On coreset construction for K-means clustering of flats and hyperplanes

Abhisek Mukherjee

Advisors: Dr. Arijit Ghosh & Dr. Arijit Bishnu

Indian Statistical Institute, Kolkata

Roll No. CS1830

June 2020

For the partial requirement of the Master of Technology degree in Computer Science

Dedicated to all those who walked for their homes, but never reached...

Abstract

Coreset is an important tool to effectively extract information from large amount of data by sampling only a few elements from it, without any substantial loss of the actual information. An ϵ-coreset is defined as a weighted set C obtained from an universe X, so that for any solution set Q for a problem (referred to as a *query* in coreset literature), $|\text{Cost}(X, Q) - \text{Cost}(C, Q)| \leq \epsilon \cdot \text{Cost}(X, Q)$.
Our work is an attempt to generalize the solution provided in the paper "k-Means Clustering of Lines for Big Data" (Marom and Feldman, NIPS, 2019), and explore if it is possible to extend to k-flats in $\mathbb{R}^d$ as well. Following the approach used in the paper mentioned, we will attempt at building a deterministic algorithm to compute an ϵ-coreset whose size is near logarithmic of the input size for a j-dimensional affine subspace in $\mathbb{R}^d$.

Acknowledgements

I would like to thank Dr. Arijit Ghosh and Dr. Arijit Bishnu for all the encouragement that they provided. Working with them was never uncomfortable, and they directed me in the right direction whenever I needed help. Indeed, even if it was the oddest time of the day, I knew I could discuss any problem with them. Perhaps the most important thing they taught me in my course of working with them was learning how to think and analyse a problem. To them, I am extremely grateful for the valuable experience they provided.

My mother, Kakali Mukherjee, has been a constant inspiration for me, and a source of huge encouragement during these hard and uncertain times. Her little sacrifices made it possible for me to work uninterrupted while locked in my home.

I also want to appreciate the help of my friend, Uttara Mazumdar, during the time I worked on my dissertation. She has constantly made sure that I face no mental fatigue during the course of my work.

Thesis Organization

We have divided this thesis into three broad parts. Chapter 1 is mostly dedicated to the explanation of a coreset , the usefulness of the coreset , its importance , and yet not having any standard universal definition as of yet. We provide a small example of a coreset to show how efficient coresets can be.

In Chapter 2 , we move to our specific problem; coresets for K-means. Chapter 2 covers a lot of previous work in detail , coresets for K-means for points in $\mathbb{R}^d$. We try to explain the little tricks that ultimately builds an efficient coreset.

Chapter 3 is a compilation of our own work. We try to extend the idea of coresets for K-means for flats and hyperplanes in $\mathbb{R}^d$. We follow one simple approach, if there exists some general procedure to map flats (of any dimension) or hyperplanes to points, then effectively we can use the previous techniques to build our coreset. Our work indeed makes it possible to extend it to flats, hyperplanes and even a combination of them in $\mathbb{R}^d$, but only if $K = 1$, that is for the 1-center problem.

Despite this major drawback, this work stands apart from previous attempts, where the method could be used for K-means, but only for lines in $\mathbb{R}^d$, putting a constraint on the dimensions of the objects being clustered.

Contents

1 Introduction to Coresets **7**
1.1 Introduction . . . 7
1.2 Coreset for MEB for points in $\mathbb{R}^d$ . . . 7

2 Coresets for points in $\mathbb{R}^d$ for K-means clustering **9**
2.1 Definitions . . . 9
2.2 A Simple Unsuccessful Attempt . . . 10
2.3 Importance Sampling: Concept of Sensitivity . . . 11
2.3.1 A Measure of Sensitivity . . . 12
2.3.2 Bi-Criteria Approximation Algorithm : D^2 sampling . . . 13
2.3.3 Bounding the Sensitivity using bi-criteria approximation . 14
2.4 From Single Center Set to All Center Sets . . . 16
2.4.1 Some More Definitions (And One Theorem) . . . 16
2.4.2 Some More Intuition, And Working on Them . . . 17
2.4.3 The Formal Proof . . . 18
2.5 The Construction of The Coreset . . . 19

3 Coresets for Flats and Hyperplanes: Our Attempts **21**
3.1 Proof of Theorem 6: Center Lies Inside Convex Hull . . . 23
3.2 Center by descent . . . 24
3.2.1 Definitions . . . 24
3.2.2 Inputs . . . 25
3.2.3 Analysis . . . 26
3.2.4 Discussion and Scalability . . . 29
3.2.5 Time Complexity . . . 29
3.3 A random walk attempt to find the center . . . 29
3.3.1 Modifications to algorithm 4 . . . 30
3.3.2 The Random Walk . . . 30
3.3.3 Probability of reaching the center from n steps away . . . 30
3.3.4 Expected number of steps to reach the center . . . 31
3.3.5 Scaling for higher dimensions . . . 32
3.4 Coreset For 1-center for flats and hyperplanes in $\mathbb{R}^d$ . . . 32
3.4.1 Analysis . . . 33
3.4.2 Discussion and Scalability . . . 34

Future Work **35**

Bibliography **37**

Chapter 1

Introduction to Coresets

1.1 Introduction

A coreset, defined in very broad and informal terms is a small subset of data that captures the key features of the data set that we care about. With the generation of huge volumes of data every day by billions of users, it becomes difficult in both space and time parameters to store and extract relevant information when the volume of the data is so huge. Consequently, we carefully sample certain data points only, using some intelligent distribution, that extracts all relevant information, when we run our algorithms on this smaller data set. This helps us facilitate fast approximation algorithms for various geometric optimization problems.

In this article, we talk mainly about building coresets for the K-means problem for data in $\mathbb{R}^d$. There have been substantial work in coresets for points in $\mathbb{R}^d$, with the coreset size being lower bounded by d, the number of dimensions, k, the number of clusters and ϵ, the parameter for accuracy of the approximation. Work had also been done in coresets for lines in $\mathbb{R}^d$. In this thesis, we try to extend on the previous work, and attempt to find a coreset for k-flats in $\mathbb{R}^d$.

Since our work will mostly revolve around the K-means problem, our definition of coresets will be modified accordingly. However, it is important for us to take up a simple example of a coreset to fully appreciate its power. For this, we next look at the coreset for a minimum enclosing ball for a set of points [Badoiu and Clarkson [2008]] .

1.2 Coreset for MEB for points in $\mathbb{R}^d$

Definition 1.2.1 (MEB)
The minimum enclosing ball (MEB) for a set of points P in $\mathbb{R}^d$ is the radius of the smallest sphere that contains all the points of P.

Definition 1.2.2 (Coreset for MEB)
Given a set of points $P \subset \mathbb{R}^d$ and a value $\epsilon > 0$, a coreset $S \subset P$ has the

property that the MEB containing S is within ϵ of the MEB containing P. That is, if the smallest ball containing S is expanded by $1+\epsilon$, then the expanded ball contains P.

The algorithm to construct the coreset is surprisingly simple, a greedy algorithm is sufficient for the construction. The coreset is of size at most $2/\epsilon$, where ϵ is the factor in the definition above.

