Accurate Coresets

Student Name: Sanskar Ranjan
Roll Number: 2021096

BTP report submitted in partial fulfilment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering

on May 9, 2024

BTP Track: Research

BTP Advisor
Dr Supratim Shit

Indraprastha Institute of Information Technology
New Delhi

Student Declaration

I hereby declare that the work presented in the report entitled **"Accurate Coresets"** submitted by me for the partial fulfilment of the requirements for the degree of *B.Tech. in Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under the guidance of **Dr Supratim Shit** . Due acknowledgements have been given in the report for all material used. This work has not been submitted elsewhere for the reward of any other degree.

Sanskar Ranjan **Place & Date:** May 9, 2024

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dr Supratim Shit **Place & Date:** May 9, 2024

Abstract

This paper examines the application of accurate coresets in regularized regression, with a specific focus on the relationship between lambda values and coreset sizes. Accurate coresets are essential for maintaining data properties without loss, making them valuable in scenarios like regularized regression, where preserving data characteristics is critical. By varying lambda values, we investigate how coreset sizes are affected while ensuring the loss remains constant. Our findings offer insights into selecting lambda values for efficient model approximation, contributing to the field of data summarization and optimization.

Keywords: (Accurate coresets, Regularized regression, Caratheodory's Algorithm, Lambda constant)

Acknowledgment

I am expressing my sincere gratitude to Dr. Supratim Shit for his help and support in my research. His expertise and insight were invaluable in helping me to develop my research project. His knowledge, guidance, and enthusiasm for the project were instrumental in making the project a success. I am deeply grateful for his guidance and support throughout the process. I am thankful for his dedication, commitment, and willingness to share his experience and knowledge. His inputs were invaluable, and I cannot thank him enough for his help.

Contents

Chapter 1

Introduction

1.1 Coresets

In data science and optimization, coresets act as a powerful tool for approximating different data properties. Originally, coresets are designed to preserve the structure and characteristics of data. Coresets can be divided into three categories depending on their accuracies and the guarantee with which they obtain these accuracies.

1.1.1 ϵ-Approximation with δ-Failure

This is the most general type of coreset. It guarantees that with high probability $(1 - \delta)$, any computation performed on the coreset C will produce an answer within a factor of ϵ (epsilon) of the true answer obtained using the original data set D. Mathematically, for a given function f:

$$P\left[|f(C) - f(D)| \leq \varepsilon \cdot |f(D)|\right] \geq 1 - \delta \tag{1.1}$$

The trade-off here is between the values ε & δ and the size of the accurate coreset . Smaller ε & δ an leads to a more accurate coreset with lesser failure, but requires a larger coreset size. Conversely, a higher ε & δ allows for a smaller coreset but increases the chance of getting an inaccurate result or increase the inaccuracy of the coreset.

1.1.2 ϵ-Approximation with 0 Failure

This type of coreset guarantees the same level of approximation accuracy (ε) as the first type, but it ensures that the approximation will be accurate with certainty $(\delta = 0)$. Mathematically, for a given function f:

$$|f(C) - f(D)| \leq \varepsilon \cdot |f(D)| \tag{1.2}$$

This stronger guarantee typically comes at a cost. Creating a coreset with 0 failure probability might require a larger size compared to the first type with a non-zero δ. Additionally, constructing such coresets might be computationally more expensive.

1.1.3 Accurate Coreset (0-Approximation with 0 Failure)

This is the ideal scenario where the coreset perfectly captures the properties of the original data set. It achieves both perfect accuracy ($\varepsilon = 0$) and guaranteed success ($\delta = 0$).Mathematically, for a given dataset D we can find a coreset C such that for the given loss function f_{loss}:

$$f_{loss}(C) = f_{loss}(D) \tag{1.3}$$

Figure 1.1 is an example showing how an accurate coreset helps in summarising the dataset to give the same loss.

