November, 2022

Approved by: _____ Date: _____
(Supervisor)

Approved by: _____ Date: _____
(Chairperson of Master's studies Committee)

Contents

Coreset for Rational Functions

David Denisov

Abstract

We consider the problem of fitting a rational function $f : \mathbb{R} \to \mathbb{R}$ to a given discrete time-series $g : \{1, \cdots, n\} \to \mathbb{R}$. This is by minimizing the sum of distances (loss function) $\ell(f) := \sum_{i=1}^{n} |f(i) - g(i)|$, possibly with additional constraints and regularization terms that may depend on f. A natural application is Auto-Regression, where we wish to approximate such a time-series by a recursive sequence $F_n = \sum_{i=1}^{k} \theta_i F_{n-i}$, e.g. a Fibonacci sequence, where $\theta \in \mathbb{R}^k$ are the model parameters, and $k \geq 1$ is constant. For $\varepsilon \in (0, 1)$, an ε-coreset for this problem is a small data structure that approximates $\ell(f)$ up to $1 \pm \varepsilon$ multiplicative factor, for every rational function f of degree k or smaller. We prove that every signal has an ε-coreset of size $O(n^{o(1)}/\varepsilon^2)$, and provide a construction algorithm that computes it in $O(n^{1+o(1)})$ time. Open source code is provided, as well as extensive experimental results, on both real and synthetic datasets, including comparisons to solvers that use Numpy and Scipy.

List of Algorithms

List of Figures

Chapter 1

Introduction

The original motivation for this thesis was to suggest provable and efficient approximation algorithms for fitting input data by a stochastic model or its variants, such as Hidden-Markov Models (HMM) (see [7, 39, 41, 50, 57, 71], and reference therein), Baysian Networks (see [1, 41, 43, 56], and reference therein), auto-regression [24], and Decision Markov Process [58]. Informally, and in the context of this thesis, a model defines a *time-series* (sequence, discrete signal, time-series) $G : [n] \to \mathbb{R}$ where $[n] := \{1, \cdots, n\}$, and the value $G(t)$ at time (integer) $t \geq 1$ is a function of only the previous (constant) $k \geq 1$ past values $G(t-1), \cdots, G(t-k)$ in the sequence, and the model's parameter θ.

In Markov models, the probability of visiting the ith node in the weighted graph (V, E, w) in time t might be defined by $G_i(t)$, which depends on $G_1(t-1), \cdots G_{|V|}(t-1)$, and $G(t) = \sum_{i=1}^{|V|} \theta_i F_i(t)$ is the expected node. Here, $\theta_{i,j} := w(e_{i,j})$ is the probability of transition from V_i to V_j through the edge $e_{i,j} \in E$.

From stochastic process to machine learning. In machine learning and data science, the input is usually the data or an observed time-series $G : [n] \to \mathbb{R}$, and not the model. The assumption is that the data was generated via a (hidden, unknown) model F_θ , with additional random noise N, i.e., $G \sim F_\theta + N$. Here, $\theta \in \Theta$ is a vector of parameters from a (usually infinite) set $\Theta \subseteq \mathbb{R}^k$ of parameters. Estimating this model F_θ might help to explain the time-series at hand and predict its future values.

Frameworks such as The Risk Minimization [66] (RMS), (MAP), or Maximum Likelihood (ML), are used to define a (loss) function $\ell(F_\theta, G)$ which assigns a non-negative "fitting" value to every parameters vector $\theta \in \Theta$ and its corresponding model F_θ. The goal is then to compute the model that best fits the given observed time-series G. If the noise for each time sample is independent, we may define the loss $\ell(F(i), G(i))$ for each time stamp $i \in [n]$ independently, to obtain

$$\ell(F_\theta, G) = \sum_{i=1}^{n} \ell\big(F(i), G(i)\big).$$

Formally, the desired output is then the model's parameters vector $\theta \in \Theta$ which best fits the input time-series, i.e. minimizes $\ell(F_\theta, G)$ above.

1.1 Motivation

Unfortunately, most existing results for fitting stochastic models to data seem to be based on heuristics with little provable approximation guarantees. We thus investigate a simplified but fundamental version, called *auto-regression*, which has a provable but not so efficient solution using polynomial system solvers, after applying the technique of *generating functions*; See Lemma 28. This technique is strongly related to the Fourier, Laplace and z-Transform, as explained below. We define an auto-regression inspired by [17], [72], and [24], as follows.

Definition 1. *A time-series $F : [n] \to \mathbb{R}$ is an auto-regression (AR for short) of degree k, if there exists a vector $\theta = (\theta_1, \cdots, \theta_k) \in \mathbb{R}^k$ of coefficients, such that $F(t) = \theta_1 F(t-1) + \cdots + \theta_k F(t-k)$ for every $t \geq k+1$. The polynomial $P(x) = x^k - \theta_k x^{k-1} - \cdots - \theta_1$ is called the* characteristic polynomial *of F.*

Toy example: Fibonacci sequence. The first appearance of the Fibonacci sequence dates back to the year of 1202, in a book by Leonardo of Pisa [60], known as Fibonacci. Fibonacci considered the growth of an idealized (biologically unrealistic) rabbit population, assuming that:

(i) A single newly born pair of rabbits are put in a field.

(ii) At the end of its second month, a female can produce another pair of rabbits.

(iii) Rabbits never die and a mating pair always produces one new pair every month from the second month on.

The puzzle that Fibonacci posed was: how many pairs will there be in one year?

Substituting $k = 2$, $\theta = (1, 1)$ and $F(1) = F(2) = 1$ in Definition 1 yields the *Fibonacci sequence*, $F(t) = F(t-1) + F(t-2)$, where $F(1) = F(2) = 1$.

From Auto-regression to Rational functions. In the corresponding "data science version" of Fibonacci's sequence, the input is the time-series $G(1), G(2), \cdots$, where $G(i) = F(i) + N(i)$ for every $i \geq 1$, where $N(i)$ denotes noise from some distribution. A straight forward method to recover the original model is by directly minimizing the squared error between the given noisy time-series and the fitted values, as done e.g. in [17], using simple linear regression. However, this has a major drawback: AR time-series usually grows exponentially, like geometric sequences, and thus the loss will be dominated by the last few terms in the time-series. Moreover, small changes in the time domain have exponential affect over time, so it makes more sense to assume added noise in the frequency or generative function domain. Intuitively, noise in analog signals such as audio/video signals from an analog radio/tv is added in the frequency domain, such as aliasing of channels [33, 49, 59], and not in the time domain, such as the volume.

To solve these issues, the fitting is done for the corresponding generation functions as follows.

Proposition 2 (generative function [72])**.** *Consider an AR time-series $F : \mathbb{N} \to \mathbb{R}$ and its characteristic polynomial $P : \mathbb{R} \to \mathbb{R}$ of degree k; see Definition 1. Let $Q(x) = x^k P(\frac{1}{x})$ be the polynomial whose coefficients are the coefficients of P in reverse order. Then there is a polynomial $R(x)$ of degree at most $k-1$ such that $f(x) := \sum_{i=1}^{\infty} F(i)x^{i-1} = \frac{R(x)}{Q(x)}$, for every $x \in \mathbb{R}$ where $f(x)$ is finite. The function f is called the* generative function *of F.*

Inspired by the above motivation, we define the following loss function for the AR recovery problem, where the time-series F and the AR time-series G are replaced by their generative functions f and g, respectively; a formal definition for our loss function is given in Section 3.1.

Definition 3 (Rational Function Fitting (RFF))**.** *Given a time-series $g : [n] \to \mathbb{R}$ and an integer $k \geq 1$, the rational-function fitting problem is to minimize $\sum_{x=1}^{n} |f(x) - g(x)|$ over every rational function $f : \mathbb{R} \to \mathbb{R}$ whose enumerator and denominator are polynomials of degree at most k.*

While this thesis focuses on sum of errors (distances), we expect easy generalization to squared-distances, robust M-estimators and any other loss function that satisfies the triangle inequality, up to a constant factor, as in other coreset constructions [20, 32, 38].

1.2 Coresets

Informally, for an input set P that consists of d-dimensional points, a set Q of models, an approximation error $\varepsilon \in (0, 1)$, and a loss function ℓ, a *coreset* C is a data structure that approximates the loss $\ell(P, q)$ for every model $q \in Q$, up to a multiplicative factor of $1 \pm \varepsilon$, in time that depends only on the size of C. Such a coreset C is efficient if it is much smaller than the original input P.

Why coresets? A natural motivation for constructing a coreset is to be able to compute an approximated model much faster, while losing little accuracy, by computing it on the small coreset instead on the original large data. Moreover, in many cases, a coreset for a given problem implies a solution to streaming and distributed data [9, 35], parallel computation [18], handle constrained variants of the problem [23], parameter tuning [37] and more. A coreset also enables efficient evaluation of many different heuristics on the data, or a single heuristics with multiple different hyperparameters; see e.g. [37]. This is due to the three main coreset properties: merge and reduce [2, 8, 25, 30], which are satisfied in the majority of the coresets, and the fact that a coreset approximates every model, in the given family of models, and not just the optimal model.

Coreset for rational functions. Unfortunately, similarly to [55], we show that the RFF problem with general input has no coreset which is a weighted subset of the input; see Claim 22 in Section 4.1.4. This was also the case e.g. in [32]. Some assumption on the input is thus inevitable. Similarly to [32] and [55], we first assume that out input is not just an arbitrary set of points, but a discrete signal, i.e., a function from $[n]$ to $\mathbb{R}$. This was also done, e.g., in [32] and [55].

Even under this assumption there is no coreset which is a weighed subset of the input; see Claim 22. Instead, we suggest a coreset construction that is small and approximates every rational function, but is not a weighted subset of the input signal.

As common in the literature, our coreset construction heavily depends on the optimal solution for the problem at hand. Fortunately, in Section 3.3 we show that a rough approximation, called

(α, β)-approximation, suffices. However, unlike other coreset papers, computing an approximated solution to the rational function fitting problem turns out to be a very challenging task. Hence, most of this thesis is dedicated to tackling this problem; see Section 2.1.

1.3 Related work

Polynomial approximations. While polynomials are usually simple and easy to handle, they do not suffice to accurately approximate non-smooth or non-Lipschitz functions; in such cases, high order polynomials are required, which leads to severe oscillations and numerical instabilities [51]. To overcome this problem, one might try to utilize piecewise polynomials or polynomial splines [47, 48, 61]. However, this results in a hard optimization problem [40, 62].

Rational function approximation. A natural alternative is to utilize rational functions for approximating the input signals; see comparison in Fig. 1-1. Given an input signal consisting of n pairs of points in $\mathbb{R}^2$, the rational function fitting (or RFF) problem aims to recover a rational function that best fits this input, as to minimize some given loss function; see Definition 4. Rational function approximation is a straight forward extension for polynomial approximations [64], yet are much more expressive due to polynomial in the denominator. A close relation between such functions and spline approximations has been demonstrated e.g. in [53]. This is also a variant upon Padé approximant method [6], where we consider loss minimization instead of derivatives. More applications can be found in Section 4.4 of [28] and in the popular book [10].

Fig. 1-1 demonstrates example cases where rational functions yield better approximations than polynomials. This is essentially a variation of the known Runge's phenomenon [16]. We do not use a rational function, that is commonly used to demonstrate Runge's phenomenon [16], since we also want to demonstrate the advantage of our methods upon existing solvers, and if all the points are exactly on a rational function existing rational interpolation methods can solve this case as well; for example Padé approximant as mentioned in [6], or rational function fitting for max deviation as mentioned in [51].

In the following figure we compare (black) a rational function computed using our algorithm `FRFF-coreset` from the Experimental Results (see Section 5), (blue) a rational function com-

puted using the `Scipy.optimize.minimize` which aims to minimize the same RFF loss as in (3.1), and (green) a polynomial of degree 3 computed using the `numpy.polyfit` function, which minimizes the sum of squared distances between the polynomial and the input. For fair comparison, all three methods have been allowed 4 free parameters. In particular, our RFF has degree 1 in the enumerator and 2 in the denominator (there are 4 free parameters, since the free variable in the denominator is set to 1), while the polynomial is of degree 3.

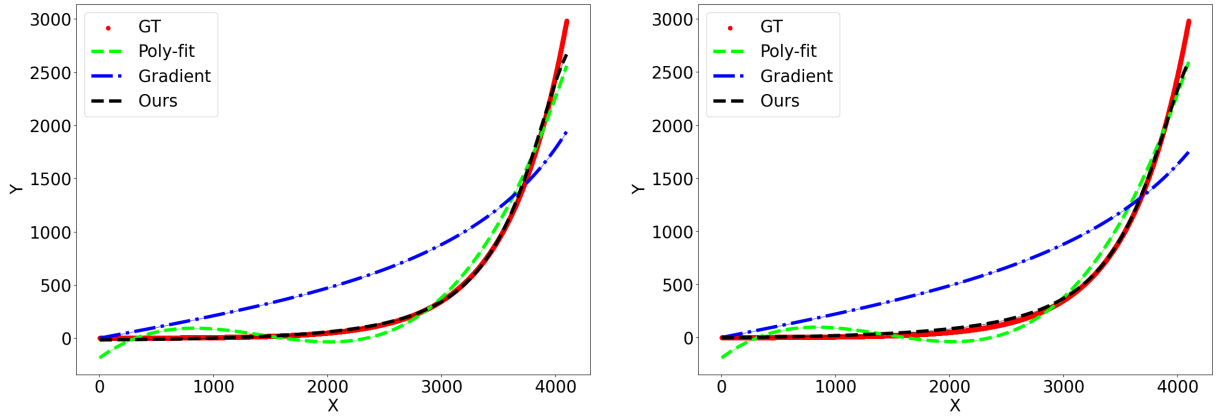

Figure 1-1: ***Rational Function Fitting (RFF).*** *A time-series $f(x) = e^{x/512}$ over $x \in \{1, \cdots, 2^{12}\}$ in red, which is denoted by GT, is the ground truth (GT). Approximation via our `FRFF-coreset` (black), `Scipy.optimize.minimize` (blue), and `numpy.polyfit` (green). (Left): All 3 methods applied on the original signal. (Right): All 3 methods were applied on our coreset whose size is 10 times smaller (than the original signal).*

Hardness of rational function approximation. To the best of our knowledge, rational function approximation has only been solved for the max deviation case (where the loss is the maximum over all the pairwise distances between the input signal and the approximation function) in [51]. Various heuristics have been suggested for other instances of the problem, see [51] with references therein.

Chapter 2

Results

This thesis suggests the first coreset for rational functions, as in Definition 6. Given an integer $k \geq 1$, approximation error $\varepsilon \in (0,1)$ and a time-series g, whose length is $|g| = n$, this coreset can be computed in $O\left(n^{\psi}\right)$ time and stored in $O\left(n^{\psi}/\varepsilon^2\right)$ memory words for arbitrarily small constant $\psi > 0$. The sum of absolute distances $\ell(P,q) = \sum_{i=1}^{n} \left|P(i) - q(i)\right|$ to a given query (rational function) q can then be computed in $O\left(n^{\psi}\right)$ time (utilizing integrals for efficient computation, see [46]), up to a factor of $1 \pm \varepsilon$.

In addition, we suggest experimental results on both synthetic and public real-world data, as well as open source for reproducing and extending our results [12].

It should be emphasized that, while the loss could be computed efficiently, due to the use of integrals the loss is non-linear and is non-trivial to minimize efficiently. Hence, it is non-trivial to use the coreset in a classic merge-reduce tree. Which was a significant source of challenge in this work.

2.1 Novelty

The suggested coreset in this thesis is very different from previous coreset papers. To our knowledge, this is the first coreset for stochastic models, such as auto-regression. Most existing coresets are motivated by problems in computational geometry or linear algebra, especially clustering, and

subspace approximation. Their input is usually a set of points (and not a time-series), with few exceptions (e.g. [15, 31]) coresets for linear regression that can be considered as a type of hyperplane approximation.

Our main challenges were also very different from existing coreset constructions. A typical coreset construction begins with what is known as an α-approximation or (α, β)-approximation for the optimal solution that can be easily constructed, e.g. using the generic algorithms in [4, 20, 13]. From this point, the main change in these papers is to compute the sensitivity or importance of each point using, e.g., the Feldman-Langberg framework [20]. The coreset is then a non-uniform random sample from the input set, based on the distribution of these sensitivities.

However, in this thesis, the sensitivity of a point is simply proportional to its distance from our (α, β)-approximation. Instead, the main challenge in this thesis is to compute an efficient (α, β)-approximation. We cannot use the existing sample techniques as in [20] since it might create too many "holes" (non-consecutive sub-signals) in the input signal. A solution for computing bicriteria for k-linear segments was suggested in [22, 32] by partitioning the input signal into consecutive equal $2k$ sub-signals, so that at least half of them would contain only a single segment, for every k-segmentation. In our case, a k rational function (or even a quadratic function) cannot be partitioned into, say, $O(k)$ linear or even polynomial functions.

Instead, we computed a "weak coreset", which guarantees a desired approximation for a constrained version of the problem; in terms of constraints on both its input signal and feasible rational functions. We then combine this weaker coreset with a merge-reduce tree for maintaining an (α, β)-approximation, which is very different from the classic merge-reduce tree that is used to construct coresets for streaming data. This tree of height $\sim \log \log n$ is also related to the not-so-common running time and space complexity of our suggested coreset.

We expect that this coreset technique would be generalized in the future for more involved stochastic models, such as graphical models, Baysian networks, and HMMs, that correspond to few dependent recursive functions, which may be formulated via signals in higher dimensional spaces.

Chapter 3

Algorithms

In this chapter we describe the main algorithm of this thesis, and its sub-routines. We begin with the required definitions and notation in Section 3.1, followed by an overview of all the algorithms in Section 3.2. Section 3.3 described the main coreset construction, based on an (α, β)-approximation, which is described in Section 3.4 and improved in Section 3.5. Section 3.6 presents the main algorithm, the compute the bicriteria and then the coreset. In Section 3.7 we suggest a quick heuristic for extracting the optimal solution from the coreset.

3.1 Notation and Definitions

Throughout this thesis, we assume that $k \geq 1$ is an integer, and denote by $\mathbb{R}^k$ the union of k-dimensional real column vectors. For every integer $n \geq 1$, we denote $[n] = \{1, \cdots, n\}$. The entries of a vector are identified by $(c_1, \cdots, c_k) \in \mathbb{R}^k$, and for every $x \in \mathbb{R}$ we denote by $\text{poly}(c, x) = \sum_{i=1}^{k} c_i x^{i-1}$ the value for $x \in \mathbb{R}$ of the polynomial of degree $k-1$ whose coefficients are the entries of c. For simplicity, we debite $\log(x) := \log_2(x)$. A *weighted set* is a pair (P, w) where $P \subseteq \mathbb{R}^2$, and $w : P \to [0, \infty)$ is called a *weights function*. A partition $\{P_1, \cdots, P_\theta\}$ of a set $P \subset \mathbb{R}^2$, where $\theta \geq 2$, is called *consecutive*, if for every $i \in [\theta - 1]$ we have $\max\{x \mid (x, y) \in P_i\} \leq \min\{x \mid (x, y) \in P_{i+1}\}$. Similarly, a partition $\{B_1, \cdots, B_\theta\}$, where $\theta \geq 2$, of an ordered set $B := (b_1, \cdots, b_{|B|})$ is called *consecutive* if for every $i \in [\theta - 1]$

we have $B_{i+1} := \{b_i \mid i \in [r_i] \setminus [r_{i+1}]\}$ for a set $r_1 \leq r_2 \leq \cdots \leq r_\theta$. A query $q \in \left(\mathbb{R}^k\right)^2$ is a pair of k-dimensional vectors. For any integer $n \geq 1$, *an n-signal* is a set P of pairs $\{(1, y_1), \cdots, (n, y_n)\}$ in $\mathbb{R}^2$. Such an n-signal corresponds to an ordered set of n reals, a discrete signal, or to the graph $(1, g(1)), \ldots, (n, g(n))$ of a function $g : [n] \to \mathbb{R}$. A set $\mathcal{P} \subset \mathbb{R}^2$ is *an interval of an n-signal* if $\mathcal{P} := \{(a, y_a), (a+1, y_{a+1}), \cdots, (b, y_b)\}$ for some $a, b \in [n]$ such that $a \leq b$. The sets $\{a, a+1, \cdots, b\}$ and $\{y_a, y_{a+1}, \cdots, y_b\}$ are the interval's *first and second coordinates*, respectively. Given a function $f : \mathbb{R} \to \mathbb{R}$, the *projection* of P onto f is defined as $\big\{\big(a, f(a)\big), \cdots, \big(b, f(b)\big)\big\}$.

We are now ready to define the Rational Function Fitting (RFF) problem, as follows.

Definition 4 (RFF)**.** *We define* $\mathrm{ratio} : \left(\mathbb{R}^k\right)^2 \times \mathbb{R} \to \mathbb{R}$ *to be the function that maps every pair* $q = (c, c') \in \left(\mathbb{R}^k\right)^2$ *and any* $x \in \mathbb{R}$ *to*

$$\mathrm{ratio}(q, x) = \mathrm{ratio}\big((c, c'), x\big) := \begin{cases} \dfrac{\mathrm{poly}(c, x)}{1 + x \cdot \mathrm{poly}(c', x)} & \text{if } 1 + x \cdot \mathrm{poly}(c', x) \neq 0 \\ \infty & \text{otherwise.} \end{cases}$$

For every pair $(x, y) \in \mathbb{R}^2$*, the* fitting loss of *approximating* (x, y) *via the rational function that corresponds to* q *is defined as*

$$D\big(q, (x, y)\big) := |y - \mathrm{ratio}(q, x)| \,.$$

For a finite set $P \subset \mathbb{R}^2$ *we define the* RFF fitting loss of q *to* P *as*

$$\ell\big(P, q\big) = \sum_{p \in P} D(q, p) = \sum_{(x,y) \in P} |y - \mathrm{ratio}(q, x)| \,.$$

More generally, for a weighted set (P, w) *we define the* RFF fitting loss of q *to* (P, w) *as*

$$\ell\big((P, w), q\big) = \sum_{p \in P} w(p) D(q, p) = \sum_{(x,y) \in P} w\big((x, y)\big) \, |y - \mathrm{ratio}(q, x)| \,. \tag{3.1}$$

A coreset construction usually requires some rough approximation to the optimal solution as its input; see Section 2.1. Unfortunately, we do not know how to compute even a constant factor approximation to the RFF problem in Definition 4, in near-linear time, but only in $(2kn)^{O(1)}$ time; see Lemma 28. Instead, our work is mostly focused on efficiently computing an (α, β) or bicriteria approximation [20], as defined bellow.

Definition 5 ((α, β)-approximation)**.** *Let $P \subseteq \mathbb{R}^2$ be an interval of an n-signal, $\beta \in [n]$, and $\alpha > 0$. Let $\{P_1, \cdots, P_\beta\}$ be a consecutive partition of P, and $q_1, \cdots, q_\beta \in \left(\mathbb{R}^k\right)^2$. The set $\{(P1, q_1) \cdots, (P_\beta, q_\beta)\}$ of β pairs is an* (α, β)-approximation *of P, if*

$$\sum_{i=1}^{\beta} \ell(P_i, q_i) \leq \alpha \cdot \min_{q \in \left(\mathbb{R}^k\right)^2} \ell(P, q).$$

If $\beta = 1$, then the $(\alpha, 1)$-approximation $B = \{(P_1, q_1)\}$ is called an α-approximation*. For every $i \in [\beta]$ we denote by P'_i the projection of P_i onto q_i. The union $\bigcup_{i=1}^{\beta} P'_i$ is called the projection of P onto B.*

A coreset for this RFF problem is defined as follows. Similarly to [55] it includes an approximation to allow a coreset construction despite the lower bound from Claim 22.

Definition 6 (ε-coreset)**.** *Let $P \subseteq \mathbb{R}^2$ be an n-signal, and $\varepsilon > 0$ be an error parameter. Let $B := \{(P_1, q_1), \cdots, (P_\beta, q_\beta)\}$ be a (α, β)-approximations approximation of P (see Definition 5), and let (C, w) be a weighted set. The tuple (B, C, w) is an* ε-coreset *for P, if for every $q \in \left(\mathbb{R}^k\right)^2$ we have*

$$\left|\ell(P, q) - \ell\big((C, w), q\big) - \ell(P', q)\right| \leq \varepsilon \cdot \ell(P, q),$$

where $P' := \left\{\big(x, \text{ratio}(q_i, x)\big) \mid i \in [\beta], (x, y) \in P_i\right\}$ is the projection of P onto B. The size for representing such coreset is defined to be $O\left(|C| + \beta k\right)$.

To efficiently compute the (α, β)-approximation in Definition 5 we will utilize restricted coresets that have similar properties but are more limited. It should be emphasized that the final coreset will not contain such a restriction. One of these requirements is the following bound on the given queries.

Definition 7 (ρ-bounded function)**.** *For every $X \subset \mathbb{R}, \rho \in [1, \infty)$, and any $(c, c') \in \left(\mathbb{R}^k\right)^2$ we say that (c, c') is ρ-bounded over X if and only if*

$$\frac{\max\limits_{x \in X} \left|1 + x \cdot \text{poly}(c', x)\right|}{\min\limits_{x \in X} \left|1 + x \cdot \text{poly}(c', x)\right|} \leq \rho,$$

and $\min\limits_{x \in X} \left|1 + x \cdot \text{poly}(c', x)\right| \neq 0$.

3.2 Overview

Algorithms 5: coreset construction. The input to Algorithm 5, which is the main algorithm, is an n-signal P, and input parameters $\varepsilon, \delta \in (0, 1/10]$. Its output, with probability at least $1 - \delta$, is an ε-coreset (B, C, w) of P whose size is in $O\left((\delta n)^{o(k)}/\varepsilon^2\right)$; see Definition 6. The main part of this algorithm aims to compute an (α, β)-approximation B, where $\alpha \in O(\log n), \beta \in \log(n)^{O(k)}$. Then this approximation is assigned in Algorithm 1 that computes a coreset from an (α, β)-approximation.

We compute B via a balanced β-tree, which is similar to the classic merge-reduce tree that is usually used to compute coresets for streaming data [9]. However, the merge and reduce steps are different, as well as the number of children in each node. Each leaf of this tree corresponds to a $(1, 1)$-approximation (i.e. optimal solution) for a consecutive partition of $\beta := \Theta\left(n^{1/\log\log n}\right)$ input points; computed via a call to Algorithm 2. Hence, there are $\Theta(n/\beta)$ leaves. An inner node in the ith level corresponds to an (α_i, β_i)-approximation in its β child nodes, $O(\beta^i)$ leaves and $O(\beta^{i+1})$ input points of its sub-tree, for every $i \in [\ell]$, where $\ell = \lceil \log\log n \rceil - 1$ is the number of levels in the tree; as $\left(n^{1/\log\log n}\right)^{\log\log n} = n$. Here, $\alpha_i = 3^i$, $\beta_i = O(1)^i$, and thus $(\alpha, \beta) = (\alpha_\ell, \beta_\ell) \in \left(O(\log n), \log(n)^{O(k)}\right)$.