Algorithm 1 (Coreset for MEB)

```
1. Choose a random point $x \subset P$
2. Include $x$ in $S$, $S$ being initialized as an empty set.
3. for $i = 1$ to $2/\epsilon$
        Let $c$ be the center of $S$
        Find $p \in P$ such that $\forall p' \in P$, $||p-c||_2 \geq ||p'-c||_2$ , where $||.||_2$ denotes the second norm.
        $S \leftarrow S \cup \{p\}$
4. Return $S$
```

The time complexity of the above algorithm is exponential in d, the number of dimensions and linear in n, the number of points. By treating S, as a multiset, it is possible, using the same algorithm, to construct a coreset with a slightly lower time complexity, but the coreset size increases to at most $1/\epsilon^2$.

We do not present a correctness proof and analysis for the above algorithm, since it was used as a simple example of a construction. We are more interested in building coresets for K-means clustering. We next look at the construction of efficient coresets for a point set in $\mathbb{R}^d$.

Chapter 2

Coresets for points in $\mathbb{R}^d$ for K-means clustering

Before we look at the actual ideas for the construction of coresets for points, we need to know a few definitions that will come in handy.

2.1 Definitions

Definition 2.1.1 ($Cost(P, C)$)
Consider a set of weighted points $P \subset \mathbb{R}^d$, with weights w_i, $i \in [|P|]$, where $[A], A \in \mathbb{Z}^+$ indicates the set $1, 2, \ldots, A$ and a set of centers $C \subset \mathbb{R}^d$, $|C| = k$. *Then $Cost(P, C) = \Sigma_{i \in [|P|]} w_i f_Q(p_i)$, $p_i \in P$, where*
$f_Q(p_i) = ||c_j, p_i||_2$, $c_j \in C$ such that $\forall c_k \in C; c_k \neq c_j$, $||c_j, p_i||_2 \leq ||c_k, p_i||_2$

Definition 2.1.2 (K-means clustering for weighted points)
Consider a set of weighted points $P \subset \mathbb{R}^d$, with weights w_i, $i \in [|P|]$, and a family of sets ζ, where each $C \in \zeta$ is set of centers, that is, $C \subset \mathbb{R}^d$, $|C| = k$. Note that ζ contains all possible set of centers in $\mathbb{R}^d$. Then the K-means algorithm returns to us the optimal center set such that if the set $C \in \zeta$ is returned then:
$\forall C' \in \zeta$, $C' \neq C$, $Cost(P, C') \geq Cost(P, C)$

Definition 2.1.3 (Coreset)
Let $\epsilon > 0$, and ζ be a family of sets such that each $C \in \zeta$ is a set of centers, that is $C \subset \mathbb{R}^d$. As before ζ contains all possible set of centers in $\mathbb{R}^d$. Then X is an ϵ-coreset of P, if:
$\forall C \in \zeta$, $|Cost(P, C) - Cost(X, C)| \leq \epsilon \; Cost(P, C)$
The above is the definition of a strong coreset. If the inequality holds only for C^*, the optimal set of centers, then X is called a weak coreset of P.

Note that a direct result from the definition of coreset is that if P is the point set and X is the coreset of P with C_P and C_X being the optimal center sets for P and X respectively, then $Cost(P, C_X) \leq (1 + 3\epsilon)Cost(P, C_P)$

2.2 A Simple Unsuccessful Attempt

We look at some simple and unsuccessful attempts to build a coreset. The trivial coreset is the point set P itself, but we are interested in building smaller coresets. Our first attempt is to sample points with a uniform probability of $1/|P|$. Intuitively, we may be able to see that why this is problematic. The spatial distribution of the chosen points will be a function of the density of clusters or sub clusters inside what should have been optimally a larger cluster. Therefore, under representation of certain points from low density clusters may take place, significantly changing the cluster formation. To counter this, we may have to sample impractically large number of points, defeating our whole aim.

Let us now formally look at the analysis of uniform sampling. For simplicity, assign weights to points in P as $\forall i, i \in [|P|] w_i = 1/|P|$.

This implies that the cost for any set of centers C for the point set P is

$$Cost(P, C) = \Sigma_{i\in[|P|]}\,(1/|P|) f_Q(p_i) = \mathbb{E}_p[f_Q(p)] \tag{1}$$

Now suppose we set the weights of the points in the coreset to be $1/m$, $|X| = m$. Then we can see that the cost for the same set of centers, that is C is :

$$\mathbb{E}[Cost(X, C)] = \Sigma_{p\in X}\,(1/m)\mathbb{E}_x[f_Q(p)] = Cost(P, C)$$

Note: This holds because each data point has been sampled with probability of $1/|P|$

We see that $Cost(X, C)$ is an unbiased estimator of $Cost(P, C)$, which shows that theoretically it is possible for us to build a coreset by using a uniform distribution for sampling. However, as discussed before, there are some caveats.

$$\begin{aligned}
Var[Cost(X, C)] &= Var[\Sigma_{p\in X}\,(1/m) f_Q(p)] \\
&= [m.(1/m^2)Var[f_Q(p)] \\
&= (1/m)Var[f_Q(p)] \leq (1/m)\mathbb{E}[f_Q(p)^2] \\
&= (1/mn)\Sigma_{p\in P}\, f_Q(p)^2 \leq (1/mn)\Sigma_{p\in P}\, f_Q(p)^2 \\
&= (n/m)Cost(P, C)^2
\end{aligned}$$

where n is the number of points in P.

Again, we are met with success. The variance is finite, so using the weak law of large numbers, we know that we can make $Cost(X, C)$ converge to its expected value. Let us use Chebyshev inequality to get a bound on the number of points required.

$Pr[|Cost(X, C) - \mathbb{E}[Cost(X, C)]| > \epsilon Cost(P, C)] \leq Var^2[Cost(X, C)]/\epsilon^2 Cost(P, C)^2$.

Knowing that $\mathbb{E}[Cost(X, C)] = Cost(P, C)$, we get the desired probability as $n/(\epsilon^2 m)$. Hence, to get a probability of δ, we need $n/(\epsilon^2\delta)$ number of points. This exceeds the initial number of points!