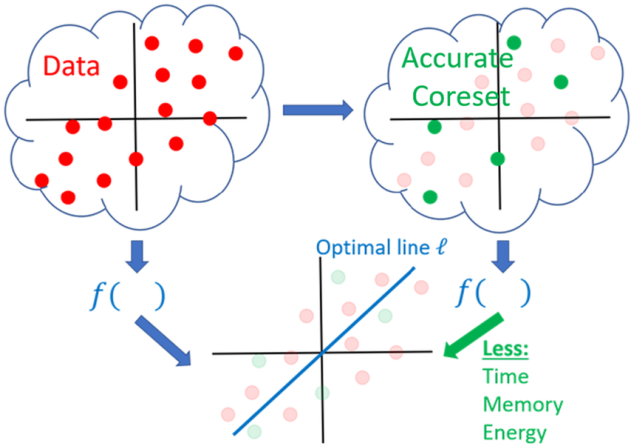

Figure 1.1: A representation of accurate coreset

This accurate coreset can further be of two types:

Subset Coresets

A subset coreset $\mathcal{C} \subseteq \mathcal{D}$ is a weighted subset of the original dataset $\mathcal{D}$, where each point $p \in \mathcal{C}$ is assigned a weight w_p such that the weighted loss function $\mathcal{F}$ computed over $\mathcal{C}$ is equal to the weighted loss over $\mathcal{D}$:

$$\sum_{p \in \mathcal{C}} (w_p \mathcal{F}_{\text{loss}}(p)) = \sum_{p \in \mathcal{D}} (w_p \mathcal{F}_{\text{loss}}(p))$$

For example, in the 1-center problem, the subset coreset contains only the two extreme points $p_{\min}$ and $p_{\max}$ of $\mathcal{D}$, representing the minimum and maximum extent along a line segment.

Non-Subset Coresets

A non-subset coreset $\mathcal{C}$ is not restricted to points from the original dataset $\mathcal{D}$. It can contain synthetic or summarized information derived from $\mathcal{D}$. Two examples are:

1. For the vector sum problem,the coreset can be define as the set C= c such that $u(c)(c-x) = \sum_{p\in\mathcal{D}} w(p)(p-x)$, where $u(c)$ is the weight assigned to c and x is a reference point.
2. For the 1-mean problem, $\mathcal{C}$ might contain a tuple $\left(\sum_{p\in\mathcal{D}} w_p, \sum_{p\in\mathcal{D}} w_p p, \sum_{p\in\mathcal{D}} w_p \|p\|^2\right)$, which encodes the sum of weights, weighted sum of points, and weighted sum of squared norms from $\mathcal{D}$, respectively.

Non-subset coresets provide a compact representation of $\mathcal{D}$ by encoding relevant summary statistics or synthetic points that capture the essential information needed for the specific problem at hand.

1.2 Regularized Regression and Lambda Values

Accurate coresets can significantly reduce training costs for regularized regression models used in predictive modeling to prevent overfitting.

Normally regularization parameter $\lambda \geq 0$ controls the trade-off: larger λ favors simpler models (higher bias), while smaller λ allows more complex models (higher variance).

The same lambda value in case of Accurate coreset can help vary the size of our coreset. This change in the sizes of the coreset is something we will be looking into as a part of our project.

Chapter 2

Motivation

2.1 Motivation for Accurate Coresets

In the context of machine learning and optimization problems, accurate coresets have emerged as a powerful tool for reducing computational complexity while maintaining the same loss as the original dataset. The motivation for seeking smaller coreset sizes stems from the potential computational benefits and improved efficiency.

2.1.1 Impact of Regularization on Coreset Size

As the regularization parameter λ increases, the regularization term $\lambda\|x\|_2^2$ in the objective function becomes more dominant, favoring solutions with smaller norms. Consequently, the optimal solution tends to move closer to the origin, and the representative points in the coreset become more concentrated near the origin. When the regularization is strong enough, the optimal solution may be close to the origin, and the coreset can potentially consist of a single point (the origin) or a few points near the origin. This reduction in the number of points in the coreset translates to a smaller coreset size.

2.1.2 Computational Benefits of Smaller Coresets

By including fewer data points in the coreset, we can reduce the computational complexity of various operations involved in the optimization problem, such as distance computations, matrix operations, and overall computational overhead. This is particularly important in high-dimensional settings or when dealing with large datasets, where the computational cost can become prohibitive. Smaller coresets can lead to significant computational savings, making it possible to tackle larger and more complex problems with limited computational resources. Additionally, efficient storage and transmission of data are facilitated, as only the representative points need to be stored or transmitted, instead of the entire dataset. This can be particularly useful in distributed computing environments or when working with resource-constrained devices.