Algorithm 4: the merge-reduce step. This step is computed in each inner node of the tree. For an inner node in the ith level, where $i \in \{2, \cdots, \ell\}$, the input is a set B of size β, where for every $j \in [\beta]$ each $B_j \in B$ is an $(0, r_j)$-approximation of P_i and $P_1, \cdots, P_\beta$ is an equally-sized consecutive partition of an interval of $\mathcal{P}$ an n-signal, and the output is a $\left(2, O\left(k \cdot \max\limits_{j \in [\beta]} r_j\right)\right)$

approximation for $\mathcal{P}$; see Algorithm 4 and Fig. 3-4. This (α, β)-approximation is computed by computing the follows, for every possible subset $G \subseteq [\beta]$ of size $\beta - 6k + 3$ (see Line 6):

(i) Compute an α-approximation q_G, for $\bigcup_{j \in G} P_j$ and some $\alpha > 0$. This is by first applying the weak-coreset from Algorithm 3; see Line 11.

(ii) For each $j \in G$, set $\ell_j := \ell(P_j, q_I)$ as the loss of q_I for P_j; see Line 13.

(iii) Set $G' \subset G$ to be the union of the $6k - 3$ indices $j \in C$ with the largest value ℓ_j; see Line 14.

(iv) Set $s_G := \sum_{j \in G \setminus G'} \ell_j$ to be the sum of the losses excluding the largest $|G'|$.

The final approximation for the inner node is the one that minimizes s_G, among every subset $G \subset [\beta]$ of size $\beta - 6k + 3$; see Lines [15–20]. More precisely, we take its part that approximates the union of $|B| - 2|G'|$ children whose approximation error is s_G, and take its union with the $2|G'|$ original (input) bicriterias that correspond to the child nodes $(B \setminus G) \cup G'$. The final approximation error in the inner node is thus $\min_{G \subseteq B, |G| = \beta - 6k + 3} s_G$.

Algorithm 3: restricted coresets for rational functions. Computing the α-approximation in Step (i) above can be done using Lemma 28. However, this would take $n^{O(1)}$ time and not near-linear time as desired. To this end, Algorithm 3 constructs a limited coreset for rational functions that is restricted in the following two senses:

(i) It assumes that the input signal consists of a small number of consecutive bicriteria, which is indeed the case in Step (i) of the *merge-reduce step* above.

(ii) It approximates only specific rational functions, which are 2^k-bounded over the fist coordinate of the input signal; see Definition 7.

We handle the second assumption by removing the $O(k)$ unbounded sets (via the exhaustive search described over G above, and the removal of G'), and proving that there are at most such $O(k)$ sets. It should be emphasized that the final coreset construction has no such restrictions or assumptions for either its input or queries.

This restricted coreset is computed on each child node, so that in Step (i) above we compute the α-approximation only on the union of $|G|$ coresets that corresponds to the chosen $|G|$ internal nodes. The size of the coreset, whose construction fails with probability at most $\delta \in (0, 1/10]$, is $m \in O\left(4^k \left(\log\left(\beta n/\delta\right)\right)^2\right)$ and thus the running time of our approximation algorithm (Algorithm 4) is $m^{O(k)}$, which is in $n^{o(k)} \cdot \left(2^k \log(1/\delta)\right)^{O(k)}$. Algorithm 3 computes this restricted coreset by:

(i) Partitioning the input into subsets of exponentially increasing sizes; see Fig 3-3 and Line 6.

(ii) Computing a sensitivity based sub-sample for each subsets; see Lines [8–11].

(iii) Returning the union of those coresets after appropriate re-weighting; see Lines [12–14].

As we prove, the RFF sensitivity for each point inside each sub-signal is the sensitivity of the polynomial (instead of rational) fitting problem, which is a known result; see Corollary 31. The total running time is linear in the input size.

Algorithm 1: coreset construction form (α, β)-approximation. As in Definition 6, the coreset consists of an (α, β)-approximation B of the input set P, and a weighted set (C, w). The weighted set (C, w) consists of a small sample $C \subseteq P$ and its weights function $w : C \to (0, \infty)$. As, e.g., in [22], the probability of choosing a point $p \in P$ to the sample C is proportional to its distance to its projection onto B. Hence, a point $(x, y) \in P$ whose y-value is far (not approximated well) from B would be sampled with high probability, but a point that is close to B will probably not be sampled. The weight $w(p)$ of a point is inverse proportional to its probability to be chosen, so that the sum of distances to any query (rational function) is the same as its expectation.

Coresets overview. In the explanation above, there where a number of different coresets:

(i) A sampling based coreset that is used to compute the restricted coreset above.

(ii) The restricted coreset computed in Algorithm 3, which assumes that the input signal consists of consecutive (α, β)-approximations, that approximates only specific rational functions, which are 2^k-bounded over the fist coordinate of the input signal; see Definition 7.

(iii) Our RFF coreset, i.e. ε-coreset as in Definition 6. This is our final goal, which use the previous coresets as sub-routines.

3.3 RFF coreset using a (α, β)-approximation

Algorithm 1 gets as input a bicretiria approximation of the RFF problem for some given input n-signal P; see Definition 5. The algorithm utilizes this rough approximation in order to compute, in near-linear time, an ε-coreset as in Definition 6 via sensitivity sampling. The formal statement is given in Lemma 8. This algorithm is a modified version of the algorithm presented for the k-segments problem in [22].

Algorithm 1: SAMPLE-CORESET$\big(B, \lambda\big)$; Lemma 8.

Input : A bicirteria approximation $B := \big\{(P_1, q_1), \cdots, (P_\beta, q_\beta)\big\}$ of some n-signal P; see Definition 5.
–An integer $\lambda \geq 1$ for the sample size.

Output: A tuple (B, C, w), where B is a bicriteria approximation of P, and (C, w) is a weighted set.

$c := \sum_{i=1}^{\beta} \ell(P_i, q_i)$

if $c \in \{0, \infty\}$ **then**

  Let $w : \mathbb{R}^2 \to \{0\}$ such that for every $p \in \mathbb{R}^2$ we have $w(p) = 0$.

  return $\big(B, \emptyset, w\big)$.

$s(p) := D(q_i, p)/c$ for every $i \in [\beta]$ and every $p \in P_i$.

Pick a sample $S \subset P$ of λ points from P, where S is a multi-set and each point $p \in S$ is sampled i.i.d. with probability $s(p)$; observe that there might be repetitions in S.

Set $r(p)$ as the number of repetitions of p in the multi-set S, for every $p \in P$.

$w(p) := r(p)/\big(\lambda \cdot s(p)\big)$ for every $p \in S$.

$S' := \{(x, \text{ratio}(q_i, x)) \mid i \in [\beta], (x, y) \in P_i \cap S\}$ `// project the labels of every set` $P_i \cap S$ `onto` q_i `and take their union.`

for *every* $i \in \{1, \cdots, \beta\}$ **do**

  $w\Big(\big(x, \text{ratio}(q_i, x)\big)\Big) := -w(p)$ for every $p \in S \cap P_i$

$C := S \cup S'$

return $\big(B, C, w\big)$.

The following lemma states the desired properties of Algorithm 1; see Lemma 21 for its proof.

Lemma 8. *Let $B := \big\{(P_1, q_1), \cdots, (P_\beta, q_\beta)\big\}$ be an (α, β)-approximation of some n-signal P, for*

some $\alpha > 1$; see Definition 5. Put $\varepsilon, \delta \in (0, 1/10]$, and let

$$\lambda \geq \frac{c^*}{\varepsilon^2}\alpha\big(k^2 \log \alpha + \log(1/\delta)\big)$$

be an integer, where $c^* \geq 1$ is a constant that can be determined from the proof. Let (B, C, w) be the output of a call to SAMPLE-CORESET(B, λ)*; see Algorithm 1. Then, Claims (i)–(ii) hold as follows:*

(i) The size for representing $\big(B, C, w\big)$ is in $O\left(\lambda + \beta k\right)$.

(ii) With probability at least $1 - \delta$, we have that (B, C, w) is an ε-coreset of P; see Definition 6.

The main remaining challenge is to compute the bicriteria approximation for the RFF, which is the main focus and main contribution of this work.

3.4 (α, β)-approximation

Optimal solution ($\alpha = \beta = 1$) in polynomial time. Using previous work [54] regarding the fractional polynomial problem it can be proven that, given any set $P \subset \mathbb{R}^2$ of points, we can compute in $(2kn)^{O(k)}$ time the optimal fitting rational function of degree k to the points, i.e., the rational function that minimizes (3.1) in Definition 4; see Lemma 28.

Efficient $(1, \beta)$-approximation for large β. Using the polynomial time optimal solution above, we can compute a $(1, \beta)$-approximation to a general signal P efficiently, for some $\beta \sim n$. This is by partitioning the input into β consecutive sets, and apply the optimal solution to each subset, which is relatively fast as each set is very small. The approximation is then the union of these solutions. This is formalized in Algorithm 2 and its corresponding analysis.

Overview of Algorithm 2. Algorithm 2 takes as input an n-signal P and an integer $\beta \geq 1$. It aims to partition P into $\psi \in \Theta\left(\beta\right)$ sets $P_1, P_2, \cdots, P_\psi$ of the same size, and to compute, for every such set P_i, the query $q_i \in \left(\mathbb{R}^k\right)^2$ that minimizes the RFF loss $\ell(P_i, q)$ for this set. The algorithm outputs the sets P_i in the partition of P, each equipped with its optimal query q_i.

As the time to compute the optimal query for each set in the partition of P depends polynomially on the size of the set, we need those sets to be small. Unfortunately, this implies that, we must plug a large value of β. To this end, this algorithm on its own does not suffice in order to compute the desired (α, β)-approximation with small values of both α and β. However, this algorithm is still utilized in Algorithm 5 as some sort of initialization.

Algorithm 2: BATCH-APPROX(P, β); Lemma 9.

Input : An n-signal P, where n is a power of 2, and an integer $\beta \geq 1$.

Output: An ordered set B of size $\psi \in O(\beta)$ that contains $(B_1, \cdots, B_\psi)$, where every B_i is a $(1,1)$-approximation (i.e., optimal) of a set P_i in some consecutive partition $\{P_1, \cdots, P_\psi\}$ of P; see Definition 5.

1 Compute a partition $\{P_1, \cdots, P_\psi\}$, where $\psi \in \left[\lfloor \beta/2 \rfloor, \beta\right]$, of P, where $|P_1| = \cdots = |P_\psi| = 2^m$, for some integer $m \geq 1$.

2 For every $i \in [\psi]$, let $B_i := \{(P_i, q_i)\}$ be a $(1,1)$-approximation of P_i, i.e. $q_i \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell(P_i, q)$; see Lemma 28.

3 Set $B := (B_1, \cdots, B_\psi)$.

4 **return** B.

The following lemma states the desired properties of Algorithm 2; see Lemma 29 for its proof.

Lemma 9. *Let P be an n-signal, where n is a power of 2. Let $\beta \geq 1$ be an integer, and $B := (B_1, \cdots, B_\psi)$, where $\psi \in \left[\lfloor \beta/2 \rfloor, \beta\right]$, be the output of a call to* BATCH-APPROX(P, β)*; see Algorithm 2. Identify $\{(P_i, q_i)\} := B_i$, for every $i \in [\psi]$. Then, $B' := \{(P_1, q_1), \cdots, (P_\psi, q_\psi)\}$ is a $(1, \beta)$-approximation of P; see Definition 5. Moreover, the output of the call to* BATCH-APPROX(P, β) *can be compute in $n \cdot (2kn/\beta)^{O(k)}$ time.*

3.5 From larger to smaller values of β

Algorithm 4 gets B an (α, β)-approximation of $\mathcal{P}$ where β is large, and returns an (α', β')-approximation with $\beta' < \beta$ but a larger $\alpha' > \alpha$. This is obtained by computing an approximation to the projection of $\mathcal{P}$ onto B. This is implemented in Algorithm 4, which calls Algorithm 3.

Overview of Algorithm 3. The input to the algorithm is $\mathcal{P}_i$ the projection of an interval of an n-signal onto some rational function. This rational function is its $(0,1)$-approximation; see Definition 5. The algorithm also receives an integer $\lambda \geq 1$ which controls the size of the output coreset. The algorithm computes a (restricted) coreset for $\mathcal{P}_i$, that approximates only rational functions that are 2^k-bounded over the first coordinate of the input interval; see Definition 7 and Lemma 10. Algorithm 3 computes this restricted coreset as mentioned in section 3.2.

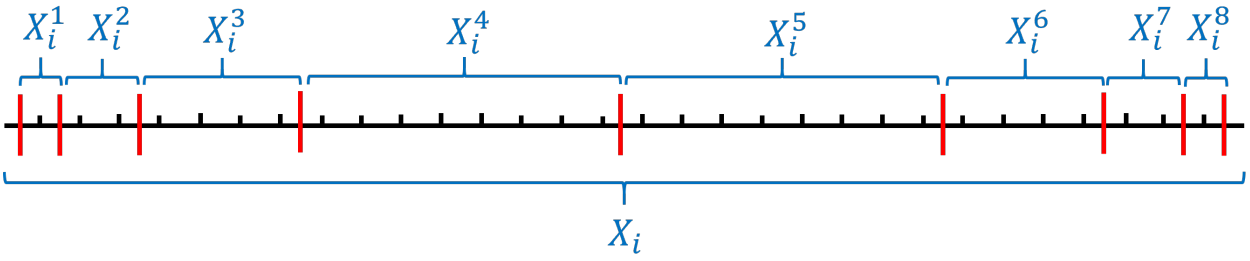

Figure 3-1: A partition $\{X_i^1, \cdots, X_i^8\}$ of a set $X_i = [30]$ into consecutive sets of exponentially increasing size; see Line 6 of Algorithm 3.

The following lemma states the desired properties of Algorithm 3; see Lemma 38 for its proof.

Lemma 10. *Let P be an interval of an n-signal which is projected onto some $q \in \left(\mathbb{R}^k\right)^2$, i.e., $B := (P, q)$ is a $(0,1)$-approximation B of P; see Definition 5. Let X be the first coordinate of P, i.e., $X := \{x \mid (x,y) \in P\}$. Put $\varepsilon, \delta \in (0, 1/10]$, and let*

$$\lambda \geq \frac{c^*}{\varepsilon^2} \cdot (4^k k^2 + 1)\left(k^2 \log(4^k k^2 + 1) + \log\left(\frac{k \log n}{\delta}\right)\right),$$

be an integer, where $c^* \geq 1$ is a constant that can that can be determined from the proof. Let (S, w) be the weighted set that is returned by a call to LIMITED-CORESET(B, λ)*; see Algorithm 3. Then $|S| \in O\left(k\lambda \cdot \log n\right)$ and, with probability at least $1 - \delta$, for every $q' \in \left(\mathbb{R}^k\right)^2$ that is 2^k-bounded over X (see Definition 7), we have*

$$|\ell(P, q') - \ell\left((S, w), q'\right)| \leq \varepsilon \cdot \ell(P, q'). \tag{3.2}$$

Algorithm 3: LIMITED-CORESET$(\{(P,q)\},\lambda)$; Lemma 10.

Input : An interval of an n-signal P such that $\ell(P,q)=0$ for some $q\in\left(\mathbb{R}^k\right)^2$. That is, $\{(P,q)\}$ a $(0,1)$-approximation of P; see Definition 5. This is represent by $B:=\{(P,q)\}$, which is a $(0,1)$-approximation B of P; see Definition 5.
– An integer $\lambda\geq 1$.

Output: A weighted set (S,w), i.e., $S\subset\mathbb{R}^2$ and $w:S\to\mathbb{R}$; see Section 3.1.

$X:=\{x\mid(x,y)\in P\}$, i.e., X is the union over the first coordinate of every pair in P.
Identify $(c,c'):=q$.
Let $\{X_1,\cdots,X_\eta\}$ be a consecutive partition of X into $\eta\in O(k)$ sets, such that for every $i\in[\eta]$ the function $f(x)=|1+x\cdot\text{poly}(c',x)|$ is monotonic over $\big[\min(X_i),\max(X_i)\big)$; see Lemma 36.
$S:=\emptyset$
for *every* $i\in[\eta]$ **do**
  Let $\{X_i^1,\cdots,X_i^{m_i}\}$ be a consecutive partition of X_i into $m_i\in\Theta\big(\log(|X_i|)\big)$ sets such that for every $j\in[m_i]$ we have $|X_i^j|=2^{\min\{j-1,m_i-j\}}$.
  `// See Fig. 3-1.`
  for *every* $j\in[m_i]$ **do**
    Let $s:X_i^j\to(0,\infty)$ such that $s(x)\geq\sup_c\dfrac{|\text{poly}(c,x)|}{\sum_{x'\in X_i^j}|\text{poly}(c,x')|}$ for every $x\in X_i^j$, and $\sum_{x\in X_i^j}s(x)\in O(k^2)$ where the supremum is over $c\in\mathbb{R}^{2k+1}$ such that $|\text{poly}(c,x')|>0$; see Corollary 31.
    Set $s'(x):=\dfrac{s(x)}{\sum_{x'\in X_i^j}s(x')}$ for every $x\in X_i^j$.
    $P_i^j:=\big\{\big(x,\text{ratio}(q,x)\big)\mid x\in X_i^j\big\}$ `// see Definition 4.`
    Pick a sample S_i^j of λ i.i.d. points from P_i^j, where each $(x,y)\in P_i^j$ is sampled with probability $s'(x)$.
    $S:=S\cup S_i^j$
    Set $w(p):=1/\big(\lambda\cdot s'(x)\big)$ for every $p=(x,y)\in S$.
return (S,w).

Overview of Algorithm 4. The input for the algorithm is an interval P of n-signal which is projected onto some set of (α,β)-approximations. This projection is represented by the set B, where each element $B_i\in B$ is a $(0,\beta)$-approximation for some P_i, and $P_1,\cdots,P_{|B|}$ is a consecutive partition of P. The algorithm also receives a parameter $\lambda\geq 1$ which controls the size of the output coreset, and a parameter δ which controls the trade-off between the running time and robustness. The algorithm returns a (α,β)-approximations B' of P, whose size is smaller than $\sum_{i=1}^{|B|}|B_i|$. The algorithm runs in $O(|P|^{1+o(1)})$ time.

This (α, β)-approximation is computed as mentioned in Section 3.2.

The following lemma states the desired properties of Algorithm 4; see Lemma 40 for its proof.

Lemma 11. *Let $B := \{B_1, \cdots, B_\beta\}$, where each $B_i \in B$ is an $(0, r_i)$-approximation of P_i, i.e. P_i is projected onto B_i, and $\{P_1, \cdots, P_\beta\}$ is an equally-sized consecutive partition of some interval of an n-signal P; see Fig. 3-2 and Definition 5. Put $\varepsilon, \delta \in (0, 1/10]$, and let*

$$\lambda \geq \frac{c^*}{\varepsilon^2}(4^k k^2 + 1)\left(k^2 \log_2(4^k k^2 + 1) + \log_2\left(\frac{kn}{\delta}\right)\right)$$

be an integer, where $c^* \geq 1$ is a constant that can be determined from the proof. Let B' be the output of REDUCE$(B, \lambda, 6k-3)$*; see Algorithm 4. With probability at least $1 - \delta$, we have that B' is a $(1 + 10\varepsilon, \beta^*)$-approximation of P for $\beta^* \in O\left(k \cdot \max_{i \in [\beta]} r_i\right)$; see Definition 5.*

Moreover, for $\varepsilon = 1/10$ we have that the running time of the call to REDUCE$(B, \lambda, 6k-3)$ *is in $|P| \cdot \beta^{6k-3} \cdot 4^{O(k^2)} \left(\log(n/\delta)\beta'\right)^{O(k)}$, where $\beta' = \sum_{i=1}^{\beta} r_i$.*

Note that the value of λ in the above lemma is different than in Lemma 10. This is since Algorithm 3 would be called as a subroutine at most n times, and as such we need to adjust the failure probability for each of the n usages of Lemma 10 to δ/n.

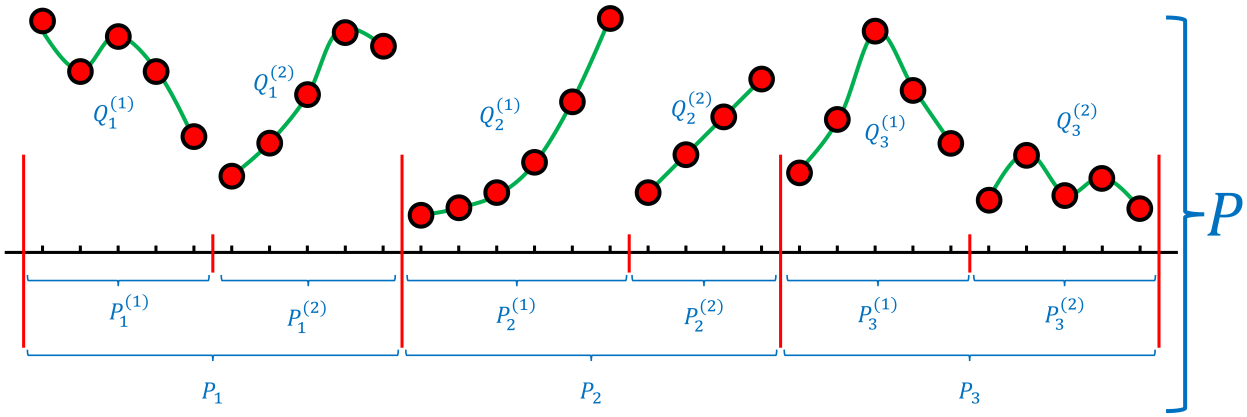

Figure 3-2: Illustration of the input to Algorithm 4. The set $\{P_1, P_2, P_3\}$ is an equally-sized consecutive partition of an n-signal P, where for every $i \in \{1, 2, 3\}$ the set P_i is projected onto an (α, β)-approximation $\left\{B_i^{(1)}, B_i^{(2)}\right\} := \left\{\left(P_i^{(1)}, Q_i^{(1)}\right), \left(P_i^{(2)}, Q_i^{(2)}\right)\right\}$.

Algorithm 4: REDUCE(B, λ, Λ); Lemma 11.

Input : A set $B := \{B_1, \cdots, B_\beta\}$, where each $B_i \in B$ is an $(0, r_i)$-approximation of P_i, i.e. P_i is projected onto B_i, and $\{P_1, \cdots, P_\beta\}$ is an equally-sized consecutive partition of P, some interval of an n-signal; see Fig. 3-2 and Definition 5. Integers $\lambda \geq 1$ and $\Lambda \geq 0$.

Output: A (α, β) approximation B' of P; see Definition 5.

$\ell^* := \infty$; $B' := \bigcup_{i=1}^{\beta} B_i$

for *every* $B_i \in B$ **do**

  Identify $\left\{B_i^{(1)}, \cdots, B_i^{(r_i)}\right\} := B_i$.

  for *every* $B_i^{(j)} \in B_i$ **do**

    $\left(S_i^{(j)}, w_i^{(j)}\right) :=$ LIMITED-CORESET $\left(\left\{B_i^{(j)}\right\}, \lambda\right)$ `// see Algorithm 3.`

for *every set* $G \subset \{1, \cdots, \beta\}$ *of size* $|G| = \beta - \Lambda$ **do**

  $S_G := \emptyset$.

  for *every* $i \in G$ *and* $B_i^{(j)} \in B_i$ **do**

    Set $w_G(p) := w_i^{(j)}(p)$ for every $p \in S_i^{(j)}$.

    $S_G := S_G \cup S_i^{(j)}$

  Set $q_G \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell\big((S_G, w_G), q\big)$; see Definition 4 and Lemma 28.

  for *every* $i \in G$ **do**

    $\ell_i := \ell(P_i, q_G)$

  Set $G' \subset G$ to be the union of the $6k - 3$ indices $i \in G$ with the largest value ℓ_i. Ties broken arbitrarily.

  if $\left(\sum_{i \in G \setminus G'} \ell_i\right) < \ell^*$ **then**

    $\ell^* := \left(\sum_{i \in G \setminus G'} \ell_i\right)$ `// update smallest loss`

    Set $\{R_1, \cdots, R_\gamma\}$ to be the smallest partition of $G \setminus G'$ such that for every $i \in [\gamma]$ we have $G' \cap [\min(R_i), \max(R_i)] = \emptyset$, and for any $i, j \in [\gamma]$, where $i \neq j$, we have $R_j \cap [\min(R_i), \max(R_i)] = \emptyset$.

    `// Via simple greedy partition; see Fig 3-3`

    $P_i' := \big\{\big(x, \text{ratio}(q, x) \mid (x, y) \in P_i\big)\big\}$ for every $i \in G$ `// the projection of` P_i `onto` q.

  $P_i^* := \bigcup_{\psi \in R_i} P_i'$, for every $i \in [\gamma]$

  `// Union of all the sets` P_ψ' `with index` ψ `in` R_i.

  $B' := \big\{\big(P_1^*, q_G\big), \cdots, \big(P_\gamma^*, q_G\big)\big\} \cup \left\{B_i^{(j)} \in B_i \mid i \in G' \cup \big([\beta] \setminus G\big)\right\}$

return B'.

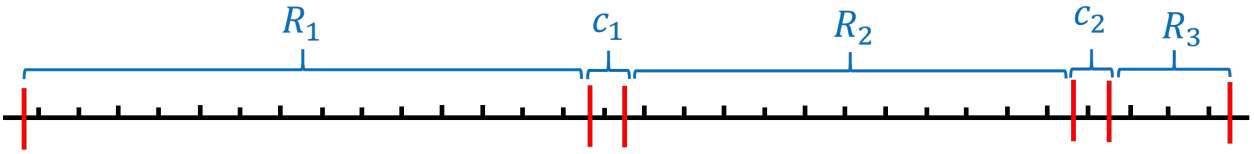

Figure 3-3: A set $G = [30]$ of indices (black ticks), a set $G' = \{c_1, c_2\} \subseteq G$ of 2 indices, and a partition $R_1 \cup R_2 \cup R_3 = G \setminus G'$ of the indices not in G, as described in Line 17 of Algorithm 4.

3.6 Wrapping it all together

Overview of Algorithm 5. Algorithm 5, which is the main algorithm in this work, takes as input an n-signal P, the desired approximation error $\varepsilon \in (0, 1/10]$ and an upper bound on the allowed failure probability $\delta \in (0, 1/10]$; see Definition 6. The algorithm runs in $O\left(n^{1+o(1)}\right)$ time and returns a coreset (B, C, w) with size for representing in $O\left(n^{o(1)}\right)$, for every constant ε and δ above. The algorithm first computes in Line 4 a $(1, \tilde{\beta})$-approximation for large $\tilde{\beta}$, i.e., a set of $\tilde{\beta}$ rational functions, each corresponds to some distinct subset of the input. Those subsets can be regarded as the leaves of a tree. In Lines 5–10, the algorithm then gradually combines β consecutive such subsets and computes a (α, β)-approximation for their union. It thus builds a tree of (α, β)-approximations with gradually decreasing β values (starting from $\beta = \tilde{\beta}$). The algorithm then utilizes in Line 12 the final computed (α, β)-approximation in order to compute a sampling based coreset via Algorithm 1.

The following theorem states the properties of Algorithm 5. See Theorem 42 for its proof.