Our initial intuition was true, uniform sampling does not help much. We need a different type of sampling, a more careful one, that takes into account the less dense areas of our point set. This type of sampling is called *importance sampling.*

2.3 Importance Sampling: Concept of Sensitivity

The difference between equality and equity is that equality aims to provide equal opportunities for everyone, irrespective of the initial conditions. Equity is more about making sure that everybody has the same footing in the end. We tried providing equality to all the data points, but unfortunately, owning to the initial conditions, representatives of sparse clusters were, on expectation, less to be selected. Hence, it is important that we now provide equity to all the data points. It is here that the concept of sensitivity comes in. Sensitivity is a measure of how "hard hit", a point can be, due to the choice of centers, i.e., the cost for that point in the worst chosen center set. Obviously, we do not want our points to be the worst hit. Hence, intuitively, if we define a distribution that gives us some kind of ratio between the total sensitivity and the sensitivity of a point, then we know how much "sensitive" the point is in comparison to the other points, and get a guess on how much increment in cost can the point cause in case a bad center is chosen. A more "sensitive" point (relative to the other points) must be, quite naturally, given more preference.
For a mathematical intuition of how sensitivity should be calculated, let us assume some distribution $I(p)$ is used to sample points. If $W_P(p)$ is the weight of the point p in P, if sampled, the weight of p is modified to $W_X(p) = W_P(p)/(mI(p))$, where m is the number of points in the coreset X. We verify that using this modified weight, the sampling scheme is unbiased.

$$\begin{aligned}\mathbb{E}[Cost(X,C)] &= \mathbb{E}[\Sigma_{x\in X} W_X(x) f_Q(x)] \\ &= \mathbb{E}[\Sigma_{x\in X} f_Q(x) W_P(x)/(mI(x))] \\ &= \mathbb{E}[f_Q(x) W_P(x)/I(x)] \qquad [\because \mathbb{E}[\mathbb{E}[x]] = \mathbb{E}[x]] \\ &= \Sigma_{x\in P} W_P(x) f_Q(x) \qquad [\because I(x) \text{ is the distribution}] \\ &= Cost(P,C)\end{aligned}$$

Now, we turn to the variance. The variance must be reduced to the minimum. What if we try to reduce the variance for a single center set, C ? Can we get an indication of what $I(p)$ in the process of reducing the variance?

$$Var[Cost(P,C)] = (1/m)Var[W_P(x) f_Q(x)/I(x)]$$

The above statement is true because the points are independently sampled according to the distribution $I(x)$ Therefore, we have:

$$\begin{aligned}Var[Cost(P,C)] &= 1/m(\mathbb{E}[W_P(x)^2 f_Q(x)^2/I(x)^2] - \mathbb{E}^2[W_P(x) f_Q(x)/I(x)) \\ &= (1/m)\Sigma_{x\in P} W_P(x)^2 f_Q(x)^2/I(x) - (1/m)\Sigma_{x\in P} W_P(x)^2 f_Q(x)^2 \\ &= (1/m)\Sigma_{x\in P} W_P(x)^2 f_Q(x)^2/I(x) - (1/m)Cost(P,C)^2\end{aligned}$$

The minimum variance is obtained when the expression for $I(x)$ is:

$$I(x) = W_P(x) f_Q(x) / \Sigma_{x' in P} \left(W_P(x') f_Q(x') \right)$$

The expression shows that we are practically checking how bad can the increase in cost be due to x, with respect to the other points if our center set C that we chose is not a good one. This is where we first see a mathematical expression for *sensitivity*. Note that this concept provides for equity, the more "sensitive" a point is, the greater the probability of it getting sampled.

It is time for us to get introduced with the concepts of sensitivity formally. Using this we shall try figuring out the optimal distribution for all center sets.

2.3.1 A Measure of Sensitivity

Langberg and Schulman [2010] formalized the concept of sensitivity by showing that to find a good distribution, we needed to consider the worst impact a point could have due to a bad center set. They defined sensitivity analogous, but slightly different, from what we proved just before:

$$\sigma(x) = \mathit{supremum}_{C \in \zeta}\, f_Q(x) / \Sigma_{x' \in P} (W_P(x') f_Q(x'))$$

Notice how similar this is to the previous expression we derived. Here, however, we choose the worst case scenario for each point, instead of a fixed center set.

It might not be feasible for us to determine the exact sensitivity; that would require the knowledge of the spatial orientation of the points, therefore, we resort to a upper bound on the value of σ.

Let $s(x) = O(\sigma(x))$, and define $S = \Sigma_{x \in P} W_P(x).s(x)$. Consider the following sampling distribution:

$$I(x) = W_P(x).s(x)/S(x)$$

Intuitively, we see that this is almost like the expression we derived earlier in section 2.3, except that $f_Q(x)$ has been replaced by $s(x)$, the (upper bound on) sensitivity. This makes sense, because earlier, we had just a single center set, so we did not have to worry about the worst case scenario for a point. Whereas earlier, we just to take into account the cost of each point from an appropriate center for a *single center set*, we now need to take into account the worst case scenario of each point, which may arise from *different center sets* for each point. Hence, the justifiable change in the expression. Let us now look at the analysis a little more formally, although the proper proof will be discussed later.

For all $C \in \zeta$ and for all points x, we define $G_C(x)$ as :

$$\begin{aligned} G_C(x) &= W_p(x) f_Q(x) / (Cost(P, C) I(x) S) \\ &= f_Q(x) / (\Sigma_{x' \in P} (W_P(x') f_Q(x')) s(x)) \end{aligned}$$

Above expression must always be less than 1 and greater than 0, because s(x) will cancel out the numerator, by definition. Since the value is always between 0 and 1 we may use Hoeffding bounds, which will prove to be useful. We will use precisely this form:

If $X_1, X_2, \ldots, X_n$ are independent random variables such that their values are bounded by $[0, 1]$, then let us define $\overline{X} = (1/n)\Sigma_{i=1}^{n} X_i$. Then the Hoeffding inequality (the form we require) states:

$$Pr(|\mathbb{E}[\overline{X}] - \overline{X}| \geq t) \leq 2e^{-2nt^2}$$

We have
$$\begin{aligned}\mathbb{E}[G_C(x)] &= \Sigma_{x \in P}\, I(x)W_P(x)f_Q(x)/(Cost(P,C)I(x)S) \\ &= (1/Cost(P,C))\Sigma_{x \in P}\, W_P(x)f_Q(x)/S \\ &= 1/S \qquad [\because \Sigma W_P(x)f_Q(x) = Cost(P,C)\ ]\end{aligned}$$

We also have $(1/m)\Sigma_{x \in X}\, G_C(x)$, as $Cost(X,C)/SCost(P,C)$, replacing $I(x)$ by its expression.

Therefore, using Hoeffding inequality now, we get:

$$Pr(|Cost(P,C) - Cost(X,C)| \geq tSCost(P,C)) \leq 2e^{-2mt^2}$$

From here we see that X is a good coreset for any single query with probability $1 - \delta$ if the number of samples required are:

$$m \geq (S^2/(2\epsilon^2)log(2/\delta)$$

A slightly depressing observation is that the bound is dependent on S^2, the square of the sum of the upper bound on sensitivities. This means that if we have a loose upper bound, we might end up needing absurd number of samples. If we choose the trivial bound $s(x) = n$ and $W_P(x) = 1/n$, then we end up with a uniform distribution scheme. In this case, note that the number of samples required is lower bounded by n^2, leading to the same absurd result we talked about in the beginning of the report. This leads to the question, do we really have a tight bound on Sensitivity? The answer, fortunately, is yes. We first try to find an approximation of the K-means clustering problem, with some approximation algorithm that returns us a set of not necessarily k centers, and a approximate bound on the cost. Such algorithms are called α-β algorithms or *bi-criteria* algorithms. And we have a great one at our disposal.