2.1.3 Balancing Regularization and Computational Complexity

The motivation for seeking smaller coreset sizes stems from the potential computational benefits, improved efficiency, and reduced storage and transmission requirements. By understanding the relationship between the regularization parameter λ and the coreset size, we can strategically choose λ values that strike a balance between the desired level of regularization and the computational complexity of the problem. Larger values of λ tend to result in smaller coresets, but they may also introduce excessive regularization, potentially leading to underfitting or biased solutions. On the other hand, smaller values of λ may yield larger coresets, increasing computational complexity while reducing the impact of regularization. Therefore, it is crucial to carefully select the regularization parameter λ to achieve the desired trade-off between model complexity, computational efficiency, and predictive performance, leveraging the potential benefits of smaller coreset sizes when appropriate.

Chapter 3

Literature Review

3.1 Introduction to Coresets: Accurate Coresets

As described in the paper [2] accurate coresets are a modern data summarization technique that approximates the original data in a provable manner without compromising accuracy. They are designed to compute the model that minimizes the objective function on a small coreset instead of the original data, while ensuring that the solution remains within a small multiplicative factor of the true solution . There are two main types of accurate coresets: subset coresets and non-subset coresets. Subset coresets are constructed by selecting a weighted subset of points from the original dataset such that the weighted loss function computed over the coreset is equal to the weighted loss over the original data. Non-subset coresets, on the other hand, are not restricted to points from the original dataset and can contain synthetic or summarized information derived from the data. These coresets are often constructed using techniques like Caratheodory's theorem, matrix basis, and singular value decomposition (SVD)

3.1.1 Example: The 1-Center Problem (Subset Coreset)

The 1-center problem is a fundamental problem in computer science and has numerous applications in various fields. Given a set of points $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ in $\mathbb{R}^d$, the goal is to find the point $c \in \mathbb{R}^d$ that minimizes the maximum distance to any point in P, i.e., $\min_{c \in \mathbb{R}^d} \max_{p \in P} \|p - c\|$. This problem can be solved using a subset coreset by constructing a weighted subset of the original data points such that the sum of their weights is equal to the sum of weights of the original points . As illustrated in Figure 3.1, the 1-center problem aims to find the point c (the center) that minimizes the maximum distance to any point in the set P. A subset coreset for this problem can be constructed by selecting the two extreme points $p_{\min}$ and $p_{\max}$ that represent the minimum and maximum extents along the line segment containing all points in P. The weights of $p_{\min}$ and $p_{\max}$ are chosen such that their weighted sum is equal to the sum of weights of all points in P. This coreset captures the essential information required to solve the 1-center problem accurately.

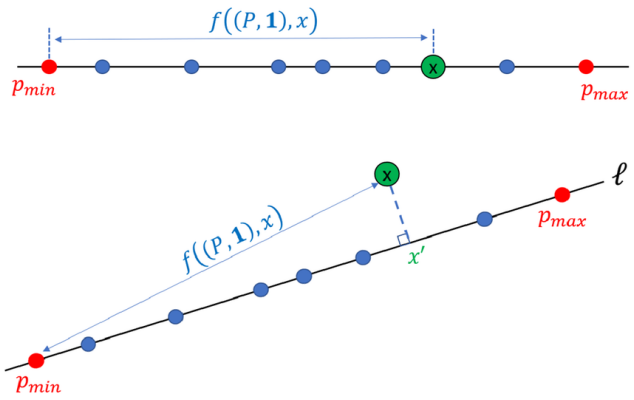

Figure 3.1: Visualization of the 1-center problem

3.1.2 Importance of Caratheodory's Theorem and Linear Algebra Tools

Caratheodory's theorem and other linear algebra tools like QR decomposition play a crucial role in constructing accurate non-subset coresets. Caratheodory's theorem provides a way to represent a point in a high-dimensional space as a convex combination of a small number of points. This theorem can be used to construct non-subset coresets by representing the original data as a convex combination of a few synthetic points, which form the coreset. QR decomposition and other matrix factorization techniques can be used to find low-dimensional representations of high-dimensional data, which can then be used to construct non-subset coresets. These techniques help identify the essential information in the data and capture it in a compact form, facilitating the construction of coresets that accurately represent the original data while being computationally efficient.

3.2 Sharper Bounds for Regularized Data Fitting

The paper [1] focuses on constructing coresets for data fitting problems such as ridge regression and ridge low-rank approximation using advanced sampling techniques and regularization methods. The authors leverage tools like the Subsampled Randomized Hadamard Transform (SRHT) and leverage scores to build coresets that provide approximate solutions while minimizing computational complexity.