Theorem 12. *Let P be an n-signal, for $n \geq 2k$ that is a power of 2, and put $\varepsilon, \delta \in (0, 1/10]$. Let (B, C, w) be the output of a call to* CORESET$(P, k, \varepsilon, \delta)$*; see Algorithm 5. With probability at least $1 - \delta$, (B, C, w) is an ε-coreset of P; see Definition 6. Moreover, (B, C, w) can be computed in $2^{O(k^2)} \cdot n \cdot n^{O(k)/\log\log(n)} \cdot \log(n)^{O(k\log(k))} \cdot \log(1/\delta)^{O(k)}$ time and $k^{O(1)} \cdot \log(n)^{O(1)+\log k} \cdot \log(1/\delta)/\varepsilon^2$ space. In particular, if k is constant then the running time is in $O\left(n^{1+o(1)}\delta^{o(1)}\right)$ and the space is in $O\left((n\delta)^{o(1)}/\varepsilon^2\right)$.*

Algorithm 5: CORESET$(P, k, \varepsilon, \delta)$; Theorem 12.

Input : An n-signal P, where $n \geq 2k$ is a power of 2, an integer $k \geq 1$, and $\varepsilon, \delta \in (0, 1/10]$ that represents approximation error and failure probability, respectively.

Output: A tuple (B, C, w), where B is a (α, β)-approximation of P, and (C, w) is a weighted set.

$\beta := \lceil n^{1/\log\log(n)} \rceil$; $\tilde{\beta} = \lceil n/\beta \rceil$; $\Lambda = 6k - 3$

Set $c^* \geq 1$ to be a constant that can be determined from the proof of Theorem 12.

$$\lambda_1 := \left\lceil c^*(4^{k+1}k^2 + 1)\left(k^2 \log(4^{k+1}k^2 + 1) + \log\left(\frac{kn}{\delta}\right)\right)\right\rceil$$

$B := (B_1, \cdots, B_{|B|}) :=$ BATCH-APPROX$(P, \tilde{\beta})$ `// see Algorithm 2.`

while $|B| > 2\beta$ **do**

  Set $\left\{B'_1, \cdots, B'_{|B|/\psi}\right\}$ to be a consecutive partition of B into equally-sized sets, each of size $\psi \in [\beta, 2\beta]$, a power of 2; i.e., $B'_i := \{B_a, \cdots, B_{a+\psi}\}$ for some $a \in [|B|]$ and every $i \in [|B|/\psi]$.

  Set $B_i :=$ REDUCE$(B'_i, \lambda_1, \Lambda)$ for every $i \in \{1, \cdots, \psi\}$; see Algorithm 4.
  `// Reduce every set` B_i `in the partition of` B.

  $B := (B_1, \cdots, B_\psi)$

$B' := \{B_1, \cdots, B_{|B|}\}$ `//` B' `is an (un-ordered) set.`

$B :=$ REDUCE(B', λ_1, Λ).

$$\lambda_2 := \left\lceil \frac{c^*}{\varepsilon^2} \cdot \log(n) \cdot \left(k^2 \log\log(n) + \log\left(\frac{n}{\delta}\right)\right)\right\rceil$$

$(B, C, w) :=$ SAMPLE-CORESET(B, λ_2) `// see Algorithm 1.`

return (B, C, w).

The main structure of Algorithm 5 can be illustrated using the following figure.

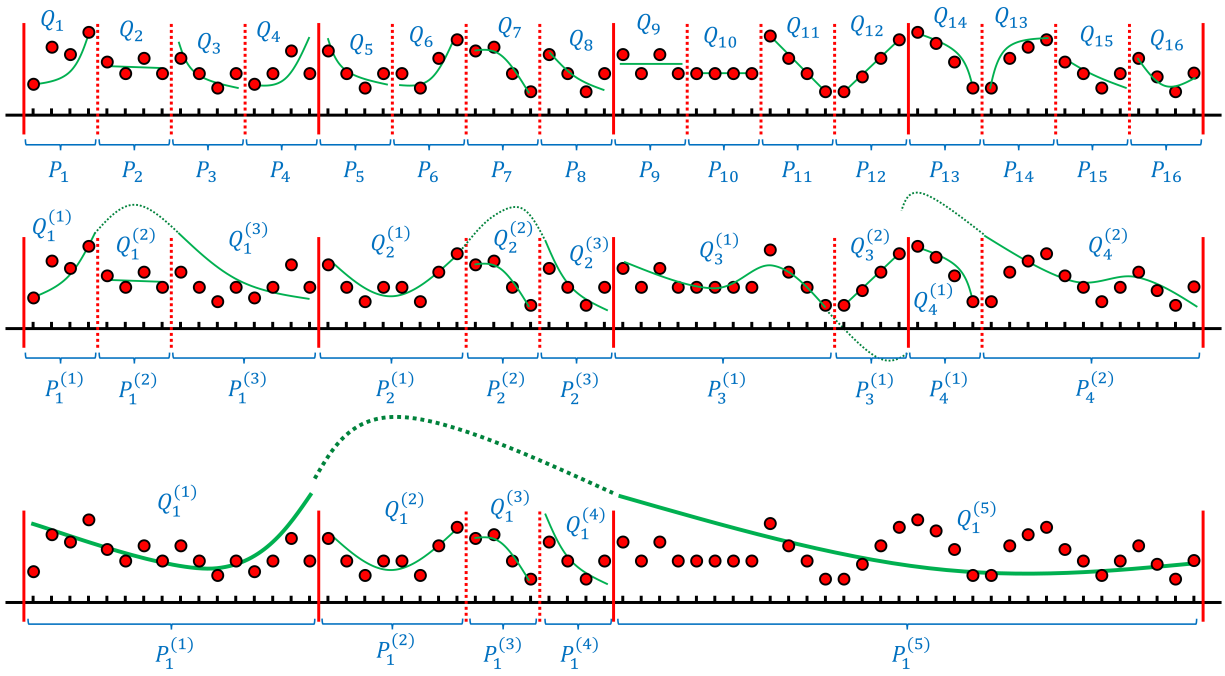

Figure 3-4: Illustration for Algorithm 5. ***(Top)*** *In Line 4, an input n-signal P (black ticks and red dots) and its partition $\{P_1, \cdots, P_{16}\}$ into $\psi = 16$ sets via a call to Algorithm 2. This call also computes $(1,1)$-approximation $B_i = \{(P_i, Q_i)\}$ for every P_i (green curves), see Definition 5. In Line 6, the set $\{B_1, \cdots, B_{16}\}$ is partitioned into $B = B'_1 \cup B'_2 \cup B'_3 \cup B'_4$, where each such set contains 4 elements from B.* ***(Middle)*** *In Line 7, for every $i \in [4]$ we set $B_i = \left\{\left(P_i^{(1)}, Q_i^{(1)}\right), \cdots, \left(P_i^{(|B_i|)}, Q_i^{|B_i|}\right)\right\}$ as the output of a call to* REDUCE(B'_i)*. For every $i \in [3]$ the projection of every $\bigcup_{i=1}^{|B_i|} P_i^{(j)}$ onto B_i is denoted in green.* ***(Bottom)*** *The process above is repeated, with $B := \{B_1, B_2, B_3, B_4\}$. B is therefore partitioned into sets of size $\beta = 4$, i.e., only 1 such set B, and then* REDUCE(B) *is called, which reduces the size of B to 5.*

3.7 Fast practical heuristic

Unfortunately the running time of the algorithms is still large. Therefore,we suggest a heuristic to run on top of our coreset. We later prove that, under some assumptions, this heuristic gives a constant factor approximation. For this heuristic we need the following definitions.

Definition 13. *Let S be a set of $2k$ points on the plane. We define* SOLVER(S) *as an arbitrary $(c, c') \in \left(\mathbb{R}^k\right)^2$ that satisfies $\ell(S, q) = 0$ if there is such a pair, otherwise it is empty.*

In Lemma 49, we prove, that if $\left|\{x \cdot y \mid (x, y) \in S\}\right| = 2k$, then SOLVER is never empty and that it can be computed in $O\left(k^3\right)$ time. In our companion code, we sample G directly from P.

Algorithm 6: FAST-CENTROID-SET$\left(P, \beta\right)$; Lemma 50.

Input : A finite set $P \subset \mathbb{R}^2$ of at least $2k$ points, where $\forall S \subset P, |S| = 2k : \left|\{x \cdot y \mid (x, y) \in S\}\right| = 2k$, and an integer $\beta \geq 1$.

Output: A set $G \subset \left(\mathbb{R}^k\right)^2$ of size $|G| \leq \beta$.

$G := \{S \subseteq P \mid |S| = 2k\}$.

if $|G| := \binom{|P|}{2k}$ **then**

    // $|G| \geq \beta$

    return $\bigcup_{S \in G}$ SOLVER(S)// see Definition 13.

Pick a sample $G' \subset G$ of $|G'| = \beta$ sets of points, where each set $S \in G'$ of points is sampled i.i.d. and uniformly at random from G.

return $\bigcup_{S \in G'}$ SOLVER(S)// see Definition 13.

Chapter 4

Analysis

4.1 Algorithm 1; Coreset given an (α, β)-approximation

The coreset construction that we use in Algorithm 1 is a non-uniform sample from a distribution, which is known as sensitivity, that is based on the (α, β)-*approximation* defined in Definition 5. To apply the generic coreset construction we need two ingredients:

(i) A bound on the dimension induced by the query space ("complexity") that corresponds to our problem as formally stated and bounded in subsection 4.1.1. This bound on the dimension induced by the query space determines the required size of the random sample picked in Algorithm 1.

(ii) A bound on the sensitivity as formally stated and bounded in the proof of Lemma 21. This bound on the sensitivity determines the required size of the random sample that is picked in Algorithm 1.

4.1.1 Bound on the dimension of the query space

We first define the classic notion of VC-dimension, which is used in Theorem 8.14 in [3], and is usually related to the PAC-learning theory [34].

Definition (VC-dimension [36]). *Let $F \subset \left\{\mathbb{R}^d \to \{0,1\}\right\}$ and let $X \subset \mathbb{R}^d$. Fix a set $S = \{x_1, \cdots, x_n\} \subset X$ and a function $f \in F$. We call $S_f = \{x_i \in S \mid f(x_i) = 1\}$ the induced subset of S by f. A subset $S = \{x_1, \cdots, x_n\}$ of X is shattered by F if $|\{S_f \mid f \in F\}| = 2^n$. The VC-dimension of F is the size of the largest subset of X shattered by F.*

Theorem 14. *Let h be a function from $\mathbb{R}^m \times \mathbb{R}^d$ to $\{0,1\}$, and let*

$$\mathcal{H} = \{h_\theta : \mathbb{R}^d \to \{0,1\} \mid \theta \in \mathbb{R}^m\}.$$

Suppose that h can be computed by an algorithm that takes as input the pair $\theta \in \mathbb{R}^m \times \mathbb{R}^d$ and returns $h_\theta(x)$ after no more than t of the following operations:

- *the arithmetic operations $+, -, \times$, and $/$ on real numbers,*
- *jumps conditioned on $>, \leq, <, \geq, =$, and $\neq$ comparisons of real numbers, and*
- *output 0, 1.*

Then the VC-dimension of $\mathcal{H}$ is $O\big(m^2 + mt\big)$.

For the sample mentioned in the start of Section 4.1 we utilize the following generalization of the previous definition of VC-dimension. This is commonly referred to VC-dimension, but to differentiate this definition from the previous, and to be in line with the notations in [19] we abbreviate it to *dimension*. This is the dimension induced by the query space which would be assigned in Theorem 20 to obtain the proof of Algorithm 1.

Definition (range space [21]). *A range space is a pair (L, ranges) where L is a set, called ground set and ranges is a family (set) of subsets of L.*

Definition (dimension of range spaces [21]). *The dimension of a range space (L, ranges) is the size $|S|$ of the largest subset $S \subseteq F$ such that*

$$|\{S \cap \text{range} \mid \text{range} \in \text{ranges}\}| = 2^{|S|}.$$

Definition 15 (range space of functions [21, 26, 20])**.** *Let F be a finite set of functions from a set $\mathcal{Q}$ to $[0, \infty)$. For every $Q \in \mathcal{Q}$ and $r \geq 0$, let* $\text{range}(F, Q, r) = \{f \in F \mid f(Q) \geq r\}$.
Let $\text{ranges}(F) = \{\text{range}(F, Q, r) \mid Q \in \mathcal{Q}, r \geq 0\}$.
Finally, let $\mathcal{R}_{\mathcal{Q},F} = (F, \text{ranges}(F))$ be the range space induced by $\mathcal{Q}$ and F.

In the following lemma, which is inspired by Theorem 12 in [36], we bound the VC-dimension which would be assigned in Theorem 20 to obtain the proof of Algorithm 1.

Lemma 16. *Let $B = \{(P_1, q_1), \cdots, (P_\beta, q_\beta)\}$ be an (α, β)-approximation of some n-signal $P = \{(1, y_1), (2, y_2), \cdots, n, y_n)\}$; see Definition 5. Let $f : P \times \left(\mathbb{R}^k\right)^2 \to [0, \infty)$, be the function that maps every $p = (x, y) \in P$, where $p \in P_i$, and any $q \in \left(\mathbb{R}^k\right)^2$ to $f(x) = D\big(q, \text{ratio}(q_i, x)\big)$. For every $i \in [n]$ let $f_i : \left(\mathbb{R}^k\right)^2 \to [0, \infty)$ denote the function that maps every $q \in \left(\mathbb{R}^k\right)^2$ to $f_i(q) = f(q, (i, y_i))$. Let $F = \{f_1, \ldots, f_n\}$. The dimension of the range space $\mathcal{R}_{\left(\mathbb{R}^k\right)^2,F}$ that is induced by $\left(\mathbb{R}^k\right)^2$ and F is in $O(k^2)$.*

Proof. For every $(q, r) = \big(c, c'), r\big) \in \left(\mathbb{R}^k\right)^2 \times \mathbb{R}$, let $h_{(c|c'|r)} : \mathbb{R} \to \{0, 1\}$ that maps every $x \in [n]$ to $h_{(c|c'|r)}(x) = 1$ if and only if $f_i(q) \leq r$, and every $x \in \mathbb{R} \setminus [n]$ to $h_{(c|c'|r)}(x) = 0$. Let $\mathcal{H} = \{h_\theta \mid \theta \in \mathbb{R}^{2k+1}\}$. For every $c \in \mathbb{R}^k$ and any $x \in \mathbb{R}$ we can compute $\text{poly}(c, x)$ with $O(k)$ arithmetic operations on real numbers and jumps conditioned on comparisons of real numbers; see, for example, Horner's scheme [42], which is used in numpy's implementation of the method polyval [27]. Therefore, for every $i \in [n]$ and any $\theta \in \mathbb{R}^{2k+1}$, by the definition of D, we can calculate $h_\theta(i)$ with $O(k)$ arithmetic operations on real numbers and jumps conditioned on comparisons of real numbers. Hence, substituting $d := n$, $m := 2k + 1$, $h := h$, $\mathcal{H} := \mathcal{H}$, and $t \in O(k)$ in Theorem 14 yields that the VC-dimension of $\mathcal{H}$ is in $O(k^2)$. Hence, by the construction of $\mathcal{H}$ and the definition of range spaces in Definition 15, we have that the dimension of the range space $\mathcal{R}_{P,F}$ that is induced by P and F is in $O(k^2)$. □

4.1.2 Sensitivity of functions

For the self containment of the thesis we state previous work on sensitivity of functions.

Observe that the following is stated in a more general form than required in this section. This is since we would re-use the stated results in later parts for the restricted coreset, while in this section we bound the sensitivity to the projection onto a bicretiria; see Section 3 and Definition 5.

Definition 17 (query space [19])**.** *Let $P \subset \mathbb{R}^2$ be a finite non empty set. Let $f : P \times \left(\mathbb{R}^k\right)^2 \to [0,\infty)$ and $\text{loss} : \mathbb{R}^{|P|} \to [0,\infty)$ be a function. The tuple $(P, \left(\mathbb{R}^k\right)^2, f, \text{loss})$ is called a* query space*. For every $q \in \left(\mathbb{R}^k\right)^2$ we define the overall fitting error of P to q by*

$$f_{\text{loss}}(P,q) := \text{loss}\left(f(p,q)_{p\in P}\right) = \text{loss}\left(f(p_1,q),\ldots,f(p_{|P|},q)\right).$$

To emphasize that the following coreset is a subset of the input set, in contrast to ε-coreset as in Definition 6, we call it subset-ε-coreset. In Section 4.1.4 we prove that there is no such coreset for the RFF problem; see Definition 4.

Definition 18 (subset-ε-coreset [19])**.** *Let $(P, \left(\mathbb{R}^k\right)^2, f, \text{loss})$ be a query space as in Definition 17. For an approximation error $\varepsilon > 0$, the pair $S' = (S,u)$ is called an* subset-ε-coreset *for the query space $(P, \left(\mathbb{R}^k\right)^2, f, \text{loss})$, if $S \subseteq P, u : S \to [0,\infty)$, and for every $q \in \left(\mathbb{R}^k\right)^2$ we have*

$$(1-\varepsilon) f_{\text{loss}}(P,q) \leq f_{\text{loss}}(S',q) \leq (1+\varepsilon) f_{\text{loss}}(P,q).$$

Definition 19 (sensitivity of functions)**.** *Let $P \subset \mathbb{R}^2$ be a finite and non empty set, and let $F \subset \{P \to [0,\infty]\}$ be a possibly infinite set of functions. The* sensitivity *of every point $p \in P$ is*

$$S^*_{(P,F)}(p) = \sup_{f\in F} \frac{f(p)}{\sum_{p\in P} f(p)}, \tag{4.1}$$

where sup *is over every $f \in F$ such that the denominator is positive. The* total sensitivity *given a* sensitivity *is defined to be the sum over these sensitivities, $S^*_F(P) = \sum_{p\in P} S^*_{(P,F)}(p)$. The function $S_{(P,F)} : P \to [0,\infty)$ is a* sensitivity bound *for $S^*_{(P,F)}$, if for every $p \in P$ we have $S_{(P,F)}(p) \geq S^*_{(P,F)}(p)$. The* total sensitivity bound *is then defined to be $S_{(P,F)}(P) = \sum_{p\in P} S_{(P,F)}(p)$.*

The following theorem proves that a coreset can be computed by sampling according to sen-

sitivity of functions. The size of the coreset depends on the total sensitivity and the complexity (VC-dimension) of the query space, as well as the desired error ε and probability δ of failure.

Theorem 20 (coreset construction [19])**.** *Let*

- $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^2$ *be a finite and non empty set, and* $f : P \times \left(\mathbb{R}^k\right)^2 \to [0, \infty)$.
- $F = \{f_1, \ldots, f_n\}$*, where* $f_i(q) = f(p_i, q)$ *for every* $i \in [n]$ *and* $q \in \left(\mathbb{R}^k\right)^2$.
- d' *be the dimension of the range space that is induced by* $\left(\mathbb{R}^k\right)^2$ *and* F.
- $s^* : P \to [0, \infty)$ *such that* $s^*(p)$ *is the sensitivity of every* $p \in P$*, after substituting* $P = P$ *and* $F = \left\{f' : P \to [0, \infty] \mid \forall p \in P, q \in \left(\mathbb{R}^k\right)^2 : f'(p) := f(p, q)\right\}$ *in Definition 19, and* $s : P \to [0, \infty)$ *be the sensitivity bound of* s^*.
- $t = \sum_{p \in P} s(p)$.
- $\varepsilon, \delta \in (0, 1)$.
- $c > 0$ *be a universal constant that can be determined from the proof.*
- $\lambda \geq c(t+1)\big(d' \log(t+1) + \log(1/\delta)\big/\varepsilon^2$.
- $w : P \to \{1\}$*, i.e. a function such that for every* $p \in P$ *we have* $w(p) = 1$.
- (S, u) *be the output of a call to* CORESET-FRAMEWORK(P, w, s, λ) *(Algorithm 1 in [19]).*

Then the following holds

- *With probability at least* $1 - \delta$*,* (S, w) *is an subset-*ε*-coreset of size* $|S| \leq \lambda$ *for the query space* $(F, \left(\mathbb{R}^k\right)^2, f, \|\cdot\|_1)$*; see Definition 18.*

4.1.3 Correctness of Algorithm 1; Proof of Lemma 8

In the following lemma we prove Lemma 8, which proves that, given values that satisfy specific properties, we have that Algorithm 1 yields an ε-coreset; see Definition 6.

Lemma 21. *Let $B := \big\{(P_1, q_1), \cdots, (P_\beta, q_\beta)\big\}$ be an (α, β)-approximation of some n-signal P, for some $\alpha > 1$; see Definition 5. Put $\varepsilon, \delta \in (0, 1/10]$, and let*

$$\lambda \geq \frac{c^*}{\varepsilon^2}\alpha\big(k^2 \log \alpha + \log(1/\delta)\big)$$

be an integer, where $c^* \geq 1$ is a constant that can be determined from the proof. Let (B, C, w) be the output of a call to SAMPLE-CORESET(B, λ)*; see Algorithm 1. Then, Claims (i)–(ii) hold as follows:*

(i) The size for representing $\big(B, C, w\big)$ is in $O\left(\lambda + \beta k\right)$.

(ii) With probability at least $1 - \delta$, we have that (B, C, w) is an ε-coreset of P; see Definition 6.

Proof. We have (i) by the construction of Algorithm 1 and the definitions in the theorem. Let c as computed in the call to SAMPLE-CORESET(B, λ); see Algorithm 1. Since B is an (α, β)-approximation of P we have that $c \neq \infty$. If $c = 0$, then the theorem holds by the construction of Algorithm 1. Hence, we assume this is not the case. Let P' be the projection of P onto B, i.e., $P' := \big\{\big(x, \text{ratio}(q_i, x)\big) \mid i \in [\beta], (x, y) \in P\big\}$; see Definition 5. Let $\mathcal{Q}$ be the union of $q = (c, c') \in \big(\mathbb{R}^k\big)^2$ such that $1 + \text{poly}(c', x) \neq 0$ for every $(x, y) \in P$. Let $q = (c, c') \in \mathcal{Q}$. We have

$$\begin{aligned} &\left|\sum_{p\in P} D(q,p) - \sum_{p\in C}\Big(w(p)\cdot D(q,p)\Big) - \sum_{p\in P'} D(q,p)\right| = \\ &\sum_{p\in P} D(q,p)\cdot\left|\frac{\sum_{p\in P} D(q,p) - \sum_{p\in P'} D(q,p)}{\sum_{p\in P} D(q,p)} - \frac{\sum_{p\in C}\Big(w(p)\cdot D(q,p)\Big)}{\sum_{p\in P} D(q,p)}\right|, \end{aligned} \tag{4.2}$$

where the previous equality is by taking $\sum_{p\in P} D(q,p)$ out of the sum. Let $s : P \to [0, \infty)$ as defined in Line 8 of Algorithm 1 in the call to SAMPLE-CORESET(B, λ), i.e., for every $i \in [\beta]$ and any

$p \in P_i$ we have

$$s(p) = \frac{D(q_i, p)}{\sum_{i=1}^{\beta} \ell(P_i, q_i)}. \tag{4.3}$$

Let $i \in [\beta]$ and $s_i^* : P_i \to [0, \infty)$ such that for every point $p = (x, y) \in P_i$ we have

$$s_i^*(p) = \frac{\Big|D(q,p) - \big|\text{ratio}(q_i, x) - \text{ratio}(q, x)\big|\Big|}{\sum_{p \in P} D(q, p)}, \tag{4.4}$$

which, due to the definition of P' as projection of P onto B, is an upper bound on the contribution of every point in P_i to the sum in (4.2). Let $p = (x, y) \in P_i$, so that

$$\begin{aligned} s_i^*(p) =& \frac{\Big|D(q,p) - \big|\text{ratio}(q_i, x) - \text{ratio}(q, x)\big|\Big|}{\sum_{p \in P} D(q, p)} \\ &= \frac{\Big|\big|y - \text{ratio}(q, x)\big| - \big|\text{ratio}(q_i, x) - \text{ratio}(q, x)\big|\Big|}{\sum_{p \in P} D(q, p)} \\ &\leq \frac{\big|y - \text{ratio}(q_i, x)\big|}{\sum_{p \in P} D(q, p)} = \frac{D(q_i, p)}{\sum_{p \in P} D(q, p)} \leq \alpha \cdot s(p), \end{aligned} \tag{4.5}$$

where the first equality is by the definition of s^* from (4.4), the second equality is by the definition of D, the inequality is by the reverse triangle inequality, the third equality is by the definition of D, and the last equality is by (4.3) and the definition of B as an (α, β)-approximation of P.

Let $\tilde{s} : P \to [0, \infty)$ such that for every $p \in P$ we have $\tilde{s}(p) = \alpha \cdot s(p)$. By (4.5), for every $i \in [\beta]$, we have that $\tilde{s}$ is a sensitivity bound for s_i^*.

For every $p = (x, y) \in P_i, i \in [\beta]$ and any $q \in \left(\mathbb{R}^k\right)^2$ let $f(q, p) = D\big(q, \text{ratio}(q_i, x)\big)$. Let $F = \{f_1, \ldots, f_n\}$, where $f_i(y) = f(q, p)$ for every $p \in P_i, i \in [\beta]$ and $q \in \left(\mathbb{R}^k\right)^2$. Let $k^* \in O(k^2)$ be the dimension of the range space $\mathcal{R}_{P,F}$ from Lemma 16 when assigning P and B.

Substituting $\varepsilon := \varepsilon, \delta := \delta, \lambda := \lambda$, the query space $(P, \mathcal{Q}, F, \|\cdot\|_1)$, $d' \in O\left(k^2\right)$ the VC-

dimension induced by $\left(\mathbb{R}^k\right)^2$ and F from Lemma 16, the sensitivity bound $\tilde{s}$, and the total sensitivity $t = \alpha \sum_{p \in P} s(p) = \alpha$ in Theorem 20, combined with the construction of Algorithm 1, yields that with probability at least $1 - \delta$, for every $q \in \mathcal{Q}$, we have

$$\left| \frac{\sum_{p \in P} D(q,p) - \sum_{p \in P'} D(q,p)}{\sum_{p \in P} D(q,p)} - \frac{\sum_{p \in C} \Big(w(p) \cdot D(q,p) \Big)}{\sum_{p \in P} D(q,p)} \right| \leq \varepsilon. \tag{4.6}$$

Combining (4.6) and (4.2) proves the theorem. □

4.1.4 Lower bound.

In this section we prove that there is no subset-ε-coreset for the query space $\Big(P, \left(\mathbb{R}^k\right)^2, D, \|\cdot\|_1\Big)$, where $P \subset \mathbb{R}^2$ is an n-signal P and D is as Definition 4, as defined in Definition 17. This justifies our Definition 6 of coreset for RFF. The main idea in the following claim is illustrated in Fig. 4-1.