2.3.2 Bi-Criteria Approximation Algorithm : D^2 sampling

If we are given a point set P and asked to guess a cluster center, given the location of a previous cluster center, our instinct would be to pick a cluster center far away from the existing cluster center, because we would hope that the existing center "takes care" of the points near it. The D^2 sampling method is built on the same principle. This algorithm returns to us βk cluster centers such that for the set of βk centers, let us call it B, satisfies:

$Cost(P, B) \leq \alpha Cost(P, OPT)$,
where OPT is the Optimum center set, $|OPT| = k$. The algorithm, due to Arthur and Vassilvitskii [2007] is provided below:

Algorithm 2 (D^2 Sampling)

```
1. Sample $p \in P$ using a uniform distribution. Add it to empty set $B$
2. for $i = 2$ to $k$
        Sample $p_i$ with probability $W_P(p_i)d(p_i, B)^2/(\Sigma_{p' \in P} W_P(p')d(p', B)^2$ and add to $B$.
3. Return $B$
```

Here $d(p, B)$ means distance of point p from the point in B to which it is closest.

It can be shown that this algorithm is $O(\log(k))$ competitive, that is

$$\mathbb{E}[Cost(P, B)] \leq 8(\log_2 k + 2)Cost(P, OPT)$$

Now, by Markov Inequality, we say that:

$$Pr(Cost(P, B) > 16(\log_2 k + 2)Cost(P, OPT)) \leq \mathbb{E}[Cost(P, B)]/16(\log_2 k + 2) \leq 1/2.$$

Therefore, if we have t trials of the experiment, and the center set with the lowest cost be B^*, then if $Cost(P, B^*) > 16(\log_2 k + 2)Cost(P, OPT)$, that means all the other trials had failed too. This happens with a probability of less than $(1/2)^t$. Then if $(1/2)^t = \delta$, then $t = \log(1/\delta)$.

Therefore, if we ran the experiment $\log(1/\delta)$ times, and chose the center set with the lowest cost, then: $Pr(Cost(P, B^*) \leq 16(\log_2 k + 2)Cost(P, OPT)) > 1 - \delta$

From this we get that $\alpha = 16(\log_2 k + 2)$ and $\beta = 1$.

2.3.3 Bounding the Sensitivity using bi-criteria approximation

Any bi-criteria approximation can be used for sensitivity. We present a previously proven theorem on the bound of the sensitivity ($\sigma(.)$) for (α, β) approximation algorithms.

Theorem 1 (Sensitivity Bound)
Let $P \subset \mathbb{R}^d$ be a point set, $|P| = n$, $W_P(p_i) = 1/n, \forall p_i \in P$. Let $B \subset \mathbb{R}^d$ be the center set returned by an α-β algorithm for K-means clustering using squared Euclidean distance. *Let for $p \in P$, $b_p \in B$ denote the cluster center of the cluster it is assigned to. Let $d(x, y)$ denote the Euclidean distance between points x and y. Let $d(x, Y)$ denote the distance between x and the most appropriate center from x from the center set Y. Also let P_p be the set of all points*

belonging to the cluster with cluster center b_p. Then the sensitivity of a point p for the function $f_Q(p) = \min d_{c \in C}(p, c)^2$ is bounded for all $p \in P$ by the function:

$$s(p) = (2\alpha d(p, b_p)^2/\overline{c}_B) + (4\alpha \Sigma_{p' \in P_p} d(p', b_p)^2/(|P_p|\overline{c}_B)) + (4n/|P_p|)$$

where $\overline{c}_B = 1/n\Sigma_{p' \in P} d(p', B)$

Also, the total sensitivity is bounded by:

$$S = (1/n)\Sigma_{p \in P} s(p) = 6\alpha + 4\beta$$

Proof of the above theorem:

Consider a point $p \in P$, and a center set $C \in \zeta$, chosen arbitrarily, and set:

$$\overline{c}_C = (1/n)\Sigma_{p' \in P} d(p', C)$$
$$\overline{c}_B = (1/n)\Sigma_{p' \in P} d(p', B)$$

Using the Minkowski inequality (this inequality proves that L^p spaces are *normed spaces*) for $p = 2$ [1], we get:

$$d(p, C)^2 \leq 2d(p, b_p)^2 + 2d(b_p, C)^2$$

In a similar fashion, $\forall p' \in P$ we have,

$$d(b_p, C)^2 \leq 2d(p', b_p)^2 + 2d(p', C)^2$$

Summing over all points $p' \in P$, we have:

$$d(b_p, C)^2 \leq (2/|P_p|)\Sigma_{p' \in P} [d(p', b_P)^2 + d(p', C)^2]$$

Now, we somehow need to get the expression for $\sigma(p)$ in the left. In the right, we have a upper bound on the left side expression, hence it serves our purpose as $s(p)$.

Note that $d(p, C)^2 = (f_Q(p)/\Sigma((1/n)^2 f_Q(p')))((1/n)\Sigma d(p', C)$. It follows from the definition of $\sigma(p)$ that

$$\sigma(p) \leq (2/\overline{c}_C)[d(p, b_p)^2 + d(b_p, C)^2]$$

We are almost there. All we need to do now is to find the relationship between $\overline{c}_C$ and $\overline{c}_B$ and we will be done with a trick. By definition, we know that

[1]Not to be confused with our convention of denoting a point as p sometimes
[2]This is the weight, check theorem 1

$\bar{c}_C \geq 1/\alpha\bar{c}_B$ and $\bar{c}_C \geq (1/|X|)\Sigma_{p' \in P}\, d(p', C)^2$. From these, and from the second *Minkowski inequality* [see above], we have :

$$\begin{aligned}\sigma(x) &\leq (2d(p, b_p)^2/\bar{c}_C) + 4(\Sigma_{p' \in P_p}\, [d(p', b_p)^2 + d(p', C)^2]/(P_p\bar{c}_C)) \\ &\leq (2\alpha d(p, b_p)^2/\bar{c}_B) + ((4\alpha\Sigma_{p' \in P_p}\, d(p, b_p)^2)/(P_p\bar{c}_B)) + (4n/P_p)\end{aligned}$$

This is the value of $s(p)$ since it upper bounds $\sigma(p)$. Furthermore, we can now derive the value of S.

$S = (1/n)\Sigma_{p \in P}\, s(p) = 6\alpha + 4\beta$

2.4 From Single Center Set to All Center Sets

Up until now, we had put aside the problem of the coreset property holding for all center sets; we have in previous sections used the Hoeffding inequality very cleverly to get to satisfy the coreset property for a *single* center set, $C \in \zeta$. It is now important for us to build an ϵ-coreset that satisfies the coreset property for all center sets.

On the top of our heads, we might want to apply the union bound on all center sets. But the caveat is there are infinite of them. We really can not have a success, unless we have some way to bound the number of queries. Our aim is to show that if some property (not necessarily the ϵ-coreset property) holds for a small subset of center sets, our required condition (the ϵ-coreset property) holds for all center sets. Fortunately, we do have such a tool; it is possible to show that if for a subset $\zeta^* \subset \zeta$, the coreset X satisfies the *strong* coreset property, with the error factor as $\epsilon/2$, then all center sets satisfy the coreset property with error factor ϵ.