3.2.1 Importance of Lambda Values

The regularization parameter λ plays a crucial role in determining the size and quality of the coresets. The paper introduces the notion of a "large λ" in Definition 13: Say that λ is large for a matrix A with largest singular value σ_1, and error parameter ϵ, if $\lambda/\sigma_1^2 \geq 1/\epsilon$. If λ is large, then $x = 0$ is a good approximate solution, and as long as we include a check that a proposed solution is no worse than $x = 0$, we can assume that λ is not large.

Larger λ values lead to stronger regularization, affecting the trade-off between fitting the data and controlling overfitting. The paper analyzes how different λ values influence the size and structure

of the coresets, ultimately impacting the accuracy of the approximated solutions.

3.2.2 Applications to Accurate Coresets

In the context of accurate coresets, where the goal is to obtain exact solutions, the concept of a "large λ" can be useful. If λ is large enough, the optimal solution may be close to zero, and a simple check can determine if the trivial solution ($x = 0$) is sufficient. This can potentially lead to smaller coresets or even eliminate the need for constructing coresets altogether. In other words very large values of lambda can lead to a coreset of zero size. However, for accurate coresets, the challenge lies in identifying the appropriate λ values that strike a balance between fitting the data accurately and controlling the complexity of the model. The paper's insights on the effects of λ on the size and effectiveness of coresets can be extended to the accurate coreset setting, potentially leading to more efficient and accurate solutions for data fitting problems.

Chapter 4

Problem Statement

4.1 Problem Statement

Consider the regularized regression problem with the following objective function:

$$\ell_{\text{loss}}(x) = |Ax - b|_2^2 + \lambda |x|_2^2 \tag{4.1}$$

where $A \in \mathbb{R}^{m \times d}$ is the data matrix, $b \in \mathbb{R}^m$ is the target vector, $x \in \mathbb{R}^d$ is the regression coefficient vector, and $\lambda \geq 0$ is the regularization parameter. The goal is to find an accurate coreset $\mathcal{C}_\lambda$ that conserves the regularized loss function $\ell_{\text{loss}}(x)$ while reducing computational complexity. The accurate coreset $\mathcal{C}_\lambda$ is a subset of the data points and regularization points. The size of the accurate coreset, denoted by $\text{size}(\mathcal{C}_\lambda)$, is upper bounded by $(d+1)^2+1$. However, as the value of λ increases, more regularization points are included in $\mathcal{C}_\lambda$, reducing its effective size. Specifically, when λ is sufficiently large, all d regularization points are included in $\mathcal{C}_\lambda$, and the coreset size becomes $(d+1)^2+1-d$. The objective is to analyze the relationship between the regularization parameter λ and the size of the accurate coreset $\text{size}(\mathcal{C}_\lambda)$ in a generalized manner. Additionally, we aim to determine the minimum value of λ, denoted by $\lambda\min(A, b, d)$, at which all d regularization points are included in $\mathcal{C}_\lambda$. Formally, we seek:

$$\lambda_{\min}(A, b, d) = \min_{\lambda \geq 0} \left\{ \lambda : \text{size}(\mathcal{C}_\lambda) = (d+1)^2 + 1 - d \right\} \tag{4.2}$$

By analyzing the behavior of $\text{size}(\mathcal{C}_\lambda)$ as a function of λ, and estimating $\lambda\min(A, b, d)$, we can gain insights into the trade-off between the regularization strength and the computational complexity of the regularized regression problem.

We also aim to prove or disprove the hypothesis of linear dependency of $\lambda_{\min}$ on the size of coreset hypothesized in Section 5.4.

Chapter 5

Methodology and Regularization Experiments

5.1 Problem Definition

We are trying to find an accurate coreset such that the regression loss defined by the following function is conserved:

$$\ell_{\text{loss}}(x) = \|Ax - b\|_2^2 + \lambda\|x\|_2^2$$

To reach that goal, we first write this loss in the form:

$$\text{loss} = \left\| \begin{bmatrix} A & b \\ \sqrt{\lambda} I_d & 0 \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} \right\|^2$$

This problem can be simplified as finding the matrix 2-norm accurate coreset from the paper [2] with equal weights for all points in the matrix.

The size of the accurate coreset is $(d+1)^2 + 1$. This set can contain points from both the dataset points and the regularization points. The final size of the accurate coreset is decided by how many points from the dataset are included in this set. We can tune this final size using the values of λ. As we keep increasing the value of λ, more and more regularization points are included in the final coreset until all d points of regularization are refined out in the final coreset, making the final size of our coreset $(d+1)^2 + 1 - d$.