Claim 22 (A variant of Claim 5 in [55])**.** *For every integer $n \geq 2$ there is an n-signal P such that the following holds. For every $C \subseteq \mathbb{R}^2$, where $|C| < n$, there is $q \in \left(\mathbb{R}^k\right)^2$ such that $\sum_{p \in C} D(q,p) \in [0, \infty)$ and $\sum_{p \in P} D(q,p) = \infty$.*

Proof. Let $P = \{(1,0), \cdots, (n,0)\}$ and $C \subseteq \mathbb{R}^2$, where $|C| < n$. Put $(a,0) \in P, a > 0$ such that $C \cap \{(a,y) \mid y \in \mathbb{R}\} = \emptyset$; i.e., there is no point in C with x-value equal a. There is such a point since $|C| < n = |P|$. Let $c = (1,0,0,\cdots,0), c' = \left(-\frac{1}{a}, 0, 0, \cdots, 0\right)$, and $q = (c, c') \in \left(\mathbb{R}^k\right)^2$. Since $D\big((a,0), q\big) = \big|\text{ratio}(q,a) - 0\big| = \infty$ (observe that $1 + a\text{poly}(c', a) = 1 - a/a = 0$), and $(a,0) \in P$, we obtain $\sum_{p \in P} D(q,p) \geq D\big((a,0), q\big) = \infty$. On the other hand $1 - \frac{1}{a} \cdot x = 0$ only for $x = a$, therefore $\forall p \in C : D(q,p) \in [0, \infty)$, and since C is a finite set we obtain $\sum_{p \in C} D(q,p) \in [0, \infty)$. □

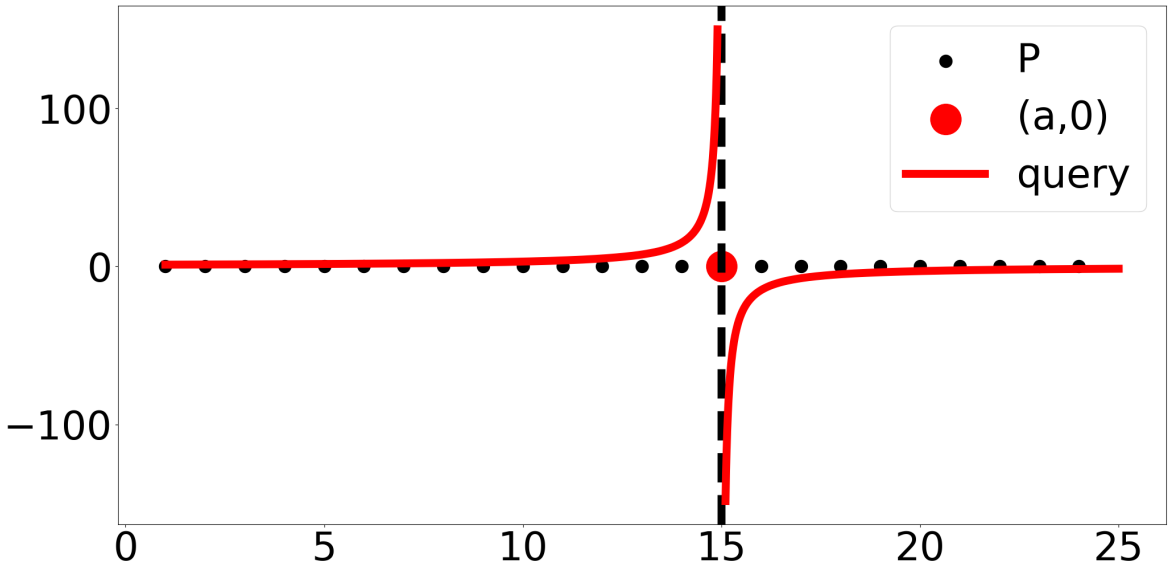

Figure 4-1: Visual illustration of the main idea behind Claim 22. The n-signal $P = \{(1,0),\cdots,(25,0)\}$ as in Claim 22 for $n = 25$ (black dots). Consider a subset C that contains all these points except a single one, say, the red point $(15,0)$. We can always find a rational function (query) whose sum of distances is close to zero for P, but close to ∞ for C, due to due its high change at the point $(15,0)$ that was not selected to C.

4.2 Inefficient solver

In the following section we will prove that we can solve the RFF problem from Definition 4 in $(2kn)^{O(k)}$ time as previously mentioned in Section 3.1.

The main idea is that, given an assignment of whether each point is below or above the best fitting rational function (to the loss in (3.1) from Definitions 4) the problem can be written as a fractional polynomial programming problem and solved in polynomial (in the input size) time; see [54]. Using previous work [38] we can bound the number of candidate assignments mentioned above to be polynomial in the input size. Hence, by constructing a semi tree we can find all the satisfiable assignments in time polynomial in the input size, which enables us to compute an optimal solution in polynomial time.

In the following section, for every $x \in \mathbb{R}$ let $\operatorname{sgn}(x) = 1$ if $x > 0$ and $\operatorname{sgn}(x) = -1$ otherwise.

Theorem 23 (Theorem 23 in [38])**.** *Let $f_1,\cdots,f_m$ be real polynomials in $k < m$ variables, each*

of degree at most $b \geq 1$. Then the number of sign sequences $(\operatorname{sgn}(f_1(x)), \cdots, \operatorname{sgn}(f_m(x)))$ over $x \in \mathbb{R}^d$ that consist of the terms $1, -1$ is at most $\left(\frac{4ebm}{k}\right)^k$.

To utilize this previous bound we state the following observation.

Observation 24. *Let $P = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subset \mathbb{R}^2$. Let $\mathcal{Q}$ be the union of $(c, c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \operatorname{poly}(c', x) \neq 0$ for every $(x, y) \in P$. There are $g_1, \cdots, g_n : \mathbb{R}^{2k} \to \mathbb{R}$ polynomial of degree in $O(n)$ with $2k$ variables such that for every $(c, c') \in \mathcal{Q}$ we have*

$$g_i(c \mid c') = \big(1 + x \cdot \operatorname{poly}(c', x)\big) \cdot \big(\operatorname{poly}(c, x) - y - y \cdot x \cdot \operatorname{poly}(c', x)\big).$$

Proof. Let $q := (c, c') \in \mathcal{Q}$. For every $(x, y) \in P$, by reorganizing the expression we have

$$\operatorname{sgn}\big(\operatorname{ratio}\big((c, c'), x\big) - y\big) = \operatorname{sgn}\Big(\big(1 + x \cdot \operatorname{poly}(c', x)\big) \cdot \big(\operatorname{poly}(c, x) - y - y \cdot x \cdot \operatorname{poly}(c', x)\big)\Big). \quad (4.7)$$

For every $i \in [n]$, let $g_i : \mathbb{R}^{2k} \to \mathbb{R}$ be a polynomial of degree in $O(n)$ with $2k$ variables that maps every $(c, c') \in \left(\mathbb{R}^k\right)^2$ to

$$g_i(c \mid c') = \big(1 + x \cdot \operatorname{poly}(c', x)\big) \cdot \big(\operatorname{poly}(c, x) - y - y \cdot x \cdot \operatorname{poly}(c', x)\big).$$

By (4.7) we have that $g_1, \cdots, g_n : \mathbb{R}^{2k} \to \mathbb{R}$ satisfy the observation. □

Using this observation and Theorem 23 we obtain the following result.

Corollary 25. *Let $P = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subset \mathbb{R}^2$. Let $\mathcal{Q}$ be the union of $(c, c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \operatorname{poly}(c', x) \neq 0$ for every $(x, y) \in P$. The number of sign sequences $\Big(\operatorname{sgn}\big(\operatorname{ratio}(q, x_1) - y_1\big), \cdots, \operatorname{sgn}\big(\operatorname{ratio}(q, x_n) - y_n\big)\Big)$ over every $q \in \mathcal{Q}$ is in $(2kn)^{O(k)}$.*

Proof. By Observation 24, let $g_1, \cdots, g_n : \mathbb{R}^{2k} \to \mathbb{R}$ be real polynomials of degree in $O(n)$ with $2k$ variables such that for every $q := (c, c') \in \mathcal{Q}$ we have

$$\operatorname{sgn}\big(\operatorname{ratio}(q, x_1) - y_i\big) = \operatorname{sgn}\big(g_i(c \mid c')\big). \quad (4.8)$$

By Claim 23, the number of sign sequences $(\operatorname{sgn}(g_1(x)), \cdots, \operatorname{sgn}(g_n(x))$ over $x \in \mathbb{R}^{2k}$ that consist of the terms $1, -1$ is in $(2kn)^{O(k)}$. Combining this with Equation (4.8) proves the Corollary. □

While, as stated before, the number of possible positions of every points is indeed polynomial, there is the problem of computing this set in polynomial time. We will solve this by utilizing previous work on polynomial programming mentioned in [54].

Observation 26. *Let $P = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subset \mathbb{R}^2$. Let $\mathcal{Q} \subset \left(\mathbb{R}^k\right)^2$ be the union of every $(c, c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \operatorname{poly}(c', x) \neq 0$ for every $(x, y) \in P$. For every vector $S \in \{0,1\}^n$ we can check in $(2kn)^{O(k)}$ the existence of $q \in \mathcal{Q}$ that satisfies $S = \Big(\operatorname{sgn}\big(\operatorname{ratio}(q, x_1) - y_1\big), \cdots, \operatorname{sgn}\big(\operatorname{ratio}(q, x_n) - y_n\big)\Big)$, i.e., satisfies the assignment by S.*

Proof. By Observation 24, let $g_1, \cdots, g_n : \mathbb{R}^{2k} \to \mathbb{R}$ be real polynomials of degree in $O(n)$ with $2k$ variables such that for every $q := (c, c') \in \mathcal{Q}$ we have

$$\operatorname{sgn}\big(\operatorname{ratio}(q, x_1) - y_i\big) = \operatorname{sgn}\big(g_i(c \mid c')\big).$$

Hence, for every assignment $S \in \{0,1\}^n$, there is $q \in Q$ such that

$$S = \Big(\operatorname{sgn}\big(\operatorname{ratio}(q, x_1) - y_1\big), \cdots, \operatorname{sgn}\big(\operatorname{ratio}(q, x_n) - y_n\big)\Big)$$

if and only if there is $x \in \mathbb{R}^d$ such that $S = \Big(g_1(x), \cdots, g_n(x)\Big)$, and the later can be written as polynomial programming and thus solved numerically in $(2kn)^{O(k)}$ time; since this equality can be written as sum of squares (SOS), see [54]. □

Using this, in the following lemma, we prove that we can generate all the satisfiable options for the function to be above the points or below the points in polynomial time and not exponential.

Lemma 27. *Let $P = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subset \mathbb{R}^2$. For every non empty $C \subset P$ let $Q(C) \subset \left(\mathbb{R}^k\right)^2$ be the union of every $(c, c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \operatorname{poly}(c', x) \neq 0$ for any $(x, y) \in C$. All the sign sequences $\Big(\operatorname{sgn}\big(\operatorname{ratio}(q, x_1) - y_1\big), \cdots, \operatorname{sgn}\big(\operatorname{ratio}(q, x_n) - y_n\big)\Big)$ over every $q \in Q(P)$ can be computed in $(2kn)^{O(k)}$ time.*

Proof. By Corollary 25, for every $C \subset P, |C| = m \geq 2k$ there are at most $m^{O(k)}$ options for the sing sequence $\Big(\operatorname{sgn}\big(\operatorname{ratio}(q,x) - y\big) \mid (x,y) \in C\Big)$ over every $q \in Q(C)$.

Let $C_1, C_2 \subset P, C_1 \cap C_2 = \emptyset$ s.t. $|C_1|, |C_2| \geq 1$, where we know all the satisfiable sign sequences $\Big(\operatorname{sgn}\big(\operatorname{ratio}(q,x)) - y\big) \mid (x,y) \in C\Big)$ over $q \in Q(C)$ for $C \in \{C_1, C_2\}$. Since $|C_1|, |C_2| \leq n$, by Corollary 25, the size of $\Big\{\Big(\operatorname{sgn}\big(\operatorname{ratio}(q,x)) - y\big) \mid (x,y) \in C\Big) \mid q \in Q(C)\Big\}$ is in $(2kn)^{O(k)}$ for $C \in \{C_1, C_2\}$. Hence, the size of the candidate sets for

$$\Big\{\Big(\operatorname{sgn}\big(\operatorname{ratio}(q,x)) - y\big) \mid (x,y) \in C_1 \cup C_2\Big) \mid q \in Q(C_1 \cup C_2)\Big\}$$

is in $(2kn)^{O(k)}$. Therefore, by utilizing Corollary 26 to validate each candidate, we can compute in $(2kn)^{O(k)}$ the set $\Big\{\Big(\operatorname{sgn}\big(\operatorname{ratio}(q,x)) - y\big) \mid (x,y) \in C_1 \cup C_2\Big) \mid q \in Q(C_1 \cup C_2)\Big\}$.

Thus, partitioning P to sets of size $O(k) \geq 2k$, for each such $C \subset P$ using Corollary 26 to compute in $(2k)^{O(k)}$ the set $\Big\{\Big(\operatorname{sgn}\big(\operatorname{ratio}(q,x)) - y\big) \mid (x,y) \in C\Big) \mid q \in Q(C)\Big\}$, and combining all the options as stated above proves the lemma. □

Using this observation, and returning to the original problem, we obtain the following solver.

Lemma 28. *For every weighed set (S,w) that contains $n = |S|$ points, a pair $q \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell\big((S,w),q\big)$ can be computed in $(2kn)^{O(k)}$ time; see Definition 4.*

Proof. Let $\mathcal{Q} \subset \left(\mathbb{R}^k\right)^2$ be the union of all the pairs $(c,c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \operatorname{poly}(c',x) \neq 0$ for every $(x,y) \in S$. By Lemma 27, in $(2kn)^{O(k)}$ time, compute all the $n^{O(k)}$ possible sign sequences $\Big(\operatorname{sgn}\big(\operatorname{ratio}(q',x_1) - y_1\big), \cdots, \operatorname{sgn}\big(\operatorname{ratio}(q',x_n) - y_n\big)\Big)$ over every $q' \in \left(\mathbb{R}^k\right)^2$.

For every such sign sequences w', where $w'(p) \in \{-1,1\}$ is the sign of every point $p \in P$, observe the following:

For every $q := (c,c') \in \mathcal{Q}$ satisfying the assignment by w' we have

$$\ell\big((S,w),q\big) = \sum_{p=(x,y) \in S} w(p)w'(p) \cdot \left(\frac{\operatorname{poly}(c,x}{1 + x \cdot \operatorname{poly}(c',x)} - y\right).$$

Hence, by taking the common denominator in the right part of the equation above, there are polynomials $f, g : \left(\mathbb{R}^k\right)^2 \to \mathbb{R}$ of degree in $(2kn)^{O(1)}$ with $\left(\mathbb{R}^k\right)^2$ as the variables and $(2kn)^{O(1)}$

parameters, such that for every $q \in \mathcal{Q}$ we have

$$\ell\big((S,w),q\big) = \frac{f_i(q)}{g(q)}.$$

Hence, utilizing [54], this problem can be solved in $(2kn)^{O(k)}$ time; the solution is a numerical solution that can be approximated to arbitrary precision. Hence, by taking the minimum over the $(2kn)^{O(k)}$ candidate solutions, we can compute q as defined in the lemma in $(2kn)^{O(k)}$ time. □

4.3 Algorithm 2: Efficient $(1,\beta)$-approximation for large β

Lemma 29. *Let P be an n-signal, where n is a power of 2. Let $\beta \geq 1$ be an integer, and $B := (B_1, \cdots, B_\psi)$, where $\psi \in \big[\lfloor \beta/2 \rfloor, \beta\big]$, be the output of a call to* BATCH-APPROX(P,β)*; see Algorithm 2. Identify $\{(P_i, q_i)\} := B_i$, for every $i \in [\psi]$. Then, $B' := \{(P_1,q_1), \cdots, (P_\psi, q_\psi)\}$ is a $(1,\beta)$-approximation of P; see Definition 5. Moreover, the output of the call to* BATCH-APPROX(P,β) *can be compute in $n \cdot (2kn/\beta)^{O(k)}$ time.*

Proof. By its construction in Algorithm 2 we have that B' is an $(1,\beta)$-approximation of P. By Lemma 28, for every $i \in [\psi]$, the computation time of every q_i in Line 2 of Algorithm 2 is in $\big(2k|P_i|\big)^{O(k)}$. Combining this with the construction of Algorithm 2 proves the lemma. □

4.4 Algorithm 3; Coreset under constraints

In this section we will prove that Algorithm 3 constructs a restricted coreset for rational functions, which, as previously mentioned in Section 3.2, will be utilized to efficiently compute an (α,β)-approximation to a given n-signal P, where $\alpha, \beta \in O(\log(n))$. It should be emphasized that the final coreset construction has no such restrictions or assumptions for either its input or queries.

For readability we split the proof into three parts:

(i) Mostly citing previous work, we bound the polynomial-fitting sensitivity.

(ii) Utilizing the previous bound we compute a sensitivity for a restricted case of the RFF fitting problem from Equation (3.1) of Definition 4 that is formally sated and bounded in Lemma 35.

(iii) We utilize the previous bound in Lemma 38, which proves the previously stated Lemma 10 that summarises the desired properties of Algorithm 3.

4.4.1 Upper bound on the polynomial-fitting sensitivity

For the polynomial-fitting sensitivity, consider the following lemma that follows from the work on sensitivity of near-convex functions by Murad Tukan [65]

Lemma 30 (Lemma 35 in [65])**.** *Let Y be a set of n points in $\mathbb{R}^d$. A function $s' : Y \to [0, \infty)$ can be computed in $O(n \cdot d^2)$ time such that for every $x \in Y$ we have $\sup_{q \in \mathbb{R}^d} \frac{|x^T \cdot q|}{\sum_{y \in Y} |y^T \cdot q|} \leq s'(x)$, where the supremum is over $q \in \mathbb{R}^d$ such that $|x^T \cdot q| > 0$, and $\sum_{x \in Y} s'(x) \in O\left(d^{3/2}\right)$.*

Using this lemma we obtain the following corollary.

Corollary 31. *Let X be a set of $n \geq 1$ reals, and $k \geq 1$ be an integer. A function $s : X \to [0, \infty)$ can be computed in $O(n \cdot k^2)$ time such that for every $x \in X$ we have $\sup_{c \in \mathbb{R}^k} \frac{\left|\text{poly}(c, x)\right|}{\sum_{y \in X} \left|\text{poly}(c, y)\right|} \leq s(x)$, where the supremum is over $q \in \mathbb{R}^d$ such that $\left|\text{poly}(c, x)\right| > 0$, and $\sum_{x \in X} s(x) \in O\left(k^{3/2}\right)$.*

Proof. Let $f : X \to \mathbb{R}^k$ be the function that maps every $x \in X$ to $(1, x, \cdots, x^{k-1})^T$. Let $Y := \{f(x) \mid x \in X\}$ denote the image of f, and let $s' : Y \to [0, \infty)$ as defined in Lemma 30. For every $x \in X, c \in \mathbb{R}^k$, where $\left|\text{poly}(c, x)\right| > 0$, we have

$$\frac{\left|\text{poly}(c, x)\right|}{\sum_{y \in X} \left|\text{poly}(c, y)\right|} = \frac{\left|f(x)^T \cdot c\right|}{\sum_{y \in X} \left|f(y)^T \cdot c\right|} \leq s'\big(f(x)\big),$$

where the first inequality is by the definition of poly, and the second inequality is by the definition of s'; see Lemma 30. Let $s : X \to [0, \infty)$ be the function that maps every $y \in X$ to $s(y) := s'\big(f(y)\big)$. By Lemma 30, s satisfies all the claims in the corollary.

Computation time of s.

For every $x \in X$ we can compute $f(x)$ in $O(k^2)$ time. Hence, the computation time of Y is in $O(nk^2)$. By Lemma 30, the computation time of s' is in $O(n \cdot k^2)$. Therefore, since, for every $y \in X$ we defined $s(y) = s'\big(f(y)\big)$, we have that the computation time of s is in $O(n \cdot k^2)$. □

4.4.2 Upper bound on the RFF sensitivity

In this section we the RFF sensitivity for the restricted case mentioned in property (ii) at the beginning of Section 4.4. For this we we define the following.

Definition 32 (Lipschitz-function)**.** *Let $r > 0$, and $a, b \in \mathbb{R}$, where $a > b$. A function $f : [a, b] \to [0, \infty)$ is a r-Lipschitz if f is non-decreasing over $[a, b]$, and for every $c \geq 1$, and any $x \in [a, b]$ we have $f(c \cdot x) \leq c^r \cdot f(x)$.*

In the following claim we state a known property of polynomial functions.

Claim 33. *Let $k \geq 1$ be an integer, and $a, b \in \mathbb{R}$ where $a > b$. Let $f : [a, b] \to (0, \infty)$ be a positive and non-decreasing polynomial over $[a, b]$ of degree at most k. Then f is a k-Lipschitz function.*

Using the claim above, we obtain the following corollary.

Corollary 34. *Let $c' \in \mathbb{R}^k$. Let $G \subseteq \mathbb{R}$ denote the extrema of the function g that maps every $x \in \mathbb{R}$ to $f(x) = |1 + x \cdot \mathrm{poly}(c', x)|$. Let $X = [a, b] \subset \mathbb{R}, |X| > 1$ such that for every $x \in X$ we have*

$$\min_{\gamma \in G} |x - \gamma| \geq \max(X) - \min(X).$$

Then $\min_{x \in X} f(x) > 0$, and $\dfrac{\max_{x \in X} f(x)}{\min_{x \in X} f(x)} \leq 2^k$

Proof. By the definition of X, if there is $x \in X$ such that $x \in G$, then by substituting $0 = \min_{\gamma \in G} |x - \gamma| \geq \max(X) - \min(X)$ we have that $\max(X) = \min(X)$ and as such $|X| = 1$, which contradicts the definition of X in the claim. Therefore, we have $X \cap G = \emptyset$, that is g has no extrema over X. If there is $x \in X$ such that $f(x) = 0$, by the definition of f we have that x is an extrema of the function f. Hence, there is no $x \in X$ such that $f(x) = 0$, which yields $\min_{x \in X} f(x) > 0$.

Since f has no extrema over X we can prove that $\frac{\max_{x\in X} f(x)}{\min_{x\in X} f(x)} \leq 2^k$ via the following

Case (i): f is constant in the range of X.

By the definition of the case it trivially holds that $\max_{x\in X} f(x) = \min_{x\in X} f(x)$.

Case (ii): f increases in the range of X.

Let $x, y \in X$ such that $x < y$. By the assumptions and Claim 33 it holds that $f(x) \leq 2^k \cdot f(y)$. Hence, $\max_{x\in X} f(x) \leq 2^k \min_{x\in X} f(x)$.

Case (iii): g decreases in the range of X.

Let $x, y \in X$ such that $x < y$. By the assumptions and Claim 33 it holds that $f(y) \leq 2^k \cdot f(x)$. Hence, $\max_{x\in X} f(x) \leq 2^k \min_{x\in X} f(X)$. □

Using Corollary 34 we obtain the following bound for the RFF sensitivity.

Lemma 35. *Let $q_1 = (c_1, c_1') \in \left(\mathbb{R}^k\right)^2$. Let $D \subseteq \mathbb{R}$ denote the extrema of the function g that maps every $x \in \mathbb{R}$ to $g(x) = |1 + x \cdot \mathrm{poly}(c_1', x)|$. Let $X \subset \mathbb{R}$ such that for every $x \in [\min(X), \max(X)]$ we have that $\min_{\gamma\in D} |x - \gamma| \geq \max(X) - \min(X)$. Let $q_2 \in \left(\mathbb{R}^k\right)^2$ be 2^k-bounded over X; see Definition 7. Let $s : X \to [0, \infty)$ be the sensitivity bound computed in Corollary 31 after substituting k with $2k + 1$ and Y by X. For every $x \in X$ we have*

$$\frac{\left|\mathrm{ratio}(q_1, x) - \mathrm{ratio}(q_2, x)\right|}{\sum_{y\in X} \left|\mathrm{ratio}(q_1, y) - \mathrm{ratio}(q_2, y)\right|} \leq 4^k \cdot s(x).$$

Proof. If $|X| = 1$, by the construction of s in Corollary 31 we have that the single value $x \in X$ satisfies $s(x) \geq 1$, which yields that the inequality in the lemma holds for this case. Therefore, from now on we assume that this is not the case, i.e., we assume that $|X| > 1$.

Identify $q_2 = (c_2, c_2')$, and let $c \in \mathbb{R}^{2k+1}$ such that for every $x \in \mathbb{R}$ we have

$$\mathrm{poly}(c, x) = \mathrm{poly}(c_1, x) \cdot \big(1 + x \cdot \mathrm{poly}(c_2', x)\big) - \mathrm{poly}(c_2, x) \cdot \big(1 + x \cdot \mathrm{poly}(c_1', x)\big).$$

Let $g_1, g_2 : \mathbb{R} \to [0, \infty)$ denote the functions that map every $y \in X$ to $g_1(y) = \frac{1}{1 + y \cdot \mathrm{poly}(c_1', y)}$

and $g_2(y) = \dfrac{1}{1 + y \cdot \mathrm{poly}(c_2', x)}$, respectively. Let $x \in X$. We have

$$\begin{aligned}\big|\mathrm{ratio}(q_1, x) - \mathrm{ratio}(q_2, x)\big| &= \left|\frac{\mathrm{poly}(c_1, x)}{1 + x \cdot \mathrm{poly}(c_1', x)} - \frac{\mathrm{poly}(c_2, x)}{1 + x \cdot \mathrm{poly}(c_2', x)}\right| = \\ &\left|(\frac{\mathrm{poly}(c, x)}{\big(1 + x \cdot \mathrm{poly}(c_1', x)\big) \cdot \big(1 + x \cdot \mathrm{poly}(c_2', x)\big)}\right| = \big|\mathrm{poly}(c, x) \cdot g_1(x) \cdot g_2(x)\big|,\end{aligned} \tag{4.9}$$

where the second equality is by assigning the definition of c, and the third equality is by assigning the definition of g_1 and g_2.

Substituting X by $[\min(X), \max(X)]$ and c' by c_1' in Corollary 34. yields that the function $f : X \to \mathbb{R}$ that maps every $x \in X$ to $f(x) = |1 + x \cdot \mathrm{poly}(c', x)|$ satisfies that $\min_{x \in X} f(x) > 0$ and $\dfrac{\max_{x \in X} f(x)}{\min_{x \in X} f(x)} \leq 2^k$. That is, since $q_1 = (c_1, c_1')$, that q_1 is 2^k well behaved over X; see Definition 7.

Let $f_1, f_2 : \mathbb{R} \to [0, \infty)$ denote the functions that map every $y \in X$ to $f_1(y) = |1 + y \cdot \mathrm{poly}(c_1', y)|$ and $f_2(y) = |1 + y \cdot \mathrm{poly}(c_2', y)|$, respectively. We have,

$$\frac{\big|\mathrm{ratio}(q_1, x) - \mathrm{ratio}(q_2, x)\big|}{\sum_{y \in X} \big|\mathrm{ratio}(q_1, y) - \mathrm{ratio}(q_2, y)\big|} = \frac{\big|\mathrm{poly}(c, x) \cdot f_1(x) \cdot f_2(x)\big|}{\sum_{y \in X} \big|\mathrm{poly}(c, y) \cdot f_1(y) \cdot f_2(y)\big|} \tag{4.10}$$

$$\leq \frac{\max_{y \in X} |g_1(y)|}{\min_{y \in X} |g_1(y)|} \cdot \frac{\max_{y \in X} |g_2(y)|}{\min_{y \in X} |g_2(y)|} \cdot \frac{|\mathrm{poly}(c, x)|}{\sum_{y \in X} |\mathrm{poly}(c, y)|} \tag{4.11}$$

$$= \frac{\max_{y \in X} |f_1(y)|}{\min_{y \in X} |f_1(y)|} \cdot \frac{\max_{y \in X} |f_2(y)|}{\min_{y \in X} |f_2(y)|} \cdot \frac{|\mathrm{poly}(c, x)|}{\sum_{y \in X} |\mathrm{poly}(c, y)|} \tag{4.12}$$

$$\leq 4^k \cdot \frac{|\mathrm{poly}(c, x)|}{\sum_{y \in X} |\mathrm{poly}(c, y)|} \tag{4.13}$$

$$\leq 4^k \cdot s(x), \tag{4.14}$$

where (4.10) is by (4.9), (4.11) follows from assigning that $x \in X$, (4.12) follows from reorganizing the expression and assigning the definitions of f_1, f_2, g_1, g_2, (4.13) holds since q_1 and q_2 are

2^k-bounded over X, and (4.14) is by assigning the definition of s in the lemma. □

4.4.3 Algorithm 3: proof of Lemma 10

In the following lemma we show a minor result that was used in Algorithm 3.