2.4.1 Some More Definitions (And One Theorem)

Before we proceed further, we will need to look at (and understand) some more definitions. Let us look at them at one by one.

Definition 2.4.1 (Shattering)
Consider a family of functions $\mathbb{F}$ and a set of points $P \subset \mathbb{R}^d$. Now let $f \in \mathbb{F}$. Geometrically the function divides the space into two parts, say A and B. Let us label all points in A as $+$ and all in B as $-$. Therefore, if we consider all points as distinct, we get a sequence of $+$ and $-$. If there exists at least one way of placing the points , such that every combination of this sequence can be generated by some function $f \in \mathbb{F}$, then we say F shatters P.

Definition 2.4.2 (VC Dimension)
The maximum value of $|P|$ for which $\mathbb{F}$ shatters it is called the VC dimension of $\mathbb{F}$. If arbitrarily large sets can be shattered by $\mathbb{F}$, then its VC Dimension is infinity.

Definition 2.4.3 (Pseudo Dimension)
Pseudo-dimension is in one way the extension and generalization of VC dimension. Consider a countably infinite domain X and a family of functions $\mathbb{F} : X \to [0, 1]$. Suppose $x_1, x_2, \ldots, x_t$ are all elements of X, and we also have a set of thresholds $th_1, th_2, \ldots, th_t \in [0, 1]$. Let $b_i, i \in [t]$ take values $+$ or $-$, i.e., $b_i = +$ if and only if for any $f \in \mathbb{F}$, $f(x_i) > th_i$. If t is the greatest cardinality, such that for all combinations of $b_i s$, we have a function $f \in \mathbb{F}$ that generates the sequence, then t is the pseudo-dimension of $\mathbb{F}$

Theorem 2 (Haussler [1995])
Packing theorems deal with the number of distinct vectors that can be packed in a cube of $[0, 1]^n$, when the vectors are dissimilar on at least k indices. Haussler's (generalized) packing lemma says that "For any set X, any probability distribution P on X, any set $\mathbb{I}$, of P-measurable functions[3] on X taking values in the interval $[0, 1]$, with the pseudo-dimension of I being $d < \infty$, and an $\epsilon > 0$, then the ϵ packing number is $O(\epsilon^{-d})$".

2.4.2 Some More Intuition, And Working on Them

Haussler's lemma directly hints at what we want. We know that $G_C(x)$ takes values between 0 and 1, and that for each center set, when we consider the coreset X, the values of $G_C(x)$ occupies a point in the cube $[0, 1]^m$. But clearly, we now know that $O(\epsilon^{-d})$ center sets (giving us equal number of $G_C(x)$) can be used to pack up the space. The elements of ζ can be covered up to a L_1 distance of $\epsilon/2$ by a set $\zeta^* \in \zeta$ of ϵ^{-d} points. This is exactly what we wanted, a small set of center sets, so that we may use the union bound!

This set ζ^* is special, if a coreset X follows the strong coreset property with an error of $\epsilon/2$ for ζ^*, it follows the strong coreset property with an error ϵ for the rest of the center sets in ζ.

Let us now state the theorem directly that we get by applying these ideas, before we go on to a proof sketch of them:

Theorem 3 (Coreset Size)
Let $\epsilon > 0$ and $\delta \in (0, 1)$, P, a weighted data set, ζ be the set of all center sets, and $f_Q(x)$ a cost function as defined previously. Let $X; |X| = m$ be the coreset sampled with replacement from P following the distribution $I(x) = W_P(x)s(x)/S$. Let the weights of the points in the coreset be changed to $W_X(x) = W_P(x)/(mI(x))$. Let d be the pseudo-dimension of $G_C(x)$. Then X is a coreset for P with probability at least $1 - \delta$ for

[3]Random Variable in this context

$$m \in \Omega((S^2/\epsilon^2)(d + log(1/\delta)))$$

Proof sketch of the above theorem:

The actual proof involves the chaining technique of proving. We will provide a proof that is $1/\epsilon$ close to the result in Theorem 3.

Remember how we used the Hoeffding bounds previously? We will do the same this time to determine a size of the coreset.

$$Pr(\exists C \in \zeta : |Cost(P, C) - Cost(X, C)| \geq \epsilon Cost(P, C))$$
$$\leq Pr(\exists C^* in \zeta^* : |Cost(P, C^*) - Cost(X, C^*)| \geq (\epsilon/2)Cost(P, C^*))$$
$$\leq 2|\zeta^*|e^{-m\epsilon^2/(2S^2)}$$

Where the last inequality comes from an union bound inequality and previous results from the Hoeffding bound.

We know that $|\zeta^*| = \epsilon^{-d}$, where d is the pseudo-dimension of $G_C(x)$

From this we get that if the set X is a coreset with probability $1 - 1/\delta$, then :

$$m \in \Omega((S/\epsilon)^2(d \log(1/\epsilon) + \log(1/\delta))$$

2.4.3 The Formal Proof

Theorem 4 (Li et al. [2001])
Let $\alpha > 0, \nu > 0, \delta > 0$, and let P be a fixed countable infinite domain, and $I(.)$ be any probability distribution over X. Let $\mathbb{F} : X \to [0, 1]$, pseudo-dimension of $\mathbb{F} = d$. Let X a coreset sampled according to $I(.)$ and $|X| = m$. Now if $m \in \Omega((1/(\alpha^2\nu))(d \log(1/\nu) + \log(1/\delta))$, then with a probability of $1 - \delta$, it holds that:

$$\forall f \in \mathbb{F} : d_\nu(\Sigma_{p\in P} I(p)f(p), (1/|X|)\Sigma_{p\in X} f(p)) \leq \alpha$$

where $d_\nu = |a - b|/(a + b + \nu)$

We proceed by choosing $\nu = 1/2$ and $\alpha = \epsilon/(3S)$. Applying the above theorem to the family of functions $G_C(.)$, this means for $m \in \Omega((S^2/\epsilon^2)(d + \log(1/\delta)))$, with probability $1 - \delta$

$$(|\Sigma_{p\in P} I(p)G_C(p) - (1/|X|)\Sigma_{p\in X} G_C(p)|) \;/\; (\Sigma_{p\in P} I(p)G_C(p) + (1/|X|)\Sigma_{p\in X} G_C(p) + (1/2)) \leq \alpha$$

Since $I(p)G_C(p)$ and $(1/|X|)\Sigma_{p\in X}G_C(p)$ are both bounded by 1, we can write

$$|\Sigma_{p\in P}\, G_C(p)I(p) - (1/|X|)\Sigma_{p\in X}\, G_C(p)| \leq 3\alpha = \epsilon/S$$

By definition of $G_C(.)$, we have:

$$\Sigma_{p\in P}\, I(p)G_C(p) = 1/S \text{ and } (1/m)\Sigma_{p\in X}\, G_C(p) = Cost(X,C)/(SCost(P,C))$$

Multiplying with $SCost(P,C)$ implies:

$$\forall C \in \zeta,\ |Cost(P,C) - Cost(X,C)| \leq \epsilon Cost(P,C)$$

which completes the proof.