To understand the trend of how the size of this final coreset changes with λ, we conducted experiments with some toy examples.

5.2 Stacked Identity Case

We experimented with a particular type of matrix A, which is a stacked identity matrix of the form:

$$A = \begin{bmatrix} I_d & 0 \\ I_d & 0 \\ \vdots & \vdots \\ I_d & 0 \\ \sqrt{\lambda} I_d & 0 \end{bmatrix}$$

where I_d is the $d \times d$ identity matrix, and A has k stacked copies of I_d with one $\sqrt{\lambda} I_d$ stacked at the end.

To find an accurate coreset on this matrix, we apply the same method as in the paper [2]. The method constructs a weighted subset C of the input points P such that the weighted sum of the matrices formed by the outer products of points in C equals the weighted sum of the matrices formed by the outer products of points in P. Mathematically, this can be expressed as:

$$\sum_{p \in P} w(p) pp^\top = \sum_{p \in C} u(p) pp^\top$$

where $w(p)$ and $u(p)$ are the weights assigned to points $p \in P$ and $p \in C$, respectively.

In our given case, $w(p) = \frac{1}{d \cdot k + d}$ as each point is assigned equal weight.

We then form a matrix M with flattened out $pp^\top$ matrices, which gives us the final matrix.

In this ideal case, we end up with a sparsified matrix M with just d non-zero columns. The Caratheodory's algorithm on this dataset gives us a final output with $d + 1$ rows.

We experimented with varying values of k (1 to 200), $d = 3$, and λ (1 to 100, where all regularized points are in the coreset, taking values like 1.5, 2, 2.5, etc.), and tried to understand the relationship between the λ value and the number of regularized data points in the coreset.

We plotted a scatter plot showing how many regularization points were found in the coreset against the size parameter of the matrix (k) and the regularization parameter (λ). The following were the results:

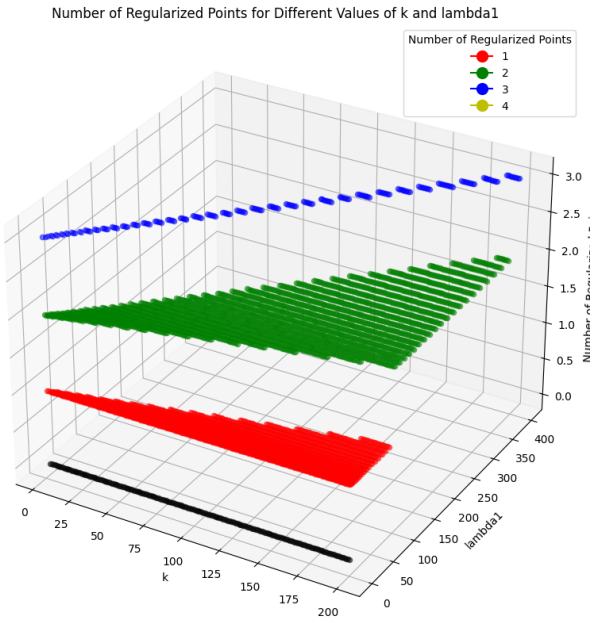

Figure 5.1: Scatter plot for the stacked identity case

The following graph shows the values of λ at which all three regularized points were included in the final accurate coreset.

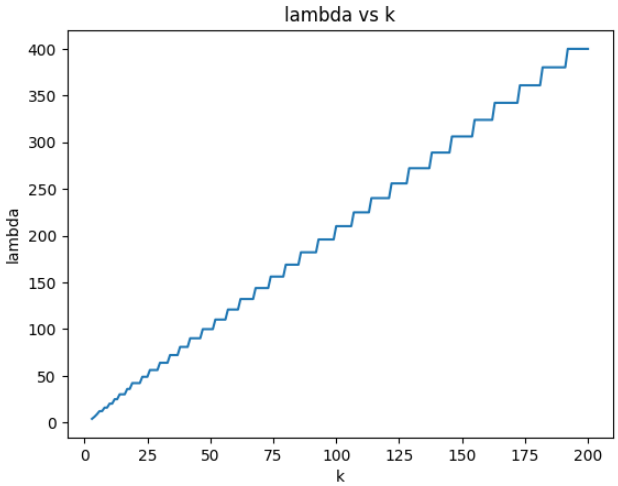

Figure 5.2: Values of λ for all regularized points to be in the coreset for the stacked identity case

Based on Figure 5.2, a hypothesis can be made that in the case of stacked identity matrices, the value of λ at which the coreset starts to include all d points from the regularization set (let's denote it as λ_d) is linearly dependent on the size of the dataset, or the number of stacked identity matrices in this case.