Lemma 36. *Let $c \in \mathbb{R}^k$ and $X = \{a, a+1 \cdots, b\} \subset [n]$ be a non empty interval of $[n]$. Let $f : \mathbb{R} \to [0, \infty)$ be the function that maps every $x \in \mathbb{R}$ to $f(x) = |1 + x \cdot \text{poly}(c, x)|$. There is a partition $\{X_1, \cdots, X_\eta\}$ of X into $|\eta| \leq 2k - 1$ sets, such that for every $i \in [\eta]$ the function f is monotonic over $\big[\min(X_i), \max(X_i)\big)$, and for every $i, j \in [\eta]$, where $i \neq j$ we have $X_i \cap \big[\min(X_j), \max(X_j)\big] = \emptyset$. Moreover, this partition can be computed in $(k+1)^{O(1)} \cdot |X|$ time.*

Proof. Let $g : \mathbb{R} \to [0, \infty)$ be the function that maps every $x \in \mathbb{R}$ to $g(x) = 1 + x \cdot \text{poly}(c, x)$, which is an polynomial of degree at most k. Observe that, by the fundamental theorem of algebra, any non zero polynomial (due to its construction g is non zero) of degree at most k has at most k roots. Thus, since the derivative of g is a polynomial of degree at most $k-1$, g has at most $k-1$ extrema. Hence, as any extrema of f is either a root or extrema of g, f has at most $2k-1$ extrema. Partitioning X according to the $2k-1$ extrema of f yields the partition $\{X_1, \cdots, X_\eta\}$ from the lemma.

Running time: Observe that in the root finding presented above (including in the computation of the extrema of g) it suffices to only search for roots in the range of X, and for the non integer roots only for which integer a they are in $(a, a+1)$. Each integer candidate for the roots can be validated by a simple assignment in the polynomial in $(k+1)^{O(1)}$ time. By Sturm's-theorem [63], we can validate each interval candidate for a root in $(k+1)^{O(1)}$ time. Since there are $|X|$ candidates for roots (intervals and integer), all of the roots can be computed for the sufficient precision in $(k+1)^{O(1)} \cdot |X|$ time. Since f has at most $2k-1$ extrema, the partition of X can be computed in $O(k \cdot |X|)$ time. Hence, the overall running time is in $(k+1)^{O(1)} \cdot |X|$. □

Lemma 16 bounds the VC-dimension that corresponds to the function $D(q, p_i)$ over the points $p_1, \cdots, p_n$ in a n-signal projected onto its bicriteria; see Definition 4 and Definition 5. We now give

a similar bound for the distance function $D(q,(x,y)) = |\mathrm{ratio}(q,x) - y|$ between every rational function q to the point $p_i = (x,y)$, where the points are a general set of points in the plane.

The following lemma, similarly to Lemma 16, is inspired by Theorem 12 in [36].

Lemma 37. *Let $P = \{(1,y_1),\cdots,(n,y_n)\}$ be an n-signal. For every $p_i = (i,y) \in P$ and any $q \in \left(\mathbb{R}^k\right)^2$ let $g_i(q) = D(q,p_i) = |\mathrm{ratio}(q,i) - y|$; see Definition 4. Let $G = \{g_1,\ldots,g_n\}$. The dimension of the range space $\mathcal{R}_{\left(\mathbb{R}^k\right)^2,G}$ that is induced by $\left(\mathbb{R}^k\right)^2$ and G is in $O(k^2)$.*

Proof. For every $(q,r) = \big((c,c'),r\big) \in \left(\mathbb{R}^k\right)^2 \times \mathbb{R}$, let $h_{(c|c'|r)} : \mathbb{R} \to \{0,1\}$ that maps every $i \in [n]$ to $h_{(c|c'|r)}(i) = 1$ if and only if $f_i(q) \leq r$, and every $x \in \mathbb{R} \setminus [n]$ to $h_{(c|c'|r)}(x) = 0$. Let $\mathcal{H} = \{h_\theta \mid \theta \in \mathbb{R}^{2k+1}\}$. For every $c \in \mathbb{R}^k$ and any $x \in \mathbb{R}$ we can compute $\mathrm{poly}(c,x)$ with $O(k)$ arithmetic operations on real numbers and jumps conditioned on comparisons of real numbers; see, for example, Horner's scheme [42], which is used in numpy's implementation of the method polyval [27]. Therefore, for every $x \in \mathbb{R}$ and any $\theta \in \mathbb{R}^{2k+1}$, by the definition of D, we can calculate $h_\theta(x)$ with $O(k)$ arithmetic operations on real numbers and jumps conditioned on comparisons of real numbers. Hence, substituting $d := n$, $m := 2k+1$, $h := h$, $\mathcal{H} := \mathcal{H}$ and $t \in O(k)$ in Theorem 14 yields that the VC-dimension of $\mathcal{H}$ is in $O(k^2)$. Hence, by the construction of $\mathcal{H}$ and the definition of range spaces in Definition 15, we have that the dimension of the range space $\mathcal{R}_{P,F}$ that is induced by P and F is in $O(k^2)$. □

This combined with the previous results yields the following restricted coreset construction that utilizes the previous reduction of the RFF sensitivity to polynomial sensitivity.

Lemma 38. *Let P be an interval of an n-signal which is projected onto some $q \in \left(\mathbb{R}^k\right)^2$, i.e., $B := (P,q)$ is a $(0,1)$-approximation B of P; see Definition 5. Let X be the first coordinate of P, i.e., $X := \{x \mid (x,y) \in P\}$. Put $\varepsilon,\delta \in (0,1/10]$, and let*

$$\lambda \geq \frac{c^*}{\varepsilon^2} \cdot (4^k k^2 + 1)\left(k^2 \log(4^k k^2+1) + \log\left(\frac{k \log n}{\delta}\right)\right),$$

be an integer, where $c^* \geq 1$ is a constant that can that can be determined from the proof. Let (S,w) be the weighted set that is returned by a call to LIMITED-CORESET(B,λ)*; see Algorithm 3. Then*

$|S| \in O\left(k\lambda \cdot \log n\right)$ *and, with probability at least* $1-\delta$*, for every* $q' \in \left(\mathbb{R}^k\right)^2$ *that is* 2^k*-bounded over* X *(see Definition 7), we have*

$$|\ell(P, q') - \ell\left((S, w), q'\right)| \leq \varepsilon \cdot \ell(P, q'). \tag{4.15}$$

Proof. By the construction of Algorithm 3 we have that $|S| \in O\left(k\lambda \cdot \log n\right)$, which follows from the bounds on the order of η and every m_i stated in Algorithm 3. Let $\delta' := \lceil c_2^*(k \log n)/\delta \rceil$ for a constant $c_2^* > 0$ that can be determined from the proof, more specifically see Equation (4.22). Consider the set $X_i^j \subset X$, for $i \in [\eta]$ and $j \in [m_i]$, that was constructed during the execution of the i-th iteration of the outer "for" loop and j-th iteration of the inner "for" loop in the call to LIMITED-CORESET(B, λ). Let $(c, c') := q$, and let $D \subseteq \mathbb{R}$ denote the extrema of the function f that maps every $x \in \mathbb{R}$ to $f(x) = |1 + x \cdot \text{poly}(c', x)|$. By the construction of Algorithm 3, the size or diameter of X_i^j is smaller than its distance from the edges of X_i, i.e., $|X_j^i| \leq \min\left(|x - \max(X_i^j), ||x - \min(x_i^j)|\right)$. Since X_i has no extreme points, this implies that for every $x \in X_i^j$ we have $\min_{\gamma \in D} |x - \gamma| \geq |X_i^j| = \max(X_i^j) - \min(X_i^j)$. Hence, assigning $X := X_i^j, q_1 := q, q_2 : -q$, and the function $s : X_i^j \to [0, \infty)$ computed in the call to LIMITED-CORESET(B, λ) for X_i^j in Lemma 35 yields for every $x \in X_i^j$ that

$$\frac{\left|\text{ratio}(q, x) - \text{ratio}(q', x)\right|}{\sum_{y \in X_i^j} \left|\text{ratio}(q, y) - \text{ratio}(q', y)\right|} \leq 4^k \cdot s(x). \tag{4.16}$$

Let P_i^j and S_i^j be as defined in Line 10 and Line 11, respectively, during the execution of the call to LIMITED-CORESET(B, λ), and let $w_i^j : S_i^j \to [0, \infty)$ such that for every $p \in S_i^j$ we have $w_i^j(p) = w(p)$. Let $\mathcal{Q}$ denote the union over every $q' \in \left(\mathbb{R}^k\right)^2$ that is 2^k-bounded over X; see Definition 7.

We now prove that S_i^j is an ε-subset corset for the query space $\left(P_i^j, \mathcal{Q}, D, \|\cdot\|_1\right)$. Indeed, the corresponding dimension of $\mathcal{R}_{\mathcal{Q},G}$, where is G is as defined in Lemma 37, is in $O(k^2)$. The sensitivity of every $(x, y) := p \in P_i^j$ is $s(x) = \max_{q \in \mathcal{Q}} D(q, p)/\ell(P_i^j, q) \leq 4^k s(x)$ where the inequality is by Equation (4.16). The total sensitivity is thus $t = \sum_{x \in X_i^j} 4^k s(x) \in 4^k \cdot O(k^2)$, where the is by the inequality is by definition of s computed in the call to LIMITED-CORESET(B, λ).

Thus, substituting $\delta := \delta'$, the query space,$\left(P_i^j, \mathcal{Q}, D, \|\cdot\|_1\right)$ in Theorem 20 combined with the construction of Algorithm 3 yields that, with probability at least $1-\delta'$, we have (S_i^j, w_i^j) is an ε-subset-coreset for the query space $\left(P_i^j, \mathcal{Q}, D, \|\cdot\|_1\right)$. That is, with probability at least $1-\delta'$, for every $q' \in \left(\mathbb{R}^k\right)^2$ we have

$$\left|\ell(P_i^j, q') - \ell\Big((S_i^j, w_i^j), q'\Big)\right| \leq \varepsilon \cdot \ell(P_i^j, q'). \tag{4.17}$$

Taking the union over every $i \in [\eta]$ and $j \in [m_i]$, under the assumption that Equation (4.17) holds for each pair, yields

$$\begin{aligned}
\left|\ell(P, q') - \ell\big((S, w), q'\big)\right| &= \left|\sum_{i=1}^{\eta}\sum_{j=1}^{m_i}\left(\ell\left(P_i^{(j)}, q'\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q'\right)\right)\right| && (4.18)\\
&\leq \sum_{i=1}^{\eta}\sum_{j=1}^{m_i}\left|\ell\left(P_i^{(j)}, q'\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q'\right)\right| && (4.19)\\
&\leq \sum_{i=1}^{\eta}\sum_{j=1}^{m_i}\varepsilon \cdot \ell\left(P_i^{(j)}, q'\right) && (4.20)\\
&= \varepsilon \cdot \ell(P, q'), && (4.21)
\end{aligned}$$

where Equation 4.18 is by the construction of the weighed set (S, w) and the partition $\left\{P_i^{(j)}\right\}$ of P in Algorithm 3, Equation 4.19 is by the triangle inequality, Equation 4.20 is by assigning Equation (4.17) (which was assumed to hold for all the values), and Equation 4.21 follows since $\left\{P_i^{(j)}\right\}$ computed in Algorithm 3 is a partition of P.

Since $i \in [\eta]$ by Line 5 and $j \in [m_i]$ by Line 7 of Algorithm 3, we have at most $O(k \log n)$ sets $P_{i,j}$; follows by assigning the bounds on the order of η, and every m_i from Algorithm 3. By the union bound, Equation (4.17) hold simultaneously for every $i \in [\eta]$ and $j \in [m_i]$ with probability at least

$$1 - \delta' \cdot \sum_{i=1}^{\eta} m_i \geq 1 - \delta, \tag{4.22}$$

which holds for $\delta' := \lceil c_2^*(k \log n)/\delta \rceil$ for some constant $c_2^* > 0$. Hence, with probability at least $1-\delta$ we have $|\ell(P, q') - \ell\left((S, w), q'\right)| \leq \varepsilon \cdot \ell(P, q')$ as stated in Equation (4.15). □

4.5 Algorithm 4; Combining (α, β)-approximations

The input for the algorithm is P an interval of n-signal which is projected onto some set of (α, β)-approximations; see Definition 5. This projection is represented by the set B, where each element $B_i \in B$ is a $(0, \beta)$-approximation for some P_i, and $\{P_1, \cdots, P_{|B|}\}$ is a consecutive partition of P. The algorithm returns B', a bicriteria-approximation of P as in Definition 5, where the size of B' is smaller than $\sum_{i=1}^{|B|} |B_i|$. The algorithm runs in $O(|P|^{1+\varepsilon})$ time, for every constant $\varepsilon > 0$.

The desired properties of Algorithm 4 are stated and proved in Lemma 40.

For the sake of analysis, we will use the following corollary.

Corollary 39. *Let $\{R_1, \cdots, R_\beta\}$ be a set of $\beta \geq 6k$ equally sized distinct partitions of $[n]$ such that for every $i, j \in [\beta], i \neq j$ we have $R_i \cap \big[\min(R_j), \max(R_j)\big] = \emptyset$. For every $q \in \left(\mathbb{R}^k\right)^2$, there is $C \subset [\beta], |C| = \beta - 6k + 3$ such that q is 2^k-bounded over every R_i, for every $i \in C$.*

Proof. Let $q = (c, c')$, and let G be the set of the extrema of the function $f : \mathbb{R} \to \mathbb{R}$ that maps every $x \in \mathbb{R}$ to $f(x) = |1 + x \cdot \text{poly}(c', x)|$. Let $r = |R_1| = \cdots = |R_\beta|$. W.l.o.g. assume that for every $i \in [\beta - 1]$ we have $\min(R_{i+1}) > \max(R_i)$; i.e., the sets are ordered in an increasing order. Let $R_0 = \{\min(R_1) - r, \cdots, \min(R_1) - 1\}$ and $R_{\beta+1} = \{\max(R_\beta) + 1, \cdots, \max(R_\beta) + 1 + r\}$. By the proof of Lemma 36, f has at most $2k - 1$ extrema. Therefore, removing from $[\beta]$ every index $i \in [\beta]$ such that f has an extrema in the range of either from $\{R_{i-1}, R_i, R_{i+1}\}$ yields a set $C \subset [\beta], |C| = \beta - 6k + 3$ such that for every $i \in C$ we have $\forall x \in R_i : \min_{g \in G} |x - g| \geq r = \max(R_i) - \min(R_i)$. For every $i \in C$, substituting c', G and $X := R_i$ in Corollary 34 yields that the function $g : X \to \mathbb{R}$ that maps every $x \in R_i$ to $g(x) = \dfrac{1}{1 + x \cdot \text{poly}(c', x)}$ is well defined, and satisfies $\dfrac{\max_{x \in X} |g(x)|}{\min_{x \in X} |g(x)|} \leq 2^k$. Thus, q is 2^k-bounded over R_i for every $i \in C$; see Definition 7. Hence, C satisfies the corollary. □

Combining a removal inspired by [55], with the coreset construction from the previous section, yields the following lemma.

Lemma 40. *Let $B := \{B_1, \cdots, B_\beta\}$, where each $B_i \in B$ is an $(0, r_i)$-approximation of P_i, i.e. P_i is projected onto B_i, and $\{P_1, \cdots, P_\beta\}$ is an equally-sized consecutive partition of some interval*

of an n-signal P; see Fig. 3-2 and Definition 5. Put $\varepsilon, \delta \in (0, 1/10]$, and let

$$\lambda \geq \frac{c^*}{\varepsilon^2}(4^k k^2 + 1)\left(k^2 \log_2(4^k k^2 + 1) + \log_2\left(\frac{kn}{\delta}\right)\right)$$

be an integer, where $c^* \geq 1$ is a constant that can be determined from the proof. Let B' be the output of REDUCE$(B, \lambda, 6k-3)$*; see Algorithm 4. With probability at least $1-\delta$, we have that B' is a $(1+10\varepsilon, \beta^*)$-approximation of P for $\beta^* \in O\left(k \cdot \max_{i \in [\beta]} r_i\right)$; see Definition 5.*

Moreover, for $\varepsilon = 1/10$ we have that the running time of the call to REDUCE$(B, \lambda, 6k-3)$ *is in $|P| \cdot \beta^{6k-3} \cdot 4^{O(k^2)} \left(\log(n/\delta)\beta'\right)^{O(k)}$, where $\beta' = \sum_{i=1}^{\beta} r_i$.*

Proof. If $\beta \leq 12k-6$, by the construction of B' in Algorithm 4 in Lines 1 and 20, we have $B' := \bigcup_{i=1}^{\beta} B_i$. Since B_i is a $(0, r_i)$-approximation of P_i for every $i \in [\beta]$ and $\{P_1, \cdots, P_\beta\}$ is a partition of P, B' is an $(0, |B'|)$-approximation of P which yields that the lemma trivially holds. Thus, from now on, we assume that this is not the case.

For every $B_i \in B$ identify $\left\{B_i^{(1)}, \cdots, B_i^{(r_i)}\right\} := B_i$. Let $q \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell(P, q)$. For every $i \in [\beta]$, let $R_i := \{x \mid (x, y) \in P_i\}$, i.e. the first coordinates of the points in P_i. Since $\{P_1, \cdots, P_\beta\}$ is an equally-sized consecutive partition of some interval of an n-signal P, we have that $\{R_1, \cdots, R_\beta\}$ is a set of equally-sized partitions of $[n]$ such that for every $i, j \in [\beta], i \neq j$ we have $R_i \cap \left[\min(R_j), \max(R_j)\right] = \emptyset$. By Corollary 39, there is $G \subset [\beta], |G| = \beta - 6k + 3$ such that q is 2^k-bounded over R_i for every $i \in G$. Consider the integration of the "for" loop in the call to Algorithm 4 where G is computed, i.e., $G := G$.

Let G', S_G, w_G, and q_G be defined as in the "for" loop iteration in the call to Algorithm 4. Let P_G and $P_{G'}$ be the union of P_i over every $i \in G$ and $i \in G'$, respectively. Let $P_{G \setminus G'}$ denote $P_{G'} \cup \left(P \setminus P_G\right)$. The set $\left\{B_i^{(j)} \mid i \in G' \cup \left([\beta] \setminus G\right), B_i^{(j)} \in B_i\right\}$ is a $\left(0, \sum_{i=1}^{\beta} |B_i|\right)$-approximation for $P_{G \setminus G'}$. Hence, by the construction of B' in Line 20 of Algorithm 4 it is left to prove that

$$\ell(P_{G \setminus G'}, q_G) \leq (1+10\varepsilon) \cdot \ell(P_G, q) \leq (1+10\varepsilon) \cdot \ell(P, q). \tag{4.23}$$

The last inequality trivially holds since $P_G \subseteq P$. It is left to prove the first inequality of Eq. (4.23).

By Corollary 39, where we substitute β by $|C|$, there is a set $G^* \subset G$ of size $|G^*| = |G| - 6k + 3$ such that q_G is 2^k-bounded over R_i for every $i \in G^*$. For every $i \in G$ and any $B_i^{(j)} \in B_i$ identify $\left(P_i^{(j)}, q_i^{(j)}\right) := B_i^{(j)}$. For every $i \in G$ and any $B_i^{(j)} \in B_i$ let $\left(S_i^{(j)}, w_i^{(j)}\right) :=$ LIMITED-CORESET $\left(\left\{B_i^{(j)}\right\}, \lambda\right)$ as computed in Line 5 of the call to Algorithm 4; for $B_i^{(j)} := B_i^{(j)}$. Substituting $P = P_i^{(j)}$, $\delta = \delta/(2n), q' = q_G$, and $X := \left\{x \mid (x,y) \in P_i^{(j)}\right\}$ (the first coordinate of $P_i^{(j)}$) in Lemma 38 for every $i \in G^*$ and $j \in [|B_i|]$ (by the choice of G^*, q_G is 2^k-bounded over R_i, which contains the first coordinate of $P_i^{(j)}$) yields that with probability at least $1 - \frac{\delta}{2n}$ we have

$$\left|\ell\left(P_i^{(j)}, q_G\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q_G\right)\right| \leq \varepsilon \cdot \ell\left(P_i^{(j)}, q_G\right). \tag{4.24}$$

Let $S^* := \bigcup_{i \in G \setminus G^*, B_i^{(j)} \in B_i} S_i^{(j)}$, and $w^* : S^* \to [0, \infty)$ be the function that maps every $p \in S^*$ to $w^*(p) = w_G(p)$. Applying Equation (4.24) for every $i \in G^*$ and $j \in [|B_i|]$, along with utilizing the union bound, yields that, with probability at least $\left(1 - \delta/(2n)\right)^n \geq 1 - \delta/2$, we have

$$\begin{aligned}
\left|\ell(P_{G \setminus G^*}, q_G) - \ell\left((S^*, w^*), q_G\right)\right| &= \left|\sum_{i \in G^*} \sum_{j=1}^{|B_i|} \left(\ell\left(P_i^{(j)}, q_G\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q_G\right)\right)\right| && (4.25)\\
&\leq \sum_{i \in G^*} \sum_{j=1}^{|B_i|} \left|\ell\left(P_i^{(j)}, q_G\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q_G\right)\right| && (4.26)\\
&\leq \varepsilon \cdot \ell(P_{G \setminus G^*}, q_G), && (4.27)
\end{aligned}$$

where Equation (4.25) is by the constructions of P_{G^*} and S^*, Equation (4.26) is by the triangle inequality, and Equation (4.27) is by Equation (4.24).

Substituting $P = P_i^{(j)}$, $\delta = \delta/(2n), q' = q$, and $X := \left\{x \mid (x,y) \in P_i^{(j)}\right\}$ (the first coordinate of $P_i^{(j)}$) in Lemma 38 for every $i \in G$ and $B_i^{(j)} \in B_i$ (by the choice of G, q is 2^k-bounded over R_i, which contains the first coordinate of $P_i^{(j)}$) yields that with probability at least $1 - \frac{\delta}{2n}$ we have

$$\left|\ell\left(P_i^{(j)}, q\right) - \ell\left(\left(S_i^j, w_i^j\right), q\right)\right| \leq \varepsilon \cdot \ell\left(P_i^{(j)}, q\right). \tag{4.28}$$

Applying Equation (4.28) for every $i \in G$ and $B_i^{(j)} \in B_i$, along with utilizing the union bound, yields that, with probability at least $\big(1 - \delta/(2n)\big)^n \geq 1 - \delta/2$, we have

$$\big|\ell\big(P_G, q\big) - \ell\big((S_G, w), q\big)\big| = \left|\sum_{i \in G} \sum_{j=1}^{|B_i|} \left(\ell\left(P_i^{(j)}, q\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q\right)\right)\right| \tag{4.29}$$

$$\leq \sum_{i \in G} \sum_{j=1}^{|B_i|} \left|\ell\left(P_i^{(j)}, q\right) - \ell\left(\left(S_i^{(j)}, w_i^{(j)}\right), q\right)\right| \tag{4.30}$$

$$\leq \varepsilon \cdot \ell\big(P_G, q\big), \tag{4.31}$$

where Equation (4.29) is by the constructions of the P_G and S_G in Algorithm 4, Equation (4.30) is by the triangle inequality, and Equation (4.31) is by Equation (4.24).

If both Equations (4.25) to (4.27) and Equations (4.29) to (4.31) holds, which happens with probability at least $1 - \delta$, we have

$$\ell(P_{G \setminus G'}, q_G) \leq \ell(P_{G \setminus G^*}, q_G) \tag{4.32}$$

$$\leq \frac{1}{1-\varepsilon} \cdot \ell\big((S^*, w^*), q_G\big) \tag{4.33}$$

$$\leq \frac{1}{1-\varepsilon} \cdot \ell\big((S, w), q_G\big) \tag{4.34}$$

$$\leq \frac{1}{1-\varepsilon} \cdot \ell\big((S, w), q^*\big) \tag{4.35}$$

$$\leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \ell(P_G, q^*) \tag{4.36}$$

$$\leq (1 + 10\varepsilon) \cdot \ell(P_G, q^*), \tag{4.37}$$

where Equation (4.32) is by the choice of G' in Line 16 of Algorithm 4, Equation (4.33) is by Equations (4.25) to (4.27), Equation (4.34) is since $S^* \subset S$, Equation (4.35) is since $q \in \arg\min_{q' \in \left(\mathbb{R}^k\right)^2} \ell\big((S, w), q'\big)$ (by the construction of Algorithm 4), Equation (4.36) is by Equations (4.29) to (4.31), and Equation (4.37) holds since $\varepsilon \leq 1/2$.

Running time:

Let $\beta' := \sum_{i=1}^{\beta} r_i$ and assign $\varepsilon = \frac{1}{10}$.

If $\beta < 6k-3$, then, by the construction of Algorithm 4, the output can be computed in $O(|P| \cdot \beta')$. Hence, from now on we assume this is not the case.

Consider a single "for" iteration over the values of $G \subset [\beta]$ during the execution of Line 6 in the call to Algorithm 4. By the construction of every $\left(S_i^{(j)}, w_i^{(j)}\right)$ in Line 5 of Algorithm 4, we have that $|S_i^{(j)}| \leq \lambda$, for every $i \in [\beta]$ and $j \in [|B_i|]$. Hence, since $|S_G|$ is the union of $S_i^{(j)}$, over every $i \in G \subset [\beta]$ and $j \in [|B_i|]$, recalling the definition of β' trivially yields $|S_G| \in O(\lambda\beta')$. By Lemma 28, Line 11 can be computed in $(2k|S_G|)^{O(k)} \in \left(\lambda \cdot \beta'\right)^{O(k)}$ time. The rest of the lines can be computed in $|P| \cdot (k+1)^{O(1)}$ time, therefore, every iteration can be computed $|P| \cdot (k+1)^{O(1)} + \left(\lambda \cdot \beta'\right)^{O(1)}$ time. Assigning $\varepsilon = \frac{1}{10}$ in the definition of λ in the lemma yields $\lambda \in 4^{O(k)} \log(n/\delta)$. Since there are $\binom{\beta}{6k-3} \in O\left(\beta^{6k-3}\right)$ iterations of the "for" loop we obtain a total running time of $|P| \cdot \beta^{6k-3} \cdot 4^{O(k^2)} \left(\log(n/\delta)\beta'\right)^{O(k)}$.