2.5 The Construction of The Coreset

We are now ready to build the coreset for points in $\mathbb{R}^d$ for K-means clustering. One may notice that the pivot of the coreset construction is the intelligently designed function $G_C(.)$. This function gave us not only the correctness proof for a single center set, but also made sure that it had properties by virtue we could use the packing theorem, and consequently contained a major hurdle, the problem of infinite center sets. But the question remains, what is the pseudo dimension of $G_C(.)$? We take a look at one of the bounds of the pseudo dimension of $G_C(.)$.

If $\mathbb{D}$ is the number of dimensions, and d is the pseudo-dimension of $G_C(.)$, then $d = \mathbb{D}k\log k$. We are now absolutely ready to build our coreset.

Before we go the formal algorithm, let us take an informal look at how to build a coreset in general.

We first start by defining a sensitivity of the points, and then define a distribution which depends on the sensitivity of the points. This helps us choose points with equity.

In order to bound this sensitivity, so that it can be efficiently used to define the distribution, we use a bi-criteria approximation and bound the sensitivity based on its results.

We then define an intelligent function, that preferably takes values between $[0, 1]$. This function is constructed out of both the distribution we use to sample points, as well as the center set used (this is why $G_C(.)$ takes the cost into account as well). The function serves two purposes:
For a single query, the function gives us a bound on the difference between the cost due to the coreset, and this function, being bounded, can be used to invoke the packing lemma, allowing us to use properties for a finite number of center

sets.
Note that since the function depends on Cost, it ultimately depends on the center set. Since it also depends on the sampling distribution, it actually encodes both the properties of the point set P and the center set ζ in itself. This is why this function is so important.

Invoking the packing theorem, we build our coreset, based on a finite number of center sets.

We now present a theorem and the algorithm of the coreset following it:

Theorem 5
If $\epsilon \in (0, 1/4), \delta > 0, k \in \mathbb{N}, P \subset \mathbb{R}^n, B \subset P$, such that B is the center set with the smallest quantization error in $1/\delta$ runs of the following algorithm. Let X be the output of the following algorithm with

$$|X| \in \Omega((nk^3 \log k + k^2 \log(1/\delta)/\epsilon^2)$$

Then, with probability at least $1 - \delta$, the set X is a ϵ-Coreset for P

Proof follows from all the previous works we have done. [4]

Algorithm 3 (Coresets for K-means for points)

```
Input : $P, k, B, m$
Output :$X$
1. $\alpha \leftarrow 16(logk + 2)$
//Preparing to bound the sensitivity. Form all clusters first.
2. for each $b_i$ in $B$
    $B_i \leftarrow$ Set of all points whose cluster center is $b_i$
3. $c \leftarrow (1/|P|)(\Sigma_{p' \in P}\, d(p', B))$
//Now calculating s, the upper bound of $\sigma(.)$
4. for each $b_i \in B$ and $p \in B_i$
    $s(x) \leftarrow (\alpha d(p, B)/c) + (2\alpha(\Sigma_{p' \in B_i}\, d(p', B))/(B_i c)) + (4|P|/B_i)$ // Bounded
5. for each $p \in P$
//Forming the distribution
    $I(p) \leftarrow s(p)/(\Sigma_{p' \in P}\, s(p'))$
//Once sampling is done, we now pick points and assign them appropriate new weights so that the cost is reduced. Note that our original set was an unweighted set of points.
6. $X \leftarrow$ Sample m points by using the distribution $I(p)$. Assign to each sampled point the weight $1/(mI(p))$
7. Return $X$
```

[4]A good reference for the topic and methods discussed above can be found in Olivier Bachem [2017]

Chapter 3

Coresets for Flats and Hyperplanes: Our Attempts

In the previous section, we talked about coresets for points; but data may not always be simple points. It may be the case that some attribute values are missing; these points would not be a point in $\mathbb{R}^d$, these would be hyperplanes and flats. It is important that we develop clusters for such objects too.

Marom and Feldman [2019] developed a theory for coresets for lines; but they used their own general framework[Feldman and Langberg [2011]], that made the calculations complicated. Their solution was specific to lines only, and they left an open problem at the end of their paper for coresets for $k-flats$ in general.

In our attempts to solve the problem, we noticed two features on the previously used methods: There was no general function $G_C(.)$, that could be used, in fact, the nature of $G_C(.)$ should be such that the sampling distribution must be embedded into the function. However, the sampling distribution itself depended on a bi-criteria approximation, that was unique for each flat, and could also depend on the total number of dimensions. This makes constructing a general coreset for any flat extremely difficult using this process.

What we have tried to do is to use some kind of reduction technique. Indeed, if we could have a function that mapped the flats to specific points; we could attempt to use the Coreset building technique for points for any flat in any dimension. That has been our primary aim, although this would result in an overhead due to the mapping and inverse mapping of the flats to points.

Algorithms and proofs provided in this section are mostly for lines, but they can be easily generalized to any flats. Also, before we start working on coresets for flats, we may need a few theorems and definitions that can come in handy.

Definition 3.0.1 (1-center or center)
For a set of hyperplanes or flats $\mathbb{F}$ in $\mathbb{R}^d$, the 1-center or simply center (used interchangeably unless where mentioned explicitly as a footnote) is a point $p \in \mathbb{R}^d$ such that

$$\forall p' \in \mathbb{R}^d, p' \neq p, \Sigma_{f \in \mathbb{F}}\, dist(f,p) \leq \Sigma_{f \in \mathbb{F}}\, dist(f,p')$$

where $dist(a,b)$ denotes the Euclidean distance between the flat a and the point b.

Theorem 6
Lines in $\mathbb{R}^n$ in general position have their $1-center$ inside the convex hull formed by the points due to the intersection of the lines.

Proof: Before we go to the proof, a few lemmas will come in handy

Lemma 1
Consider a line l passing through an intersection point i. Then if p and p' are two points on l, such that p' is closer to i than p, then p' is closer to the lines forming the intersection point i than p.

Proof: Trivial

Lemma 2
Consider two points p and p' which are contenders for being the 1-center of the lines in $\mathbb{R}^n$. Then p' is a better contender for the 1-center of the lines than p if p' is or at least as close or closer than p to all points $i \in I$, the set of all intersecting points, and that for at least one $i \in I$, p' is closer to i than p.

Proof follows from previous lemma.

Note: The lemma says point p' must be closer to all intersection points, rather than a specific point or a subset of points, because only then can we ignore the number of lines intersecting at an intersection point i.

Lemma 3
The 1-center for k points in $\mathbb{R}$ lies between the leftmost and rightmost point, i.e., inside the convex hull of the points.