5.3 Random Value Matrix

The same experiment was repeated for a matrix A with n random data points of dimension $d+1$ with the following form (random values are from 1 to 9 for simplicity):

$$A = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0d} & b_0 \\ a_{10} & a_{11} & \cdots & a_{1d} & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nd} & b_n \\ \sqrt{\lambda} I_d & \cdots & \cdots & \cdots & 0 \end{bmatrix}$$

With values of n varying from 1 to 300 and λ increasing by 0.5 until we get all regularized points in the coreset, we obtained the following scatter plot and graph for λ versus n vs no of regularization points.

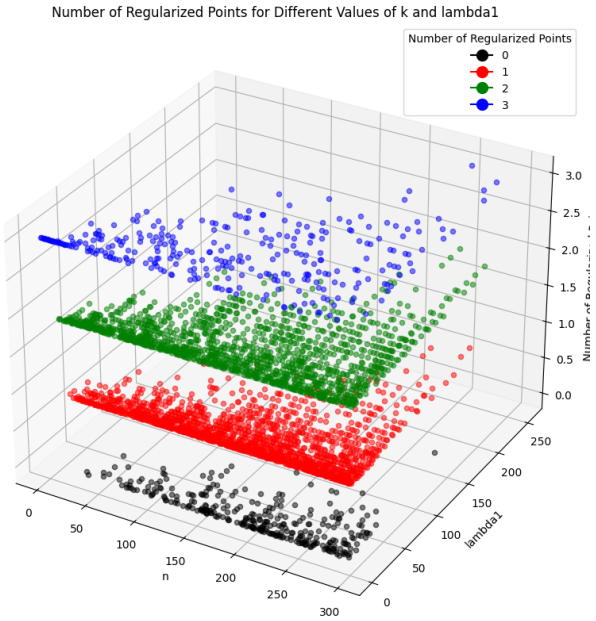

Figure 5.3: Scatter plot for the random value matrix case

After graphing out the values of lambdas where we get all the regularization points in the coreset, we smoothed out the curve by taking the moving average with a window size=5 to get to this graph.

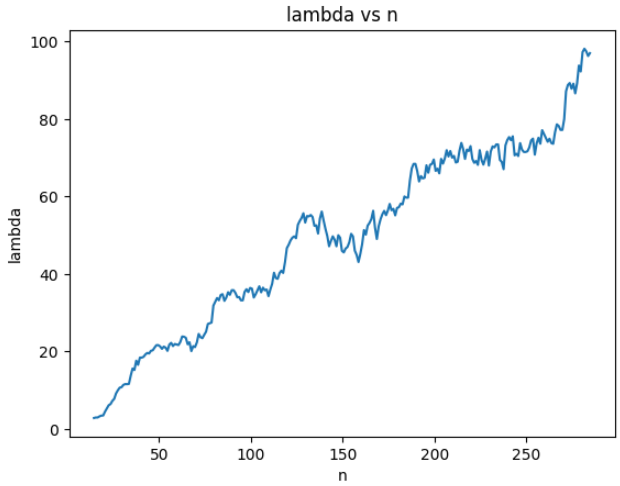

Figure 5.4: λ vs. n graph for the matrix with random values

This graph, gives a similar result to the one in the case of stacked identity matrices.

Using both of these results, we can hypothesize that the λ value required to get all the regularization points in the accurate coreset is linearly dependent on the value n, which is the total number of points in the dataset.

5.4 Summary and Hypothesis Formation

In summary, our experiments with the stacked identity matrix and the random value matrix suggest that the regularization parameter λ plays a crucial role in determining the number of regularization points included in the final accurate coreset. Specifically, we observe a linear relationship between λ and the dataset size, where larger values of λ tend to include more regularization points in the coreset. This finding can help guide the selection of appropriate λ values to achieve desired coreset sizes and regularization levels in ridge regression problems.

Chapter 6

Caratheodory's Theorem and Algorithm

6.1 Caratheodory's Theorem

Caratheodory's theorem states that if a point z lies in the convex hull of a set $P \subseteq \mathbb{R}^d$, then z can be expressed as a convex combination of at most $d+1$ points in P. Equivalently, z lies in some d-dimensional simplex with vertices in P.