□

4.6 Algorithm 5; Coreset computation

In this section we prove that Algorithm 5, which gets P an n-signal of length $n \geq 2k$ is a power of 2, and $\varepsilon, \delta \in (0, 1/10]$, returns an ε-approximation of P with failure probability at most δ. The formal statement and its proof is given in Theorem 42.

For the analysis we will use the following corollary, which is inspired by Lemma 3.6 in [9] and proves that projecting a dataset on a corresponding approximation for some function yields a coreset for this function.

Corollary 41. *Let P be an n-signal. Let $B := \{(P_1, q_1), \cdots, (P_\beta, q_\beta)\}$ be an (α, β)-approximation of P; see Definition 5. For every $i \in [\beta]$ let $P_i^* := \left\{\left(x, \text{ratio}(q_i, x)\right) \mid (x, y) \in P_i\right\}$, and let $P^* := \bigcup_{i=1}^{\beta} P_i^*$; i.e., P^* is the projection of P onto B. Let $q = (c, c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \text{poly}(c', x) \neq 0$ for every $(x, \cdot) \in P$. Then $\ell(P^*, q) \leq (1+\alpha) \cdot \ell(P, q)$.*

Proof. We have

$$\ell(P^*, q) = \sum_{i=1}^{\beta} \sum_{(x,y)\in P_i} \left|\text{ratio}(q, x) - \text{ratio}(q_i, x)\right| \tag{4.38}$$

$$\leq \sum_{i=1}^{\beta} \sum_{(x,y)\in P_i} \left(\left|\text{ratio}(q_i, x) - y\right| + \left|\text{ratio}(q, x) - y\right|\right) \tag{4.39}$$

$$= \sum_{i=1}^{\beta} \sum_{p\in P_i} D(q_i, p) + \sum_{p\in P} D(q, p) \tag{4.40}$$

$$\leq (1 + \alpha) \cdot \ell(P, q), \tag{4.41}$$

where Equation (4.38) is by assigning the definition of D and the definitions from the corollary, Equation (4.39) is by the triangle inequality, Equation (4.40) by the definition of D, and Equation (4.41) is by the definitions from the corollary. □

In [9] there was presented the Merge-Reduce scheme, that can be informally summarised as constructing an on-line coreset by combining pairs of coresets in a balanced tree fashion.

At first glance, it may seem that there would be a significant problem using the previously discussed variant of the Merge-Reduce scheme presented in [9]. Indeed, using the classic version where combining each pair of coresets would yield that the guarantee on the final approximation would be that the cost of the up to a polynomial in the data-set's size factor from the optimal solution (hence, the sample in SAMPLE-CORESET would be larger than the original dataset).

This can be fixed by combining a significantly larger number of nodes, which would yield that a tree of height $\lceil \log \log n \rceil$, and as a consequence that the final approximation would be poly-logarithmic in the dataset size.

The following theorem and Algorithm 5 are the main result of this thesis.

Theorem 42. *Let P be an n-signal, for $n \geq 2k$ that is a power of 2, and put $\varepsilon, \delta \in (0, 1/10]$. Let (B, C, w) be the output of a call to* CORESET$(P, k, \varepsilon, \delta)$*; see Algorithm 5. With probability at least $1 - \delta$, (B, C, w) is an ε-coreset of P; see Definition 6. Moreover, (B, C, w) can be computed in $2^{O(k^2)} \cdot n \cdot n^{O(k)/\log\log(n)} \cdot \log(n)^{O(k\log(k))} \cdot \log(1/\delta)^{O(k)}$ time and $k^{O(1)} \cdot \log(n)^{O(1)+\log k} \cdot \log(1/\delta)/\varepsilon^2$*

space. In particular, if k is constant then the running time is in $O\left(n^{1+o(1)}\delta^{o(1)}\right)$ and the space is in $O\left((n\delta)^{o(1)}/\varepsilon^2\right)$.

Proof. Let $\beta := \left\lceil n^{1/\log\log(n)}\right\rceil$ and $\tilde{\beta} = \lceil n/\beta \rceil$ be the values that are defined in the call to CORESET$(P, k, \varepsilon, \delta)$. Let $(B_1, \cdots, B_{\beta'})$ be the output of the call to BATCH-APPROX$(P, \tilde{\beta})$ in Line 4 of Algorithm 5; see Algorithm 2. Let $i \in [\beta']$ and identify $(\tilde{P}_i, q_i) := B_i$. Let $P_i := \left\{(x, y) \mid (x, \cdot) \in \tilde{P}_i\right\}$, i.e., $\{P_1, \cdots, P_\psi\}$ is the partition computed in the call to BATCH-APPROX$(P, \tilde{\beta})$. By the construction B_i in Line 3 of Algorithm 2 we have $q_i \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell(P_i, q)$. Let $\mathcal{Q}$ be the union over the pairs $(c, c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \text{poly}(c', x) \neq 0$ for every $(x, \cdot) \in P_i$. Hence, for every $q \in \mathcal{Q}$ by assigning $P := P_i, B := \{(P, q)\}, P^* := \tilde{P}_i$ and $\alpha, \beta = 1$ in Corollary 41, we obtain

$$\ell(\tilde{P}_i, q) \leq 2 \cdot \ell(P_i, q). \tag{4.42}$$

Let $\left\{B'_1, \cdots, B'_\psi\right\}$ be a set of biciterias, as defined as in Line 6 during the first iteration of the "while" loop in the call to Algorithm 5; see Definition 5. Let $i \in [\psi]$, and $P' := \bigcup_{(Y,q') \in B \in B'_i} Y$, i.e., the union of all the sets of points in the (α', β')-approximations in B'_i. Let $B_i :=$ REDUCE (B'_i, λ_1) as computed in Algorithm 5. Identify $B_i = \left\{(\tilde{P}_1, q_1), \cdots, (\tilde{P}_r, q_r)\right\}$, and, for every $a \in [r]$, put $P'_a := \left\{(x, y) \in P' \mid (x, \cdot) \in \tilde{P}_a\right\}$, i.e. every P'_a is the set of points in P' that are approximated by q_a. Replacing ε with $1/10$, B with B'_i and δ with $\delta/(4n)$ in Lemma 11 yields that there is λ_1 as defined in Line x of Algorithm 5 such that B_i with probability at least $1 - \delta/(4n)$ is an $(2, |B_i|)$-approximation to P'. That is, with probability at least $1 - \delta/(4n)$ we have

$$\sum_{a=1}^{r} \ell(P'_a, q_a) \leq 2 \min_{q \in \left(\mathbb{R}^k\right)^2} \sum_{a=1}^{r} \ell(P'_a, q). \tag{4.43}$$

For every $B_a \in B'_i$, let $P_a = \{(x, y) \in P \mid x \in P'_a\}$, i.e., the points in P which are approximated by the (α', β')-approximation B_a; see Definition 5. Plugging Equation (4.42) in the right side of Equation 4.43 yields that, with probability at least $1 - \delta/(4n)$, we have

$$\sum_{a=1}^{r} \ell(P_a, q^*_a) \leq 4 \min_{q \in \left(\mathbb{R}^k\right)^2} \sum_{a=1}^{r} \ell(P_a, q).$$

That is, with probability at least $1-\delta/(4n)$, B_i is a $(4,|B_i|)$-approximation to $\bigcup_{a=1}^{r} P_a$. Let $\mathcal{Q}$ be the union over the pairs $(c,c') \in \left(\mathbb{R}^k\right)^2$ such that $1 + x \cdot \text{poly}(c',x) \neq 0$ for every $(x,\cdot) \in P_i^*$. Put . Hence, with probability at least $1-\delta/(4n)$, for every $q \in \mathcal{Q}$ by assigning $P := \bigcup_{a=1}^{r} P_a, B := B_i, P^* := \bigcup_{a=1}^{r} \tilde{P}_a, \alpha = 4$ and $\beta = |B_i|$ in Corollary 41, we obtain

$$\sum_{a=1}^{r} \ell(\tilde{P}_a, q) \leq 5 \sum_{a=1}^{r} \ell(P_a, q).$$

By the construction of Algorithm 5 there are $\lceil \log\log(n) \rceil - 2$ iterations of the "while" loop, and $2n$ calls to REDUCE. Hence, repeating the proof above recursively for every iteration of the "while" loop and the last call to REDUCE in Line 10 yields that with probability at least $(1-\delta/(4n))^{2n} \geq 1-\delta/2$ we have that B computed in Line 10 is an $\left(3^{\lceil \log\log(n) \rceil}, \beta^*\right)$-approximation to P, for some integer $\beta^* \geq 1$.

By the construction of Algorithms [4,5], and that there are $\lceil \log\log(n) \rceil - 2$ iterations of the "while" loop we have $\beta^* \leq (24k)^{\lceil \log\log(n) \rceil}$. Therefore, combining this with Lemma 8 yields that there is λ_2 as defined in the call to Algorithm 5, and proves the theorem.

Space complexity. Let λ_1 as defined in Algorithm 5 (conditions were set in the previous part of the proof). From the previous section we have that B is a $O\left(\log(n), \log(n)^{O(1)+\log(k)}\right)$-approximation of P. Hence, by the construction of Algorithm 1 and assigning λ_2 (conditions were set in the previous section) we have that (B,C,w) can be represented in $2^{O(k)} \cdot \log(n)^{O(1)+k} \cdot \log(1/\delta)/\varepsilon^2$ space, which proves the claim for memory size.

Running time. Note that for every iteration of the "while" loop in the call to CORESET(P,k,ε,δ):

- By Lemma 11 and the previous analysis, we have that each call to REDUCE(B_i', λ_1) in CORESET(P,k,ε,δ) takes $n_i \cdot \beta^{O(k)} + (\beta\beta^*\lambda_1)^{O(k)}$, where $n_i = \sum_{B_i \in B_i'} \sum_{(Y,q') \in B_i} |Y|$.
- Put $\psi \leq n$ as the number of sets B_i' computed in this iteration of the "while" loop.

- Hence, since $n = \sum_{i=1}^{\psi} n_i$, we have that the computation time of the iteration is in

$$\sum_{i=1}^{\psi} \left(n' \cdot \beta^{O(k)} + (\beta\beta^* \lambda_1)^{O(k)} \right) = n \cdot \beta^{O(k)} + \psi(\beta\beta^* \lambda_1)^{O(k)} \leq n \cdot \beta^{O(k)} \cdot (\beta^* \lambda_1)^{O(k)}.$$

By the computation of λ_1, β in the call to CORESET$(P, k, \varepsilon, \delta)$, and that $\beta^* \in \log(n)^{O(1)+\log(k)}$ (from the previous sections) yields that the computation time of the iteration is in

$$2^{O(k^2)} \cdot n \cdot n^{O(k)/\log\log(n)} \cdot \log(n)^{O(k\log(k))} \cdot \log(1/\delta)^{O(k)}.$$

By the construction of Algorithm 5 there are $\lceil \log\log(n) \rceil - 2$ iterations of the "while" loop. Hence, the while-loop takes $2^{O(k^2)} \cdot n \cdot n^{O(k)/\log\log(n)} \cdot \log(n)^{O(k\log(k))} \cdot \log(1/\delta)^{O(k)}$ time. Therefore, we have that the running time of the call to Algorithm 5 is in

$$2^{O(k^2)} \cdot n \cdot n^{O(k)/\log\log(n)} \cdot \log(n)^{O(k\log(k))} \cdot \log(1/\delta)^{O(k)}. \qquad \square$$

The running time of the algorithms is still large.

Therefore, we suggest a heuristic to run on top of our coreset. We later prove that, under some assumptions, this heuristic gives a constant factor approximation.

4.7 Algorithm 6; Fast practical heuristic

Unfortunately, since for the slow solver we used [54] which utilizes polynomial programming which takes significant running time, the running time of our robust algorithms is still large. Therefore, we suggested a heuristic in Algorithm 6 to run on top of our coreset. In this section we prove that, under some assumptions, this heuristic gives a constant factor approximation.

For the sake of the analysis of Algorithm 6 we prove the following result for fitting a hyperplane to a set of points.

4.7.1 Fitting plane to points

In Lemma 45 we will prove that a common heuristic for fitting hyperplanes to points does give approximation guarantees for points of bounded coordinates. This result would later be combined with some assumptions to give the desired approximation guarantees for Algorithm 6.

Lemma 43. *Let* $\{(x_1, y_1) \cdots, (x_n, y_n)\}$ *be a set of* $n \geq 1$ *points on the plane, where* $x_i \neq 0$ *for every* $i \in [n]$*, let* $a := \frac{1}{n} \sum_{i=1}^{n} |x_i|$*, and let* $k \geq 1$ *an integer such that*

$$\forall i \in [n] : \frac{a}{k} \leq |x_i| \leq ka. \tag{4.44}$$

For every $i \in [n]$ *let* $c_i := \frac{y_i}{x_i}$*, and let* $G := \{c_1, \cdots, c_n\}$ *be a multi-set of size* n*. Let* c *be a random item from* G*, sampled uniformly. Then, with probability at least* $1/2$ *we have*

$$\sum_{(x_i,y_i)\in P} |x_i \cdot c - y_i| \leq \left(2k^2 + 1\right) \cdot \min_{c' \in \mathbb{R}} \sum_{(x_i,y_i)\in P} |x_i \cdot c' - y_i|. \tag{4.45}$$

Proof. Let $c^* \in \arg\min_{c' \in \mathbb{R}} \sum_{(x,y)\in P} |x \cdot c' - y|$. By Markov's inequality, with probability at least $1/2$ we have

$$|c - c^*| \leq \frac{2}{n} \sum_{c' \in G} |c^* - c'| := k'. \tag{4.46}$$

Suppose this event indeed occurs. For every $(x, y) \in P$ and $c' \in G$ such that $xc' = y$ (there exist such c' by the definition of G in the lemma), we have

$$\begin{aligned}
|xc - y| &= |xc - xc'| && (4.47)\\
&= |x| \cdot |c - c'| && (4.48)\\
&= |x| \cdot |c - c^* - c' + c^*| && (4.49)\\
&\leq |x| \cdot \left(|c^* - c| + |c' - c^*|\right) && (4.50)\\
&\leq |x|k' + |x| \cdot |c^* - c'| && (4.51)\\
&\leq kk'a + |c^*x - y|, && (4.52)
\end{aligned}$$

where Equation (4.47) is by the choice of $c' \in G$ as a value satisfying $xc' = y$, Equation (4.48) and (4.49) are by reorganizing the expression, Equation (4.50) is by the triangle inequality, and Equation (4.51) is by the definition of k' from Equation (4.46), and Equation (4.52) is by assigning that $|x| \leq ka$ (from the definition of P in the lemma, substituting $x_i := x$ in Equation (4.44)) and that $xc' = y$ (from the choice of $q \in G$).

Hence,

$$\sum_{(x,y)\in P} |xc - y| - \sum_{(x,y)\in P} |xc^* - y| \leq nkk'a \tag{4.53}$$

$$= 2k \sum_{i=1}^{n} a|c_i - c^*| \tag{4.54}$$

$$= 2k \cdot \sum_{i=1}^{n} \frac{a|x_i c_i - x_i c^*|}{|x_i|} \tag{4.55}$$

$$= 2k \cdot \sum_{(x,y)\in P} \frac{a|y - xc^*|}{|x|} \tag{4.56}$$

$$\leq 2k \cdot \sum_{(x,y)\in P} k|y - xc^*| \tag{4.57}$$

$$= \left(2k^2\right) \cdot \sum_{(x,y)\in P} |xc^* - y|, \tag{4.58}$$

where Equation (4.53) is by summing over every $(x, y) \in P$ utilizing Equations (4.47) to (4.52), Equation (4.54) is by assigning the definition of k' from Equation (4.46), Equation (4.55) is by multiplying and dividing the expression inside the sum by a_i, Equation (4.56) follow from the definition of c_i as $\frac{y_i}{x_i}$ in the lemma, Equation (4.57) is by Equation (4.44), and Equation (4.58) is by reorganizing the expression. □

By assuming that similar assumptions hold for every dimension separately we can generalize the previous lemma for higher dimensions, as done in the following lemma.

Lemma 44. *Let* $P = \left\{ \left(\left(x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(d)}\right), b_i \right) \right\}_{i=1}^{n} \subset \mathbb{R}^d \times \mathbb{R}$ *be a set of* $n \geq 1$ *points, where for every* $(i, j) \in [n] \times [d]$ *we have* $x_i^{(j)} \neq 0$*, let* $a^{(j)} := \frac{1}{n} \sum_{i=1}^{n} |a_i^{(j)}|$ *for every* $j \in [d]$*, and let* $k \geq 1$

be an integer such that

$$\forall (i,j) \in [n] \times [d] : \frac{a^{(j)}}{k} \leq |a_i^{(j)}| \leq a^{(j)} k.$$

For every $S \subset P$ of size d let $c_S \in \mathbb{R}^d$ such that $\left|a^T \cdot c_S - y\right| = 0$; there is such value by properties of linear regression. Let G be the multi-set $\{c_S \mid S \subset P, |S| = d\}$. Let c be a random item from G, sampled uniformly. Then, with probability at least 2^{-d} we have

$$\sum_{(a,y)\in G} \left|a_i^T \cdot c - y_i\right| \leq \left(2k^2+1\right)^d \cdot \min_{c' \in \mathbb{R}^d} \sum_{(a,y)\in G} \left|a_i^T \cdot c' - y_i\right|. \tag{4.59}$$

Proof. Let $c' = (c'_1, \cdots, c'_d) \in \mathbb{R}^d$, and $j' \in [d]$. For every $i \in [n]$ let $x_i := x_i^{(j')}$, let $y'_i := y_i + \sum_{j=1}^{d} x_i c'_i - x_i c'_d$, and let $c_i := \frac{y'_i}{x_i}$. Let $P' := \{(x_1, y'_1), \cdots, (x_n, y'_n)\}, G := \{c_1, \cdots, c_n\}$ be multi sets, both of size n. Let c be a random item from G', sample uniformly, and let c^* be c' were we substitute the j'th entry by c. By the definition of P' and c' we have

$$\sum_{(x,y)\in P} |x^T \cdot c^* - y| = \sum_{(x,y)\in P'} |x \cdot c - y|. \tag{4.60}$$

From the definition of P in the lemma, for every $(i,j) \in [n] \times [d]$ we have $\frac{a^{(j)}}{k} \leq \left|a_i^{(j)}\right| \leq k a^{(j)}$. Hence, by the construction of P' we have $a^{(j')} := \frac{1}{n} \sum_{(x,y)\in P'} |x|$, and for every $(x,y) \in P'$ we have $\frac{a^{(j')}}{k} \leq |x| \leq k a^{(j')}$. Thus, by substituting P by P' and G by G' in Lemma 43, with probability at least $1/2$, we have

$$\sum_{(x,y)\in P'} |x \cdot c - y| \leq (2k^2+1) \cdot \sum_{(x,y)\in P'} |x \cdot c'_i - y|.$$

Combining this with (4.60) yields that with probability at least $1/2$ we have

$$\sum_{(x,y)\in P} |x^T \cdot c^* - y| \leq (2k^2+1) \cdot \sum_{(x,y)\in P'} |x \cdot c'_i - y| = (2k^2+1) \sum_{(x,y)\in P} |x^T \cdot c' - y|,$$

where the equality is by the construction of P'.

Let $c' = (c'_1, \cdots, c'_d) \in \arg\min_{c' \in \mathbb{R}^d} \sum_{(x,y)\in P} |x^T c' - y|$, and let $(c_1, \cdots, c_d) := c$. For every $j \in [d]$ let $c^*_j = (c_1, \cdots, c_j, c'_{j+1}, \cdots, c'_d)$. Let $c^*_0 = c'$. By the proof above, for every $j \in [d]$, with probability at least $1/2$, we have

$$\sum_{(x,y)\in P} |x^T \cdot c^*_j - y| \leq (2k^2+1) \cdot \sum_{(x,y)\in P} |x^T \cdot c^*_{j-1} - y|.$$

Assigning $c^*_0 = c' \in \arg\min_{c' \in \mathbb{R}^d} \sum_{(x,y)\in P} |x^T c' - y|$ and $c^*_d = c$ in the above, combined with the construction of G in the lemma yields Equation (4.59). □

By repeatedly taking a sample from the set G in the lemma above we obtain the following lemma that gives the desired approximation guarantees for linear regression under assumptions.

Lemma 45. *Let $P = \left\{ \left(\left(a_i^{(1)}, a_i^{(2)}, \cdots, a_i^{(d)}\right), b_i \right) \right\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$ be a set of $n \geq 1$ points, where there is an integer $k \geq 1$ such that*

$$\forall (i,j) \in [n] \times [d] : \frac{1}{nk} \sum_{i'=1}^n |a_{i'}^{(j)}| \leq |a_i^{(j)}| \leq \frac{k}{n} \sum_{i'=1}^n |a_{i'}^{(j)}|. \tag{4.61}$$

For every $S \subset P$ of size d let $c_S \in \mathbb{R}^d$ such that $\sum_{(a,y)\in S} \left|a^T \cdot c - y\right|$; there is such value by properties of linear regression. Let G be the multi-set that is the union over every $S \subset P$ of size $|S| = d$. Let $\delta \in (0,1), \varepsilon = 2^{-d}$, and let $\lambda := \max\left\{ \left\lceil \log_{1-\varepsilon}(\delta) \right\rceil, 1 \right\}$. Let $S \subset G, |S| = \lambda$ where each value is sampled i.i.d. and uniformly. With probability at least $1 - \delta$ there is $c \in S$ such that

$$\sum_{(x,y)\in P} |x^T c - y| \leq (2k^2+1)^d \cdot \min_{c' \in \mathbb{R}^d} \sum_{(x,y)\in P} |x^T c' - y|. \tag{4.62}$$

Proof. By Lemma 44, for a uniformly sampled $c \in G$ with probability most $1 - \varepsilon$ we have

$$\sum_{(x,y)\in P} |x^T c - y| > (2k^2+1)^d \cdot \min_{c' \in \mathbb{R}^d} \sum_{(x,y)\in P} |x^T c' - y|. \tag{4.63}$$

Hence, by the union bound, the probability that (4.63) holds for every $c \in S$ is at most

$$(1-\varepsilon)^{\lambda} \leq (1-\varepsilon)^{\log_{1-\varepsilon}\delta} = \delta,$$

where the inequality is since $\lambda \geq \log_{1-\varepsilon}\delta$. □

4.7.2 Computing SOLVER; RFF interpolation

In this section we prove, for $S \subset \mathbb{R}^2$ of size $2k$ satisfying $\left|\{x \cdot y \mid (x,y) \in S\}\right| = 2k$, that SOLVER$(S)$ is never empty and that it can be computed in $O\left(k^3\right)$ time; see Definition 13.

This method is a generalization of a technique shown in [44]. While it is plausible that there are previous work that show the robustness of an equivalent method to ours, this section is still important for the self containment of the work.

For this we define the following global definitions which would be used in the following section.

Definition 46. *Let $S = (x_1,y_1),(x_2,y_2),\cdots,(x_{2k},y_{2k}) \in \mathbb{R}^2$, where $\{x \cdot y \mid (x,y) \in S\}$ and $\{x \mid (x,\cdot) \in S\}$ are both of size $2k$. Let $A_1, A_2 \in \mathbb{R}^{(2k)\times k}$ as follows*

$$A_1 = \begin{pmatrix} 1, x_1, x_1^2, \cdots, x_1^{k-1} \\ 1, x_2, x_2^2, \cdots, x_2^{k-1} \\ \vdots \\ 1, x_{2d}, x_{2d}^2, \cdots, x_{2k}^{k-1} \end{pmatrix}, A_2 = \begin{pmatrix} -y_1 \cdot x_1, -y_1 \cdot x_1^2, \cdots, -y_1 \cdot x_1^k \\ -y_2 \cdot x_2, -y_2 \cdot x_2^2 \cdots, -y_2 \cdot x_2^k \\ \vdots \\ -y_{2k} \cdot x_{2k}, -y_{2k} \cdot x_{2k}^2 \cdots, -y_{2k} \cdot x_{2k}^k \end{pmatrix}. \tag{4.64}$$

Let $A = (A_1 \mid A_2)$ be a $(2k) \times (2k)$ matrix, and let $\tilde{x}, \tilde{y} = (x_1, x_2, \cdots, x_{2k}), (y_1, y_2, \cdots, y_{2k})$.

In the following lemma we derive conditions that any value that satisfies them is a candidate for an output to SOLVER(S).

Lemma 47. *Suppose that there is $b = (b_1, b_2, \cdots, b_{2k}) \in \mathbb{R}^{2k}$ such that the following holds:*

(i) It holds that $A \cdot b = \tilde{y}$, i.e. b is the solution to the constructed linear equation.

(ii) For every $(x, y) \in S$ *we have* $1 + x \cdot \text{poly}\Big((b_{k+1}, b_{k+2}, \cdots, b_{2k}), x\Big) \neq 0$.

Put $c = (b_1, b_2, \cdots, b_k)$ *and* $c' = (b_{k+1}, b_{k+2}, \cdots, b_{2k})$. *We have that*

$$\forall p \in S : D(c, c', p) = 0.$$

Proof. Using the notation from the lemma we obtain

$$A \cdot b = \tilde{y} \tag{4.65}$$

$$A_1 \cdot c + A_2 \cdot c' = \tilde{y} \tag{4.66}$$

$$\forall (x, y) \in S : \text{poly}(c, x) - y \cdot x \cdot \text{poly}(c', x) = y \tag{4.67}$$

$$\forall (x, y) \in S : \text{poly}(c, x) = y + y \cdot x \cdot \text{poly}(c', x) \tag{4.68}$$

$$\forall (x, y) \in S : \text{poly}(c, x) = y \cdot \Big(1 + x \cdot \text{poly}(c', x)\Big) \tag{4.69}$$

$$\forall (x, y) \in S : \frac{\text{poly}(c, x)}{1 + x \cdot \text{poly}(c', x)} = y, \tag{4.70}$$

where (4.65) is by property (i) satisfied by b, (4.66) is by the definition of c, c' as $c = (b_1, b_2, \cdots, b_k)$ and $c' = (b_{k+1}, b_{k+2}, \cdots, b_{2k})$, and by the definition of A as $A = (A_1 \mid A_2)$, (4.67) is by the construction of A_1 and A_2 in (4.64), (4.68) is by adding $y \cdot x \cdot \text{poly}(c', x)$ to both sides of the equation, (4.69) is by rewriting the right side of the equation, (4.70) is by dividing both sides of the equation by $1 + x \cdot \text{poly}(c', x)$, which is a legal operation by property (i) satisfied by b. □

In the following lemma we will prove the existence of b as defined in the previous lemma.

Lemma 48. *There is* $b = (b_1, b_2, \cdots, b_{2k}) \in \mathbb{R}^{2k}$ *such that the following holds:*

(i) It holds that $A \cdot b = \tilde{y}$, *i.e.* b *is the solution to the constructed linear equation.*

(ii) For every $(x, y) \in S$ *we have* $1 + x \cdot \text{poly}\Big((b_{k+1}, b_{k+2}, \cdots, b_{2k}), x\Big) \neq 0$.