Proof: (By contradiction) Suppose that there was a point outside the convex hull of the k points, which is the optimum 1-center. Without loss of generality, we assume that this point p is to the left of all the k points. Let us denote the leftmost point of this k points as i. Draw a perpendicular to the axis on which the k points are situated, passing through i. Take the reflection of point p against this perpendicular. Let the reflected point be p'. There can be two probable places where p' can lie:

p' lies to the right of all the k points. Clearly, this means that p is too far to the left from the k points and exists a point which is closer to all the k points. Hence, p is not an optimum 1-center.

p' lies inside the convex hull of the k points. Clearly, p' is a better choice for the 1-center, since if k_1 and k_2 are the leftmost points among the k points whose 1-center we want to find, and distance of p to k_1 is d, and distance between k_1 and k_2 is d', then $|pk_1| = d$ and $|pk_2| = d + d'$, but $|p'k_1| = d$ and $|p'k_2| = d - d'$. Using similar arguments we can show that p' is closer to all k points than p. From **lemma 2**, p' is a better contender than p to be the $1 - center$.

Lemma 4
Let p be a point outside the convex hull of a set of k points. Let f be the face of the polytope where p is incident. Let p' be the image of p against f. Consider a point q which forms another face f' and q is not a part of the face f. Then $|pq| > |pq'|$.

Proof: Drop a perpendicular from q to the line joining p and p'. Let the intersection point of the perpendicular and the line joining p and p' be r. Then there can be two cases:

Let r lie inside pp'. Then let $\angle qpr = \phi$ and $\angle qp'r = \theta$. Then $\tan\phi = qr/pr$ and $\tan\theta = qr/rp'$. Since pr must be greater than rp' (because p' is a reflection of p and the face is equidistant from p and p' and r must be inside the convex hull due to convexity), we get $\theta > \phi$. Now consider $\Delta pqp'$. Since $\theta > \phi$, we have $pq > p'q$.

Let r lie outside pp'. Then $\angle pp'q > \pi/2$ (Consider $\Delta p'qr$, since $\angle p'rq = \pi/2$, we have $\angle qp'r < \pi/2$, hence $\angle pp'q > \pi/2$), while $\angle qpp' < \pi/2$ (since $\angle prq = \pi/2$, and considering Δpqr). Now consider $\Delta pp'q$ and we are done.

3.1 Proof of Theorem 6: Center Lies Inside Convex Hull

The proof of the theorem follows by induction on the number of dimensions.

Base case : See **lemma 3**

Let the theorem be true till d dimensions. The proof requires to show that it is true for $d + 1$ dimensions. Consider k points in $d + 1$ dimensions. Let us now project these k points on d dimensions. By the strong induction assumption, we know that the 1-center lies inside the convex hull of these k points in d dimensions. Let this 1-center be p.

Now add back the $(d+1)^{\text{th}}$ dimension. Clearly, the position of p with respect to the previous d dimensions do not change; however, it has a new component in its position in the $(d+1)^{\text{th}}$ dimension. By **lemma 3**, we know that this component of p must lie between the two most extreme $(d+1)^{\text{th}}$ components of the k points. If p lies inside the convex polytope of the k points, we are done.

Let p not lie inside the convex hull. Then consider its reflection p' against the face f it is incident on. We know from **Lemma 4** that for points on the adjoining faces of f, and not a part of f, the distance from p' is lesser than that of p. For all other points lying on other faces on the visible side (from p) of the convex hull, a similar argument will provide similar results. For points on the non-visible side p' is definitely closer than p. From **Lemma 2**, we are done.

Once we have proven this, we now move on to find the 1-center or simply *center* of a set of flats or hyperplanes in $\mathbb{R}^d$

3.2 Center by descent

We try to determine the center, or 1-center of a set of flats or hyperplanes in $\mathbb{R}^d$. We use a descent algorithm to converge on the 1-center. We then attempt to find the 1-center for flats or hyperplanes using a random walk, but we show that it is not feasible. Our algorithm and proofs are for lines, however, it is scalable and can be used for flats and hyperplanes of any dimension.

We first look at some definitions, and the set of inputs that we will provide to the algorithm.

3.2.1 Definitions

L: The set of lines in $\mathbb{R}^2$

$C(p)$: The cost of point p with respect to the set of lines.

$d(p_1, p_2)$: The Euclidean distance between points p_1 and p_2

$D(p, l)$: The L_2 distance between the line l and the point p

$I(L)$: The set of intersection points of the lines in L

$CH(I)$: Convex hull of I

$Cr(p, r)$: The circle of radius r and center[1] at p

3.2.2 Inputs

L, the set of lines.

r, the parameter denoting the radius of a circle.

t, the number of divisions;

k, the cost parameter

```
**Algorithm 4** (Center by Descent)
1: $p \leftarrow$ A random point in $CH(I)$.

2: Construct $Cr(p, r)$.

3: Divide $Cr(p, r)$ into $t$ parts such that all the arcs formed by the division
are of equal length. Let these set of points defining the arcs be $P$

4: Find $p_1 \in P$, such that $C(p_1) \leq C(p_i)$, $p_i \in P$, $i \in [1, |P|]$. In other
words, $p_1$ is the point $\in P$ with the lowest cost. Let $p_2 \in P$ and $p_3 \in P$ be the
two neighbouring points of $p_1 \in P$.

5: if ( $C(p_1) < C(p) - k$ )
        $p \leftarrow p_1$
        return to Step 2
// else:

6: Divide arc $p_3 p_1 p_2$ into $t$ equal parts. Find the point $p'_1$ which has the most
reduction in cost. (Similar to step 4), let $p'_2$ and $p'_3$ be its neighboring points.

7: if ( $C(p'_1) < C(p) - k$ )
        $p \leftarrow p'_1$
        return to Step 2
// else:

8: Output : $p$.
```

[1]This center is not the 1-center ; this is the center of a circle

3.2.3 Analysis

Correctness

By our definition of the cost function, it is clearly a convex function. Therefore, there is only one local as well as global minima. Therefore, if $p_1 \in P$ is the point with the least cost, and $p_2 \in P$ and $p_3 \in P$ are the neighbouring points, then one of the two things might happen:

Either, $C(p_1) < C(p) - k$. In this case, we choose p_1 as the new center of the circle, and we move towards the optimum center.

Or, $C(p_1) \geq C(p) - k$. This means that there are again two sub possibilities here:

1. The cost difference between p and the optimal 1-center is less than k, and therefore, no point on the circle has a cost k lower than the cost for p. In this case, p is a good approximation of the optimal 1-center.

2. The other possibility is that when we chose the points in P, we neglected a point $x \notin P$. Due to convexity, we claim that if such points exist, then it must lie in the arc $p_3p_1p_2$, because since our function is convex and p_1 is of a lesser cost than both p_2 and p_3, hence either p_1 is the local (as well as global) minima or the minima lies in the arc $p_3p_1p_2$.

. To reduce our error probability, we further divide the arc $p_3p_1p_2$ further into t equal parts, just like we did before. We again look for the point with minimum cost and compare it with the current cost. If still, we find that the reduction in cost is less than k, we return the point p. Here, we can have the following analysis:

The maximum cost reduction can be due to a point that lies between $p'_2p'_1p'_3$. This is a length of $2\pi r/t^2$. However, given that the lines are placed randomly, and therefore, the location of the point with minimum cost is random, the probability that the optimal point lies in this area is (Let E be the event that the optimal point lies in this arc)

$$Pr(E) = 1/t^2$$

Hence with probability $1 - (1/t^2)$, we can say that the optimal center lies within a distance of k from the point p.