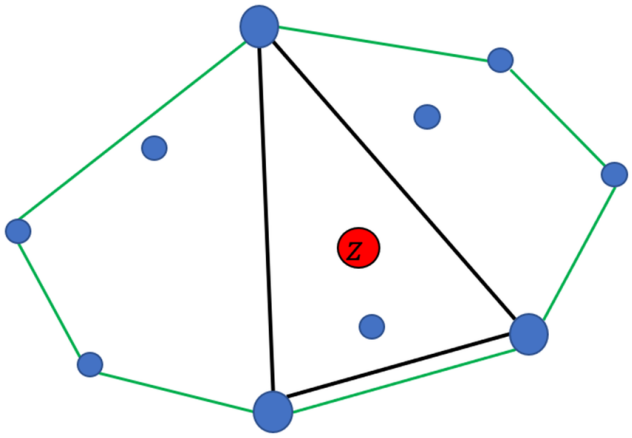

Figure 6.1: Caratheodory's visualization

6.2 Caratheodory's Algorithm

Caratheodory's algorithm is a computational method to find a subset $C \subseteq P$ of at most $d+1$ points such that the weighted sum of points in C equals the weighted sum of points in P:

$$\sum_{p \in P} w(p)p = \sum_{p \in C} u(p)p \tag{6.1}$$

The Caratheodory set also conserves the sum of weights, which are normalized to sum to 1.

The following is the classic algorithm for Caratheodory's theorem:

Algorithm 1 Caratheodory's Algorithm

Require: A weighted set (P, w) of n points in $\mathbb{R}^d$, where $P = \{p_1, \ldots, p_n\}$, $w : P \to [0, 1]$, and $\sum_{p \in P} w(p) = 1$

Ensure: A weighted set (S, u) computed in $O(n^2 d^2)$ time such that $S \subseteq P$, $|S| \leq d + 1$, $u : S \to [0, 1]$, $\sum_{s \in S} u(s) = 1$, and $\sum_{s \in S} u(s)s = \sum_{p \in P} w(p)p$

if $n \leq d + 1$ **then**
  return (P, w)
end if
for $i = 2, \ldots, n$ **do**
  $a_i \leftarrow p_i - p_1$ ▷ p_i is the i-th point of P
end for
$A \leftarrow (a_2 | \cdots | a_n)$ ▷ $A \in \mathbb{R}^{d \times (n-1)}$
Compute $v = (v_2, \ldots, v_n)^\top \neq 0$ such that $Av = 0$
$v_1 \leftarrow -\sum_{i=2}^{n} v_i$
$\alpha \leftarrow \min \left\{ \frac{w(p_i)}{v_i} \mid i \in \{1, \ldots, n\} \text{ and } v_i > 0 \right\}$
for $i = 1, \ldots, n$ such that $u(p_i) > 0$ **do**
  $u(p_i) \leftarrow w(p_i) - \alpha v_i$
end for
$S \leftarrow \{p_i \mid i \in \{1, \ldots, n\} \text{ and } u(p_i) > 0\}$
if $|S| > d + 1$ **then**
  $(S, u) \leftarrow$ CARATHEODORY(S, u) ▷ Recursive call to reduce S
end if
return (S, u)

6.3 Faster Caratheodory's Algorithms

We experimented with different and faster algorithms for calculating the Caratheodory set. Inspired by the k-partition algorithm from the paper [2], we tried an algorithm that utilizes different k values for splitting the dataset.

The basic idea is as follows:

1. While the size of the dataset $|P|$ is greater than $d + 1$:

   2. Divide the dataset into k_0 partitions.

   3. For each partition, normalize the weights, find the Caratheodory set, and then restore the original weights.

   4. Join the Caratheodory sets from each partition.

5. Divide the value of k by a ratio r.

6. Repeat the process until you get a set of d+1 points.

The algorithm, called Variable K-partition Caratheodory's Algorithm, is presented below:

Algorithm 2 Variable K-partition Caratheodory's Algorithm

Require: A set $P = \{p_1, \ldots, p_n\}$ of n points in $\mathbb{R}^d$

A weight function $w : P \to [0, \infty)$ such that $\sum_{p \in P} w(p) = 1$

Initial partition size $k_0 \in \{1, \ldots, n\}$

Partition size reduction ratio by dividing by $r > 1$

Ensure: A Caratheodory set (C, ω) of (P, w)