Proof. **Proving that** A **is invertible, i.e., proving property (i):**

Recall the definition of A_1, A_2, and A from Definition 46. Let

$$\begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = A_1, \begin{pmatrix} B_3 \\ B_4 \end{pmatrix} = A_2, (B_1, B_2, B_3, B_4) \in \mathbb{R}^{k \times k}.$$

Since B_1 and B_2 are Vandermonde matrix for different x-value $\det(B_1), \det(B_2) \neq 0$; the x-values are different by the definition of S from Definition 46. Hence, by block matrix properties for A to be invertible it suffices that $\det\left(B_4 - B_2 B_1^{-1} B_3\right) \neq 0$; see [52]. Let $\tilde{y}_1, \tilde{y}_2, \tilde{x}_1, \tilde{x}_2 \in \mathbb{R}^k$ such that $(\tilde{y}_1 \mid \tilde{y}_2) = \tilde{y}$, $(\tilde{x}_1 \mid \tilde{x}_2) = \tilde{x}$; the definition of $\tilde{x}$ and $\tilde{y}$ in Definition 46. For every $c \in \mathbb{R}^k$ let $\text{diag}(c) \in \mathbb{R}^{k \times k}$ be a diagonal matrix, whose diagonal is c. Let $Y_1 := -\text{diag}(\tilde{y}_1), Y_2 := -\text{diag}(\tilde{y}_2), X_1 := \text{diag}(\tilde{x}_1)$, and $X_2 = \text{diag}(\tilde{x}_2)$ be $k \times k$ matrices. By the construction of A and the previous definitions we have $(B_3, B_4) = (B_1 Y_1 X_1, B_2 Y_2 X_2)$. Hence,

$$\begin{aligned} B_4 - B_2 \cdot B_1^{-1} \cdot B_3 = B_4 - B_2 \cdot B_1^{-1} \cdot B_1 Y_1 X_1 = \\ B_4 - B_2 Y_1 X_1 = B_2 Y_2 X_2 - B_2 Y_1 X_1 = B_2 \cdot \Big(Y_2 X_2 - Y_1 X_1\Big). \end{aligned} \tag{4.71}$$

By the definition of S in Definition 46 we have $\det\left(Y_2 \cdot X_2 - Y_1 \cdot X_1\right) \neq 0$. Hence, assigning that $\det(B_2) \neq 0$ yields that $\det\left(B_4 - B_2 \cdot B_1^{-1} \cdot B_3\right) \neq 0$, and consecutively that A is invertible.

Proving property (ii):

Let $b = (b_1, b_2, \cdots, b_{2k}) \in \mathbb{R}^{2k}$ such that $A \cdot b = \tilde{y}$; there is such value by the proof that A is invertible in the previous part of the proof. Let $c' = (b_{k+1}, b_{k+2}, \cdots, b_{2k})$.

It can be seen that even if there is $(x, y) \in S$ such that $1 + x \cdot \text{poly}(c', x) = 0$, we can add arbitrarily small noise to the y values (without destroying the condition from the previous part), which would tweak c' such that for every $(x, y) \in S$ we would have $1 + x \cdot \text{poly}(c', x) \neq 0$. □

In the following lemma we combine the previous lemmas to obtain the previously mentioned desired properties of SOLVER.

Lemma 49. *We have that* SOLVER(S) *is never empty and can be computed in* $O\left(k^3\right)$ *time.*

Proof. Let A and $\tilde{y}$ as defined in Definition 46. By the proof of Lemma 48 we have $\deg(A) \neq 0$. Let $b \in \mathbb{R}^{2k}$ such that $A \cdot b = \tilde{y}$. Let $c = (b_1, b_2, \cdots, b_d)$ and $c' = (b_{k+1}, b_{k+2}, \cdots, b_{2k})$.

By the of Lemma 48 for every $(x, y) \in S$ we have $1 + x \cdot \text{poly}(c', x) \neq 0$. Therefore, since b satisfies all the conditions in Lemma 47, (c, c') is a candidate for an output for SOLVER(S).

Let $(c^*, c'^*) \in \left(\mathbb{R}^k\right)^2$ such that for every $p \in S$ we have $D(c, c', p) = 0$. By the proof of Lemma 47 we have that $b^* = (c^*|c'^*)$ satisfies that $A \cdot b^* = \tilde{y}$. Since $\deg(A) \neq 0$ we have that there is a unique $\tilde{b} \in \mathbb{R}^{2k}$ such that $A \cdot \tilde{b} = \tilde{y}$. Therefore, $(c^*, c'^*) = (c, c')$, which yields that (c, c') is the unique candidate for a solution to SOLVER(S).

Computation time:

By the previous part of the proof we have that SOLVER$(S) = (c, c')$. The constructing of A in Definition 46 takes $O(k^3)$ time. Solving a system of linear equations with $2k$ equations and $2k$ parameters can be done in $O(k^3)$ time. Hence, we can compute b in $O(k^3)$, which yields that (c, c') can be computed in $O(k^3)$ time. □

4.7.3 Correctness of Algorithm 6

In the following lemma we combine the previous result with some assumptions to obtain the desired guarantees for Algorithm 6.

Lemma 50. *Let $P = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subset \mathbb{R}^2$, where $n \geq 2k$, be a set of points with unique first coordinates with non equal zero, and where for every $S \subset P$ of size $2k$ we have $\left|\{x \cdot y \mid (x, y) \in S\}\right| = 2k$. For every $(x_i, y_i) \in P$ let $\tilde{x}_i = (x^* \mid -(x_i y_i) \cdot x^*) \in \mathbb{R}^{2k}$, where $x^* = (1, x_i, x_i^2, \cdots, x_i^{k-1})$. For every $i \in [n]$ let $\left(a_i^{(1)}, \cdots, a_i^{(2k)}\right) = \tilde{x}_i$. Suppose there is $k' \in [1, \infty)$ such that for every $(i, j) \in [n] \times [2k]$ we have*

$$\frac{1}{nk'} \sum_{\psi \in [n]} |a_\psi^{(j)}| \leq |a_i^{(j)}| \leq \frac{k'}{n} \sum_{\psi \in [n]} |a_\psi^{(j)}|.$$

Let $\delta \in (0, 1), \varepsilon = 4^{-k}$, and let $\lambda := \max\left\{\left\lceil \log_{1-\varepsilon}(\delta) \right\rceil, 1\right\}$. Let G be the output of a call to FAST-CENTROID-SET(P, λ)*; see Algorithm 6. Let $(c_1, c_1') \in \underset{q \in \left(\mathbb{R}^k\right)^2}{\arg\min}\, \ell(P, q)$. Let $\alpha = (2k'^2 + 1)^{2k}$,*

and suppose that there is $(c, c') \in G$ *such that*

$$\sum_{(x_i,y_i)\in P} \left|\tilde{x}_i^T \cdot (c \mid c') - y_i\right| \leq \alpha \cdot \sum_{(x_i,y_i)\in P} \left|\tilde{x}_i^T \cdot (c_1 \mid c_1') - y_i\right|, \tag{4.72}$$

which, by the construction of FAST-CENTROID-SET *happens with probability at least* $1-\delta$*; follows from assigning* $P := \{(\tilde{x}_1, y_1), \cdots, (\tilde{x}_n, y_n)\}$ *in Lemma 45. Suppose that there is* $\rho \geq 1$ *such that for every* $(x, y) \in P$ *we have*

$$\left|1 + x \cdot \mathrm{poly}(c', x)\right| \leq \rho \cdot \left|1 + x \cdot \mathrm{poly}(c_1', x)\right|, \tag{4.73}$$

then, we have that

$$\ell\big(P, (c, c')\big) \leq \alpha\rho \cdot \ell\big(P, (c_1, c_1')\big).$$

Proof. We have that

$$\begin{aligned}
\ell\big(P, (c, c')\big) &= \sum_{(x_i,y_i)\in P} \left|\frac{\mathrm{poly}(c, x_i)}{1 + x \cdot \mathrm{poly}(c', x_i)} - y_i\right| && (4.74)\\
&= \sum_{(x_i,y_i)\in P} \left|\frac{\mathrm{poly}(c, x_i) - y_i \cdot \big(1 + x_i \cdot \mathrm{poly}(c', x_i)\big)}{1 + x_i \cdot \mathrm{poly}(c', x_i)}\right| && (4.75)\\
&= \sum_{(x_i,y_i)\in P} \left|\frac{|\tilde{x}_i^T \cdot (c \mid c') - y_i|}{1 + x_i \cdot \mathrm{poly}(c', x_i)}\right| && (4.76)\\
&\leq \alpha \cdot \sum_{(x_i,y_i)\in P} \left|\frac{|\tilde{x}_i^T \cdot (c_1 \mid c_1') - y_i|}{1 + x_i \cdot \mathrm{poly}(c', x_i)}\right| && (4.77)\\
&\leq \alpha\rho \cdot \sum_{(x_i,y_i)\in P} \left(\left|\frac{1 + x_i \cdot \mathrm{poly}(c', x_i)}{1 + x \cdot \mathrm{poly}(c_1', x_i)}\right| \cdot \left|\frac{|\tilde{x}_i^T \cdot (c_1 \mid c_1') - y_i|}{1 + x_i \cdot \mathrm{poly}(c', x_i)}\right|\right) && (4.78)\\
&= \alpha\rho \cdot \sum_{(x_i,y_i)\in P} \left|\frac{|\tilde{x}_i^T \cdot (c_1 \mid c_1') - y_i|}{1 + x_i \cdot \mathrm{poly}(c_1', x_i)}\right| && (4.79)\\
&= \alpha\rho \cdot \sum_{(x_i,y_i)\in P} \left|\frac{\mathrm{poly}(c_1, x_i)}{1 + x \cdot \mathrm{poly}(c_1', x_i)} - y_i\right| && (4.80)\\
&= \alpha\rho\ell\big(P, (c_1, c_1')\big), && (4.81)
\end{aligned}$$

where Equation (4.74) is by the definition of ℓ in Definition 4, Equation (4.75) is by reorganizing the expression (taking common denominator), Equation (4.76) is by assigning the construction of $\tilde{x}_i$ in the lemma, Equation (4.77) is by assigning (4.72), Equation (4.78) is by assigning (4.73), Equation (4.79) is by moving $\left|\frac{1+x\cdot\mathrm{poly}(g,x)}{1+x\cdot\mathrm{poly}(c',x)}\right|$ inside the absolute value which is a legal operation since $\forall s,t\in\mathbb{R}\ :|s|\cdot|t|=|st|$, Equation (4.80) is by reorganizing the expression and assigning the definition of $\tilde{x}_i$, and Equation (4.81) is by the definition of ℓ in Definition 4. $\square$

Chapter 5

Experimental Results

We implemented our coreset construction from Algorithm 5 and all its sub-routines in Python 3.8. In this section we evaluate its empirical results, both on synthetic and real-world datasets. Open-source code can be found in [12].

Hardware: We used a PC with an Intel Core i7-10700, NVIDIA GeForce GTX 1660 SUPER (GPU), and 16GB of RAM.

Global parameters: We focused on rational functions of degree k=2 in our experiments, since it seemed as a sufficient degree to enable a visually pleasing approximation as seen, for example, in Fig. 5-6, without being too large and "over-fitting" the data-sets.

Reducing the gap between theory and practice: In practice, we apply two minor changes to our theoretically-provable algorithms, which have seemingly neglectable effect on their output quality, but aim to improve their running times. Those changes are:

(i) Line 2 of Algorithm 2 computes $q_i \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell(P_i, q)$. Instead, to reduce the computational time we iterate only over the queries in FAST-CENTROID-SET$(P_i, 64)$, where 64 was heuristically chosen as a compromise between quality and speed; see Algorithm 6. Note that this change is essentially substituting the computation by a heuristic that yields an approximation with high probability under assumptions on the input; see Section 4.7.

(ii) In Line 3 of Algorithm 3 we substitute the precise computation of the partition presented in

Lemma 36 by an "approximated partition", where the roots of the polynomials in Lemma 36 are approximated by numeric methods. In particular, using the method `roots` from the library numpy, which seems to implement the algorithm from [29].

(iii) In Line 11 of Algorithm 3, the set S_i^j is sampled from an interval of an n-signal P_i^j, where each point $(x, y) \in P_i^j$ is sampled with probability $s'(x)$. In practice, we observed that the probabilities assigned by s' were sufficiently close to $1/|P_i^j|$ for most of the points. Hence, to reduce the computational time, we sampled the set S_i^j uniformly from P_i^j.

(iv) Line 11 of Algorithm 4 computes $q_G \in \arg\min_{q \in \left(\mathbb{R}^k\right)^2} \ell\big((S_G, w_G), q\big)$. Instead, to reduce the computational time, we iterate only over the queries in FAST-CENTROID-SET$(S_G, 64)$; see Algorithm 6. The choice of 64 and the implication of this change are as in property (i) above.

Tested methods. We consider the following compression schemes:

(i): `Our-RFF`(P, λ) - The implementation based on Algorithm 5, where we set the internal parameters as $\beta = 32$, $\tilde{\beta} = 32$, $\lambda_1 = 32$, and $\Lambda = 0$.

(ii): `Our-FRFF`(P, λ) - A heuristic modification to `Our-RFF` above, where the call to REDUCE at Line 7 of Algorithm 5 is replaced with a call to FAST-CENTROID-SET; see Algorithm 6. This should boost the running time by slightly compromising accuracy.

The Scipy library [67] provides a function called `Scipy.optimize.minimize` from the `optimize` library [67]. The function takes as input a function $f : \mathbb{R}^m \to \mathbb{R}$, where m is a positive integer, and aims to find its minimum.

(iii): `Gradient`(P, λ) - A rational function $q \in \left(\mathbb{R}^k\right)^2$ was fitted to the input signal via the function `Scipy.optimize.minimize` with the default values, where the function minimized is $f : \left(\mathbb{R}^k\right)^2 \to [0, \infty)$ such that for every $q \in \left(\mathbb{R}^k\right)^2$ we define $f(q) = \ell(P, q)$, and the starting position is $\left(\{0\}^k\right)^2$. Then, a coreset was constructed using the algorithm SAMPLE-CORESET$(\{q\}, \{P\}, \lambda)$.

(iv): L_∞`Coreset`(P, λ) - Similarly to the previous method, this method also aims to compute an approximated query which minimizes $\ell(P, q)$ over $q \in \left(\mathbb{R}^k\right)^2$, and then samples a coreset using the algorithm SAMPLE-CORESET$(\{q\}, \{P\}, \lambda)$. The rational function here is $q \in \arg\min_{q' \in \left(\mathbb{R}^k\right)^2} \max_{p \in P} D(p, q')$, that was computed based on [51].

(v): `RandomSample`(P, λ) - returns a uniform random sample of size $1.5 \cdot \lambda$ from P. The multiplication by 1.5 is to account for not saving the weights and to yield an equal coreset size for fair comparison.

(vi): `NearConvexCoreset`(P, λ) - The coreset construction from [65], chosen to be of size $1.5 \cdot \lambda$. The multiplication by 1.5 is to account for not saving the weights and to yield an equal coreset size for fair comparison. This paper computes a coreset for the family of near convex functions; as defined there. This is done also via sensitivity sampling. We compute the coreset by assuming heuristically that the function $f : \mathbb{R}^2 \times \mathbb{R}^{2k}$ that maps every $p \in \mathbb{R}^2$ and $q := (c, c') \in \left(\mathbb{R}^k\right)^2$ to $f(p, c \mid c') = D(p, q)$, is near convex.

Coreset size. In all the following experiments, for fair comparison, the input parameters of all the competing methods above were tuned as to obtain an output coreset of the same desired size.

Evaluation. The quality of a given compression scheme (from the competing methods above) is defined as $\varepsilon(q) = 100 \cdot \dfrac{\left|\ell'(q) - \ell(P, q)\right|}{\ell(P, q)}$, where $\ell'(q)$ is the loss computed by the compression scheme for the query $q \in \left(\mathbb{R}^k\right)^2$, and $\ell(P, q)$ is the loss of the query for the full input signal as defined in (3.1). We tested a couple of different options for such a query q: **(i):** as it is hard to compute the optimal solution q^* which minimizes (3.1), we sampled a set Q of 1024 queries using Algorithm 6 and recovered the query $q \in Q$ which has the smallest loss over the full data. This query aims to test the compression accuracy in a near-optimal scenario. We then computed $\varepsilon(q)$ for the query q above. **(ii):** For every individual compression scheme, we picked the query in $q \in Q$, which yields the largest value for $\varepsilon(q)$; i.e., the worst approximation bound.

5.1 Synthetic data

In this experiment, the input is an auto-regression signal that corresponds to a homogeneous recurrence representation of degree k, with additional noise. The goal is to reconstruct the original signal via fitting to a rational function, as explained in Section 1 (specifically, see Definition 1 and Proposition 2).

Dataset. Following the motivation in Section 1, the data (n-signal) P we used in this experiment is simply the set $\{F(x) \mid x = j/n - 1/2, j \in [n]\}$ of n values of the generating function of a noisy Fibonacci series, i.e., $F(x) = \sum_{i=1}^{100} \left(s_{i+1} \cdot x^i \right)$ where s_i is the ith element in the Fibonacci series $s_{i+2} = s_{i+1} + s_i$, $s_1 = s_1 = 2$, with Gaussian noise, with zero mean and a standard deviation of 0.25 that is added for every value of x above, independently.

Results. Fig. 5-1 presents the results averaged across 100 tests, along with error bars that present the 25% and 75% percentiles.

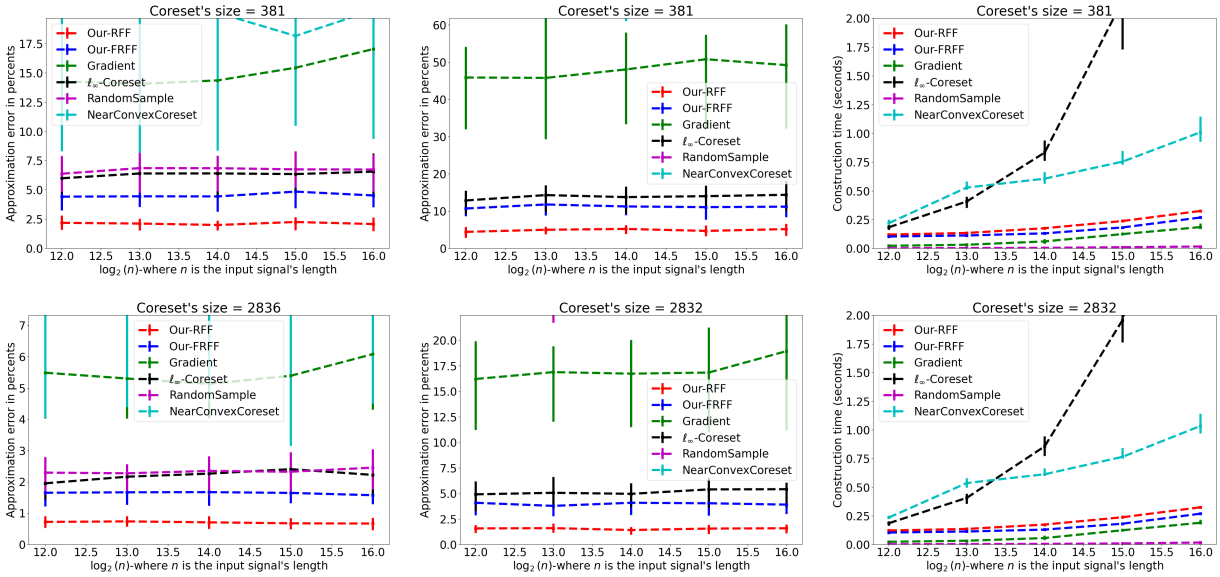

Figure 5-1: Results of the experiment from Section 5.1. ***(Left + Middle):*** *The x-axis shows, on a logarithmic scale, the length n of the input signal, and the y-axis shows the approximation error of each compression scheme, for given compression sizes of 382 and 2824 memory words respectively. The left and right rows shows Evaluations (i) and (ii) respectively.* ***(Right):*** *The running time for each of the two coreset sizes. The evaluation method does not affect those times.*

5.2 Real-world data

In the following experiments we consider an input signal which contains n readings of some quantity (e.g., temperature or pressure) over time. We aim to fit a rational function to such a signal, with the goal of predicting future readings and uncovering some behavioral pattern.

5.2.1 Air Quality Dataset

In this experiment we consider the **Beijing Air Quality Dataset** [11] from the public UCI Machine Learning Repository [5]. The dataset contains $n \in \{7344, 8760, 8760\}$, and 8784 readings, respectively for the years 2013 to 2016. Each reading contains the temperature (degree Celsius, denoted by TEMP), pressure (hPa, denoted by PRES), and dew point temperature (degree Celsius, denoted by DEWP). For each year and feature individually, we construct an n-signal of the readings over time. Missing entries were replaced with the average feature value. Fig. 5-2 presents the dataset readings along with the approximation computed in `Our-FRFF`, Scipy's rational function fitting computed via `Scipy.optimize.minimize`, and a 3th degree polynomial computed using the `numpy.polyfit` function, which minimizes the sum of squared distances between the polynomial and the input. For fair comparison, all three methods have been allowed 4 free parameters. In particular, our rational function and Scipy's rational function have degree 1 in the enumerator and 2 in the denominator (there are 4 free parameters, since the free variable in the denominator is set to 1), while the polynomial is of degree 3.

Observe that in the following examples the 3th degree polynomial yielded a slightly smaller loss than our method, this is in contrast to the case in the example in Fig. 1-1 where the 3th degree polynomial yielded significantly worse results. We believe that this occurred due the "un-noisy" data corresponding to a more "smooth" function, while in Fig. 1-1 the function was "less" smooth due to the data generation.

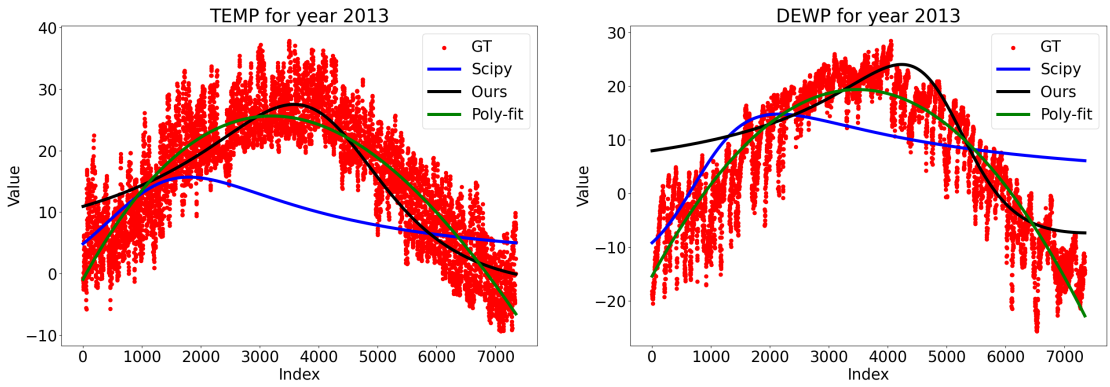

Figure 5-2: The n-signal corresponding to TEMP and DEWP properties from the year 2013 of Dataset [11], along with three fitted functions: (i) the rational function of degree 2 (i.e., degree 1 in the enumerator and 2 in the denominator) computed in our algorithm `Our-FRFF`*, (ii) the output of a call to* `Scipy.optimize.minimize` *that aims to fit a rational function of degree 2 to the input signal, and (iii) Polynomial of degree 3, computed using the* `numpy.polyfit` *function. For fair comparison, all three methods use 4 free parameters.*

Repeating the experiments in 5.1 for this dataset. Figures 5-3 and 5-4 present the results from repeating the experiments in 5.1; as previously in 5.1 the results are averaged across 100 tests and are accompanied by error bars that present the 25% and 75% percentiles.

As there is relatively little change in the computational time plot over the sample sizes and features we show the mean results for the year 2016 for the temperature feature in the following table.

Method	Time (seconds)
`Our-RFF`	0.1361
`Our-FRFF`	0.1187
`Gradient`	0.0134
L_∞`Coreset`	0.8334
`RandomSample`	0.0022
`NearConvexCoreset`	0.5045

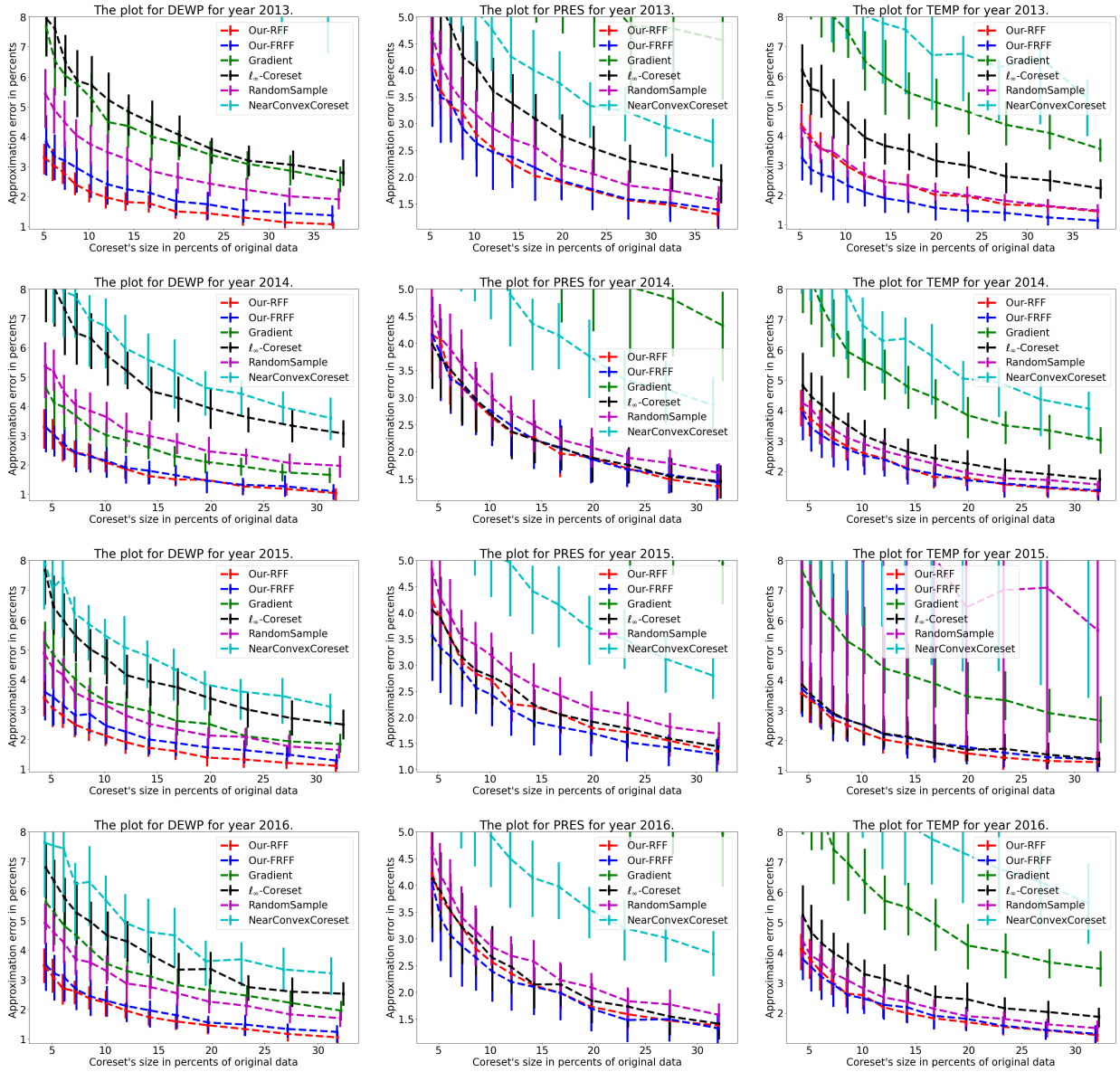

*Figure 5-3: results for the experiment from Section 5.2.1. The x-axis shows the compression ration, with respect to percents of the original data. The y-axis shows the approximation error of each compression scheme, using Evaluation method (i). The plots corresponds (from left to right) to properties DEWP, PRESS, and TEMP in the dataset [11], and to the years $2013, 2014, 2015$, and 2016 (from top to bottom). Methods **RandomSample** and **NearConvexCoreset** yielded too large errors and thus were clipped in some cases.*

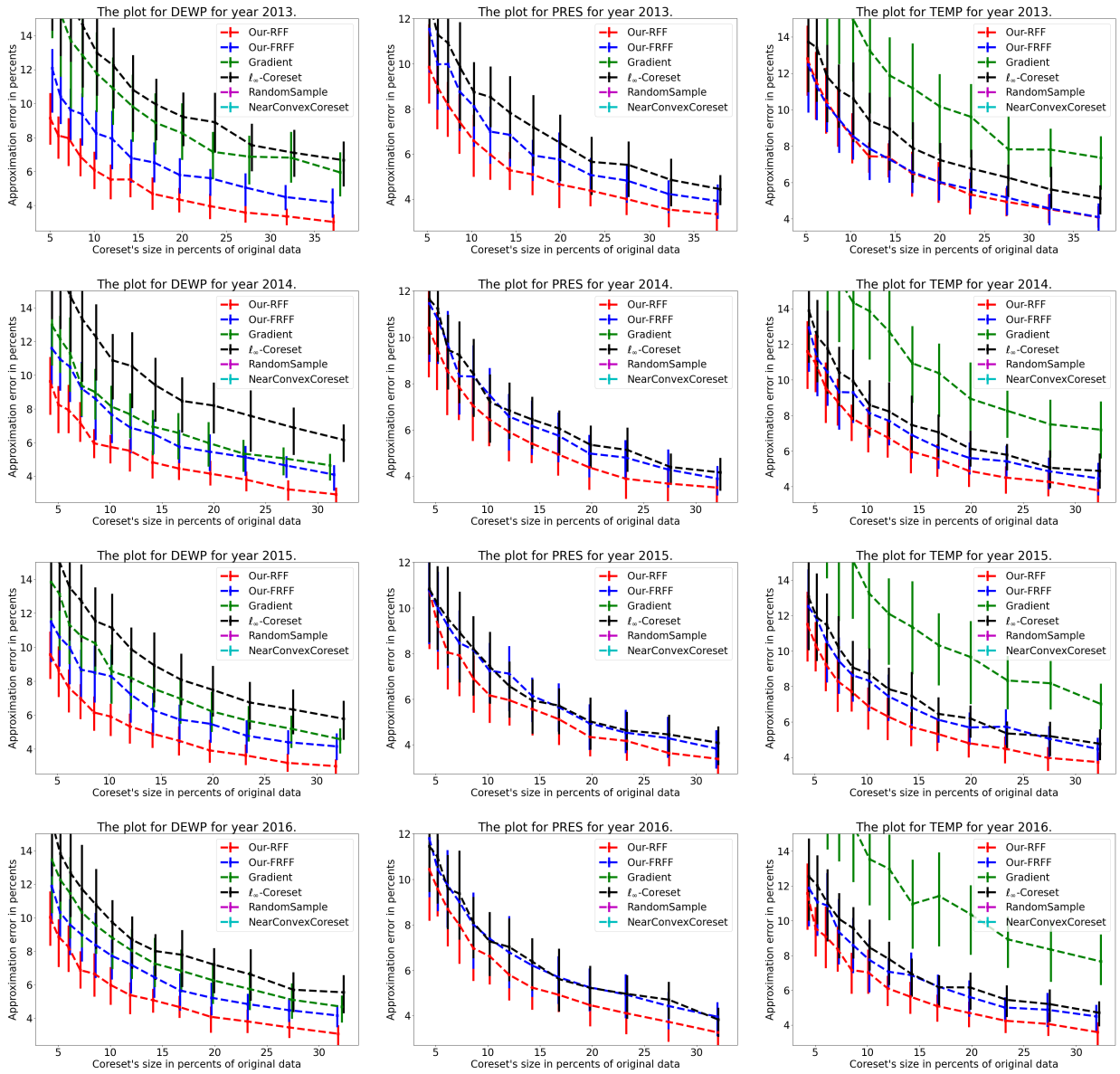

Figure 5-4: Evaluation of Method (ii), similar to Method(i) in Fig. 5-3.

5.2.2 Gas Multisensor Dataset

In this experiment we used the dataset from [68] which contains a "yearly measurement of of a gas multisensor device deployed on the field in an Italian city" (cited from [68]).

Fig. 5-5 presents the dataset readings along with the approximation computed in `Our-FRFF`, Scipy's rational function fitting computed via `Scipy.optimize.minimize`, and Polynomial

of degree 3, computed using the `numpy.polyfit` function, which minimizes the sum of squared distances between the polynomial and the input. For fair comparison, all three methods have been allowed 4 free parameters. In particular, our rational function and Scipy's rational function have degree 1 in the enumerator and 2 in the denominator (there are 4 free parameters, since the free variable in the denominator is set to 1), while the polynomial is of degree 3.

Observe (as in Fig. 5-2) that in the following examples the 3th degree polynomial yielded a slightly smaller loss than our method, this is in contrast to the case in the example in Fig. 1-1 where the 3th degree polynomial yielded significantly worse results. We believe that this occurred due the "un-noisy" data corresponding to a more "smooth" function, while in Fig. 1-1 the function was "less" smooth due to the data generation.

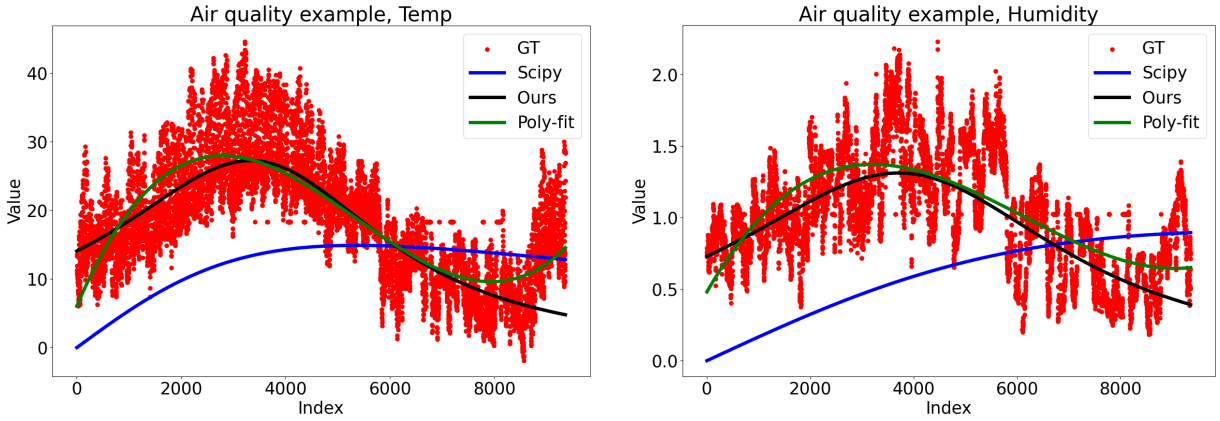

Figure 5-5: The n-signal corresponding to Temperature and Absolute Humidity properties of the dataset [68], marked as red points (Ground truth, GT), along with three fitted functions: (i) using the approximate rational function computed by `Our-FRFF`, (ii) using `Scipy.optimize.minimize`, and (iii) and a 3th degree polynomial computed using the `numpy.polyfit` function. For fair comparison, all three methods use 4 free parameters.

Repeating the experiments in 5.1 for this dataset. Repeating the test in 5.1 for this data set yields Fig. 5-6 and the following table. As previously in 5.1 the results are averaged across 100 tests and are accompanied by error bars that present the 25% and 75% percentiles.

Due to there being a relatively little change in the computational time over the features and sample sizes we show the mean results in the following table, which contains the times only for the Absolute Humidity feature in the dataset.

Method	Time for the Humidity feature (sec)
Our-RFF	0.1422
Our-FRFF	0.1172
Gradient	0.0434
L_∞Coreset	0.7476
RandomSample	0.0024
NearConvexCoreset	0.6247

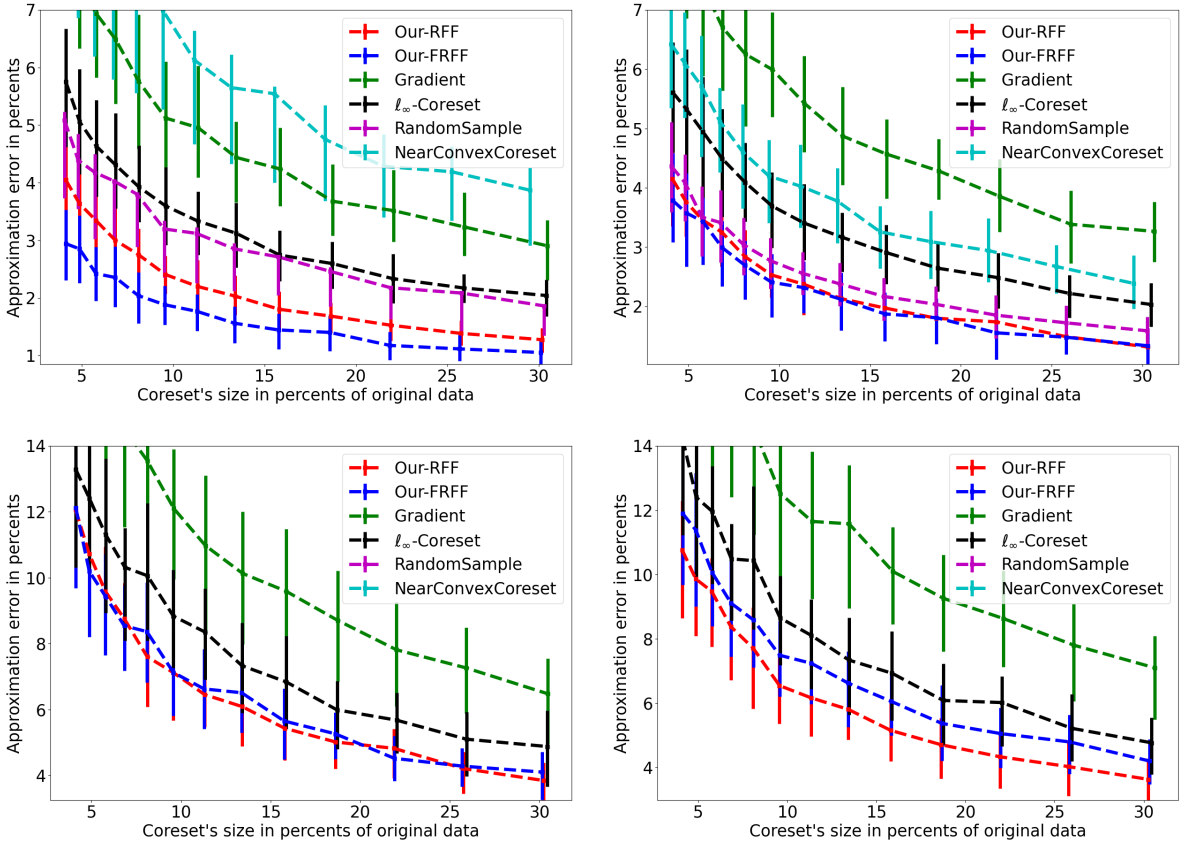

*Figure 5-6: Results for an experiment similar to the the experiment from Section 5.1, but with dataset [68]. The x-axis shows the compression ration, with respect to percents of the original data. The y-axis shows the approximation error of each compression scheme, for the properties Temperature (left column) and Absolute Humidity (right column). The upper and lower rows present Evaluations (i) and (ii) respectively. The error bars present the 25% and the 75% percentiles. Methods **RandomSample** and **NearConvexCoreset** produced very large errors and are thus in some cases were clipped.*

5.3 Discussion

5.3.1 Fig. 1-1

This figure demonstrates that rational function fitting is more suitable than polynomial fitting, for a data relatively normal dataset that is essentially is a set of samples from an exponential function. This also shows that computing any of those fitting functions either on top of the full data or our coreset produces similar results.

5.3.2 Section 5.1

In this section we demonstrated that our algorithms achieved the best accuracy consistently across the varying signal's length, where `Our-RFF` achieved the best performance and is followed by `Our-FRFF`. The main results are summarised in Fig. 5-1. While when considering Evaluation (i) `RandomSample`, that is essentially a random sample from the input, yielded a good quality, in Evaluation (ii) the values where so large that they where clipped. We, informally, believe that it accrued since the near optimal query considered in Evaluation (i) is not "very far" from any point in the generated data set and as such no point in the data is "very important" for the near optimal query, but as presented in Observation 22 this is not the case for all the queries or even for all the data sets (suppose a dataset where all the points lay on a query beside one point that has y-value approaching ∞, this single point would be "very important").

In this experiment we obtained significant speed improvement compared to all the other methods beside `RandomSample` and `Gradient`, where the later is based on SciPy's library [67] function `Scipy.optimize.minimize`, and it can be expected since a random sample would obviously be efficient and the function `Scipy.optimize.minimize` (at least for our parameters and to the best of our knowledge) used optimization methods from [45] that at some cases can yield very fast a local minimum (but not a global one). Hence, while `RandomSample` and `Gradient` had lower running time as observed in Fig. 5-1 they had significantly worse results.

Observe that while L_∞`Coreset` which uses the guaranteed approximation for max deviation from [51] might seem as a valid heuristic at first glance, our results in Fig. 5-1 demonstrated that

while it was the closest contestant to our methods in terms of quality, it came at the price of a significantly larger running time (it was the only time plot that was clipped). While it is plausible that it follows from our improper parameter tuning, we believe that it comes from the use linear programming solvers which to the best of our knowledge there are no solvers with a running time in $O\left(n^2\right)$; to the best of our knowledge, for the time of writing, the lowest bound is the one in [14].

5.3.3 Section 5.2

In the experiments we obtained very similar results as in the experiment in Section 5.1, which validates our observation above. We note that at some cases `Our-FRFF` achieved better quality than `Our-RFF`, where this mostly occurred for Evaluation (i). Informally we believe that this occurred since the approximation in `Our-FRFF` was very similar to the near optimal query considered, while the BI-criteria in `Our-RFF` might had lower loss but was farther from the query considered. Hence, this difference effected the difference between the results for Evaluation (i) and (ii), where for the later `Our-RFF` achieved almost consistently better results while in the later lose in many cases. Observe especially the PRES property in Section 5.2.1, where in Evaluation (i) we had `Our-RFF` only worse of equivalent results than `Our-FRFF`, but for Evaluation (ii) we had `Our-RFF` that gave significantly better results than `Our-FRFF`.

We note that for TEMP for the year 2013 in Fig. 5-4, where we used Evaluation (ii), and Fig. 5.1 for the Evaluation (ii) of the Temperature property we had equivalent quality between `Our-RFF` and `Our-FRFF`, where the later had better results in some instances.

5.3.3.1 Figures 5-2 and 5-6 :

In contrast to the example in Fig. 1-1 in those examples the polynomial fitting with the same number of parameters yielded slightly better results. This is a non surprising result that follows intuitively from considering that as rational functions might yield better fitting for some datasets (especially "non-smooth" [51]) for some datasets it is possible to have an opposite effect where polynomial fitting outperforms the rational functions fitting. We believe that this is a specific example of [69] that is related to the well known "No free lunch theorem" [70].

Nonetheless, even in those examples our approximation outperforms the Scipy's library [67] approximation via `Scipy.optimize.minimize` by a significant margin and gives only slightly worse results than the Numpy's library [27] polynomial fitting via `numpy.polyfit`.

Observe that this does not invalidate the real-world data experiments in Section 5, since while polynomial fitting yields lower loss we focused on the task of fitting rational functions that is a valid optimization problem on its own. Also we note that our rational function was obtained from `Our-FRFF` and it is plausible that the optimal rational function (with non constant denominator) would outperform the optimal fitting polynomial function with equal number of free parameters.

5.3.4 Discussion on the theoretical result

Another contribution of this thesis is the theoretical results. Our main result which is the basis of our tested methods is Theorem 42, which informally can be summarised as follows.

Given an n-signal P we can compress it to a sub-linear size in a quasi-linear time, and the compression with defined probability allows use to compute $\ell(P, q)$, for every query $q \in \left(\mathbb{R}^k\right)^2$, up to a multiplicative factor of $(1 \pm \varepsilon)$.

In our eyes another main contribution of this thesis is the very uncommon framework used in this work, whose overview is in Section 3.2. In particular we used the Merge-Reduce scheme presented in [9] to maintain a BI-CRITERIA tree, where it is usually presented to maintain on-line coresets. We also wish to mention the combination of the BI-CRITERIA approximations, where we essentially "trimmed" the approximation to chunks where the data is "well-behaved" and this allowed use to compute a "leaky-coreset" that fails for a bounded number of BI-CRITERIA for each query. this "leaky-coreset" is used in union with an exhaustive search over the "leaks" that are the BI-CRITERIA in the input where the coreset fails.

While those methods might be useful only the researchers in the coreset community, we hope that those novel methods might help build coresets for problems where there was a coreset only for a very specific case, which from a bottom up look can be seen as the focus of this work, i.e., we start with a coreset only for a very restricted case (both in query and form) and we build upon this to obtain an approximation and consecutively an ε-coreset without any such limitations.

Chapter 6

Conclusion and future work

This thesis provides a coreset construction that gets a time-series and returns a small coreset that approximates its sum of (fitting) distances to any rational functions of constant degree, up to a factor of $1 \pm \varepsilon$. The size of the coreset is sub-linear in n and quadratic in $1/\varepsilon$. Our main application is fitting to Auto-Regression model, whose generative functions are rational. While we focused on sum of errors (distances), we expect easy generalization to squared-distances, robust M-estimators and any other loss function that satisfies the triangle inequality, up to a constant factor, as in other coreset constructions [32]. We hope that the new suggested technique initializes a line of research that would enable sub-linear time algorithms with provable approximation for more sophisticated stochastic models such as: Hidden-Markov Models (HMM) (see [7, 39, 41, 50, 57, 71], and references therein), Baysian Networks (see [1, 41, 43, 56], and reference therein), Decision Markov Process [58], and Kalman's filter [41].

Bibliography

[1] Acar, U. A., Ihler, A., Mettu, R., and Sümer, O. Adaptive bayesian inference. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, 2007.

[2] Agarwal, P. K., Cormode, G., Huang, Z., Phillips, J. M., Wei, Z., and Yi, K. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):1–28, 2013.

[3] Anthony, M. and Bartlett, P. L. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

[4] Arthur, D. and Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[5] Asuncion, A. and Newman, D. UCI machine learning repository, 2007.

[6] Baker Jr, G. A. The theory and application of the padé approximant method. Technical report, Los Alamos Scientific Lab., Univ. of California, N. Mex., 1964.

[7] Basu, S., Choudhury, T., Clarkson, B., Pentland, A., et al. Learning human interactions with the influence model. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[8] Bentley, J. L. and Saxe, J. B. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[9] Braverman, V., Feldman, D., and Lang, H. New frameworks for offline and streaming coreset constructions. *CoRR*, abs/1612.00889, 2016.

[10] Bulirsch, R., Stoer, J., and Stoer, J. *Introduction to numerical analysis*, volume 3. Springer, 2002.

[11] Chen, S. Beijing Multi-Site Air-Quality Data. UCI Machine Learning Repository, 2019.

[12] Code. Open source code for all the algorithms presented in this paper, 2022. https://github.com/AnonRFF/Coreset-for-Rational-Functions.

[13] Cohen, M. B., Lee, Y. T., Musco, C., Musco, C., Peng, R., and Sidford, A. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190, 2015.

[14] Cohen, M. B., Lee, Y. T., and Song, Z. Solving linear programs in the current matrix multiplication time. In *Proceedings of the ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 938–942, 2019.

[15] Dasgupta, A., Drineas, P., Harb, B., Kumar, R., and Mahoney, M. W. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

[16] Epperson, J. F. On the runge example. *The American Mathematical Monthly*, 94(4):329–341, 1987.

[17] Eshragh, A., Roosta, F., Nazari, A., and Mahoney, M. W. Lsar: Efficient leverage score sampling algorithm for the analysis of big time series data. *Journal of Machine Learning Research*, 23(22):1–36, 2022.

[18] Feldman, D. Core-sets: Updated survey. *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 23–44, 2020.

[19] Feldman, D., Kfir, Z., and Wu, X. Coresets for gaussian mixture models of any shape. *CoRR*, abs/1906.04895, 2019.

[20] Feldman, D. and Langberg, M. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578, 2011.

[21] Feldman, D., Schmidt, M., and Sohler, C. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1434–1453. Society for Industrial and Applied Mathematics, 2013.

[22] Feldman, D., Sung, C., and Rus, D. The single pixel gps: Learning big data signals from tiny coresets. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 23–32, 2012.

[23] Feldman, D. and Tassa, T. More constraints, smaller coresets: Constrained matrix approximation of sparse big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 249–258, 2015.

[24] Ghosh, S., Rekatsinas, T., Mekaru, S. R., Nsoesie, E. O., Brownstein, J. S., Getoor, L., and Ramakrishnan, N. Forecasting rare disease outbreaks with spatio-temporal topic models. In *NIPS 2013 Workshop on Topic Models, Lake Tahoe, NV, USA*, 2013.

[25] Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, 2004.

[26] Har-Peled, S. and Sharir, M. Relative (p, ε)-approximations in geometry. 45(3):462–496, 2011.

[27] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[28] Higham, N. J. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics (SIAM), 2008.

[29] Horn, R. A. and Johnson, C. R. *Matrix Analysis*. UK: Cambridge University Press, 1999.

[30] Indyk, P., Mahabadi, S., Mahdian, M., and Mirrokni, V. S. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 100–108, 2014.

[31] Jubran, I., Cohn, D., and Feldman, D. Provable approximations for constrained ℓ_p regression. *CoRR*, abs/1902.10407, 2019.

[32] Jubran, I., Sanches Shayda, E. E., Newman, I. I., and Feldman, D. Coresets for decision trees of signals. In *Advances in Neural Information Processing Systems (NIPS)*, volume 34, pages 30352–30364, 2021.

[33] Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 34, 2021.

[34] Li, Y., Long, P. M., and Srinivasan, A. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001.

[35] Lu, H., Li, M.-J., He, T., Wang, S., Narayanan, V., and Chan, K. S. Robust coreset construction for distributed machine learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2400–2417, 2020.

[36] Lucic, M., Faulkner, M., Krause, A., and Feldman, D. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

[37] Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8305–8316, 2019.

[38] Marom, Y. and Feldman, D. k-means clustering of lines for big data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32, 2019.

[39] McCallum, A., Freitag, D., and Pereira, F. C. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000.

[40] Meinardus, G., Nürnberger, G., Sommer, M., and Strauß, H. Algorithms for piecewise polynomials and splines with free knots. *Mathematics of computation*, 53(187):235–247, 1989.

[41] Murphy, K. P. *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley, 2002.

[42] Neumaier, A. *Introduction to numerical analysis*. Cambridge University Press, 2001.

[43] Nikolova, O., Zola, J., and Aluru, S. A parallel algorithm for exact structure learning of bayesian networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[44] NIST/SEMATECH. e-handbook of statistical methods. `https://www.itl.nist.gov/div898/handbook/pmd/section6/pmd642.htm`, 2021.

[45] Nocedal, J. and Wright, S. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[46] Noda, M.-T. and Miyahiro, E. A hybrid approach for the integration of a rational function. *Journal of computational and applied mathematics*, 40(3):259–268, 1992.

[47] Northrop, R. B. *Signals and systems analysis in biomedical engineering*. CRC press, 2016.

[48] Nürnberger, G. *Approximation by spline functions*, volume 1. Springer, 1989.

[49] Nyquist, H. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

[50] Park, H., Yun, S., Park, S., Kim, J., and Yoo, C. Phoneme classification using constrained variational gaussian process dynamical system. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 25, 2012.

[51] Peiris, V., Sharon, N., Sukhorukova, N., and Ugon, J. Generalised rational approximation and its application to improve deep learning classifiers. *Applied Mathematics and Computation*, 389(C), 2021.

[52] Petersen, K. B. and Pedersen, M. S. The matrix cookbook. Version 20121115.

[53] Petrushev, P. P. and Popov, V. A. *Rational approximation of real functions*. Number 28. Cambridge University Press, 2011.

[54] Pizzo, A., Zappone, A., and Sanguinetti, L. Solving fractional polynomial problems by polynomial optimization theory. *IEEE Signal Processing Letters*, 25(10):1540–1544, 2018.

[55] Rosman, G., Volkov, M., Feldman, D., Fisher III, J. W., and Rus, D. Coresets for k-segmentation of streaming data. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 27, 2014.

[56] Rudzicz, F. Learning mixed acoustic/articulatory models for disabled speech. In *Advances in Neural Information Processing Systems (NIPS)*, pages 70–78, 2010.

[57] Sassi, I., Anter, S., and Bekkhoucha, A. A new improved baum-welch algorithm for unsupervised learning for continuous-time hmm using spark. *International Journal of Intelligent Engineering and Systems (IJISAE)*, 13(1):214–226, 2020.

[58] Shanahan, J. and den Poel, D. Determining optimal advertisement frequency capping policy via markov decision processes to maximize click through rates. In *Proceedings of NIPS Workshop: Machine Learning in Online Advertising*, pages 39–45, 2010.

[59] Shani, G. and Brafman, R. Resolving perceptual aliasing in the presence of noisy sensors. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, 2004.

[60] Sigler, L. *Fibonacci's Liber Abaci: a translation into modern English of Leonardo Pisano's book of calculation*. Springer Science & Business Media, 2003.

[61] Sukhorukova, N. Uniform approximation by the highest defect continuous polynomial splines: necessary and sufficient optimality conditions and their generalisations. *Journal of optimization theory and applications*, 147(2):378–394, 2010.

[62] Sukhorukova, N. and Ugon, J. Characterisation theorem for best polynomial spline approximation with free knots. *Transactions of the American Mathematical Society*, 369(9):6389–6405, 2017.

[63] Thomas, J. M. Sturm's theorem for multiple roots. *National Mathematics Magazine*, 15(8):391–394, 1941.

[64] Trefethen, L. N. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[65] Tukan, M., Maalouf, A., and Feldman, D. Coresets for near-convex functions. In *Advances in Neural Information Processing Systems (NIPS)*, volume 33, pages 997–1009, 2020.

[66] Vapnik, V. Principles of risk minimization for learning theory. In *Advances in neural information processing systems (NIPS)*, volume 4, 1991.

[67] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[68] Vito, S. Air Quality. UCI Machine Learning Repository, 2016.

[69] Wolpert, D. H. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390, 10 1996.

[70] Wolpert, D. and Macready, W. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[71] Yu, D., Deng, L., and Dahl, G. Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[72] Yuan, Q. Topics in generating functions. *Massachusetts Institute of Technology: Department of Mathematics*, pages 22–23, 2009.