$P \leftarrow P \setminus \{p \in P \mid w(p) = 0\}$ ▷ Remove points with zero weight

$k \leftarrow k_0$

while $|P| > d + 1$ **do**

  $\{P_1, \ldots, P_k\} \leftarrow$ Partition P into k disjoint subsets (clusters), each contains $\approx \lceil n/k \rceil$ points

  for $i = 1, \ldots, k$ **do**

    $\mu_i \leftarrow \frac{1}{\sum_{p \in P_i} w(p)} \sum_{p \in P_i} w(p)p$ ▷ Weighted mean of P_i

    $\rho_i \leftarrow \sum_{p \in P_i} w(p)$ ▷ Total weight of P_i

  end for

  $(\tilde{\mu}, \tilde{\rho}) \leftarrow$ CARATHEODORY$(\{\mu_1, \ldots, \mu_k\}, \rho)$ ▷ See Algorithm 1

  $C \leftarrow \bigcup_{\mu_i \in \tilde{\mu}} P_i$ ▷ Union of clusters whose representatives were chosen

  for $\mu_i \in \tilde{\mu}$ and $p \in P_i$ **do**

    $\omega(p) \leftarrow \frac{\tilde{\rho}(\mu_i) \cdot w(p)}{\sum_{q \in P_i} w(q)}$ ▷ Assign weight for each point in C

  end for

  $P \leftarrow C$, $w \leftarrow \omega$ ▷ Update P and w for the next iteration

  $k \leftarrow \lfloor k/r \rfloor$ ▷ Reduce partition size for the next iteration

end while

$(C, \omega) \leftarrow$ CARATHEODORY(P, w) ▷ Apply Caratheodory's algorithm on the remaining small set

return (C, ω)

6.4 Time Complexity Analysis and Comparison

The time complexity of the Fast Caratheodory Set algorithm can be optimized by analyzing the following function:

$$O(r, k_1) = \frac{n^2 d^2}{k_1} + k_1 d^2 (d+1)^2 \left(r + 1 + \frac{1}{r} + \ldots \right) \tag{6.2}$$

where n is the number of input points, d is the dimension, r is the partition size reduction ratio, and k_1 is the partition size.

The optimal value of k_1 that minimizes $O(r, k_1)$ is given by:

$$k_1^* = \left\lfloor \frac{n\sqrt{r-1}}{r(d+1)} \right\rfloor \tag{6.3}$$

Substituting k_1^* into $O(r, k_1)$, we get the optimal time complexity:

$$O\left(\frac{nd^3r}{\sqrt{r-1}}\right) \approx O(nd^3) \tag{6.4}$$

This algorithm performs better than the original slow Caratheodory's theorem, which has a time complexity of $O(n^2d^2)$, and the fixed k-partition Caratheodory's algorithm. However, its time complexity is the same as the streaming point algorithm mentioned in the paper [2].

There are various other algorithms that we looked into with some different time complexities as functions of d and n, but most of them were not significantly enough or had worse complexity than $O(nd^3)$.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The main goal of this research was to study the relationship between the regularization parameter λ and the size of the accurate coreset in ridge regression problems. Through experiments with stacked identity matrices and random value matrices, we observed a linear dependence between λ and the dataset size. Larger values of λ tend to include more regularization points in the coreset, ultimately leading to a smaller coreset size compared to the state-of-the-art result when all regularization points are included. This finding provides insights into selecting appropriate λ values to achieve desired coreset sizes and regularization levels, potentially improving computational efficiency in ridge regression tasks.

7.2 Future Work

A natural extension of this research would be to explore the construction of accurate coresets for low-rank matrix approximation problems. Low-rank approximation techniques are widely used in various domains, such as dimensionality reduction, compression, and recommendation systems. Developing efficient coresets for these problems could lead to significant computational savings, particularly in large-scale datasets. Future work could investigate theoretical guarantees and practical algorithms for constructing accurate coresets for low-rank approximation, leveraging the insights gained from the ridge regression analysis.

Bibliography

[1] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. *Sharper Bounds for Regularized Data Fitting*. 2017. arXiv: 1611.03225 [cs.DS].

[2] Ibrahim Jubran, Alaa Maalouf, and Dan Feldman. *Introduction to Coresets: Accurate Coresets*. 2019. arXiv: 1910.08707 [cs.LG].