Note that t need not be a large constant. If $t = 10$, the algorithm provides the correct output with 99 percent accuracy.

Also note that the success of our algorithm is independent of both r and k. However, r and k are not independent of each other; a bad value of k can fail the algorithm decisively.

Analysis of Convergence

Theorem 7
The algorithm "Center by Descent" converges to the optimal 1*-center within an error range of k for appropriate values of k, as the number of trials $i \to \infty$. By appropriate values, we mean that k is upper bounded by $|L|^2 r/2$, where r is the radius of the circles used in the algorithm and $|L|$ is the number of lines in L.*

To analyse the convergence, we first claim that the cost function $C(x)$ is Lipschitz. Indeed if p and q are two points such that they are r distance apart, then $|C(p) - C(q)| \leq |L|r$, where for the specific problem, $|L|$ is a constant.

We also claim that the Cost function is not differentiable. Indeed, this can be easily seen; if there is only one line $l \in \mathbb{R}^2$ and a point moves towards in a line perpendicular to l, then the cost function follows a $C(x) = |x|$ curve.

Our third claim is that the convergence algorithm is exactly like a gradient descent algorithm except that the cost function is non differentiable. It is easy to see how: in each step of a gradient descent algorithm, the algorithm must move the function towards the global minimum, i.e., the after each iteration, the value of the function must decrease. This is exactly how our algorithm is. In every iteration, we move the point by a distance of r only in the direction where there is a decrease of cost by at least k [2]. Our algorithm advances with a fixed step of step size r.

The implication of the previous statement is that we will be unable to use gradients in our analysis, we instead use subgradients, defined by Shor, that is exactly like a gradient (and hence also unique) when the function is differentiable, but where it is not, it gives us all the tangent lower bounding the function at that point.

With these observations[3], we are now ready to do an analysis of convergence of our algorithm. Our algorithm iterates in the following manner:

$$x^{(i+1)} = x^i - \alpha_i g^i$$

Here g^i is one of the many subgradients at point t. Note that in general where

[2]Note the difference with Stochastic gradient descent. If a random walk algorithm provided us with any kind of guarantee to reach the center, it would be equivalent to SGD

[3]Note that the Cost function is also not strongly convex. However, we do not need this observation in this analysis.

the function is differentiable, the subgradient is unique and equal to the gradient. By the definition of our algorithm, we choose the direction where the cost reduction is maximum, with a probability of at least $(t-1)/t$ and at most $(t^2-1)/t^2$. The value of $C(x)$ that we get at this best direction at the i^{th} iteration, we call it $C^i_{best}(x)$. For our algorithm, the choice of α_t is constant and independent of i, it is equal to r.

For such convergence with fixed steps, we have a standard result that says that if $C^*(x)$ is the optimal minimum cost, then we have a guarantee that

$$\lim_{i\to\infty} C^i_{best}(x) - C^*(x) < k$$

which is what we want. Therefore, we know that the algorithm converges with surety. If the step size was not constant, it would be possible for an algorithm to exactly hit the minima without an error range.

However, we are also interested in the rate of convergence, i.e, how close x is to the argmin after t iterations. Note that our function is Lipschitz, which means that the subgradients are bounded by some constant. Let G be this upper bound on the subgradient. Then we have another result that says:

$$C^k_{best}(x) - C^*(x) \leq (d(x^{(1)}, x^*)^2 + G^2r^2i)/(2ri)$$

Note that as $t \to \infty$, the difference between the $C^k_{best}(x)$ and C^* converges to $G^2r/2$. This shows us that we converge decisively as t increases.

The proof of this convergence is omitted here, and can be found in many texts; Bubeck [2015] and Stephen Boyd [2003] covers the proof in details, as well as provides some insight into subgradients.

Relationship between k and r

We have earlier said that there needs to be some relationship between k and r. Arbitrary values may not converge at all; or more precisely, may have such a large convergence bound that the algorithm gets useless. In other cases, it might happen that the radius is so small that it does not even find any point with decrease in cost equal to or more than k, and give us a wrong result. We need to look at all scenarios while determining the relationship.

Note that our algorithm converges to within $G^2r/2$ of the optimum minima; but we need this to be within k. Therefore, we have

$$G^2r/2 = k$$

Now, in the previous section, we found a Lipschitz bound for the Cost function $|L|r$, if two points are r distance away. This bound is tight, if we have parallel lines and a point approaching towards them, such that there is a hyperplane that separates all the lines and the point, then moving a distance of r towards the optimum does indeed decrease the cost by $|L|r$.

This also implies that all subgradients are bounded by $|L|$. Therefore, we have

$$|L|^2 r/2 = k$$

However, we can not have a too large value for k, it might be the case that when we have the circle, the value of k is too large for any point on the circle to actually have that amount of decrease. Therefore, we merely have an upper bound on k, which is

$$k \leq |L|^2 r/2$$

Do we have a lower bound on k? We need not have one. The smaller the value of k, the more probable it is that within a radius of r, a point with a decrease of k is found. Note that this will effect the rate of convergence; however, we know that it will ultimately converge. It is best to choose k as a hyperparameter with an upper bound, with multiple "guesses" to what the correct value of k should be.

3.2.4 Discussion and Scalability

The algorithm can be easily extended for lines or flats in $\mathbb{R}^n$, where we take the hypersphere $\mathbb{S}^n$ instead of a circle. The correctness of this algorithm can be easily proven like above, the convergence rate will not change either.

If we are dealing with hyperplanes in $\mathbb{R}^n$, we know that there is a convex polytope formed by the intersection points, where the center must lie. In this case, we choose $x^(1)$, the initial point to be a point inside this convex polytope. However, if we are dealing with flats, there are no such guarantees. This implies that the starting point can be anywhere, and therefore, the convergence may take a longer time.

3.2.5 Time Complexity

Each step of the algorithm must check the Cost for $O(2t)$ points, until the convergence occurs. Each operation costs $O(|L|)$ time. Hence, if there are s iterations, the time complexity is $O(|L|t)$.

3.3 A random walk attempt to find the center

Can a random walk using concepts from the above algorithm find the center? As we will see, it is not, in general, feasible.

3.3.1 Modifications to algorithm 4

Let us change our previous **deterministic** algorithm as follows: As earlier, we divide the circle in sectors, but for simplicity we discard the fine tuning that we have done in the previous algorithm, in case a optimum point was not found in the first set of points on the circle. In this case, if we do not find an optimum point, we return the center of the circle.

In order to not compromise on the probability, we increase the number of sectors; for demonstration, let us take the number of sectors to be 40. Then, if no optimal point is found on the circle, we declare the center of the circle is at least k cost close to the optimal center, with a probability of 0.95 [4]

3.3.2 The Random Walk

Suppose, if we took only the optimal paths, we would require n circles (steps) to reach the optimal point. Now clearly, we leave the realm of deterministic algorithm and use a random walk approach; the computer chooses any of the 40 points on the circles with uniform probability (of 1/40). We compute the cost of this point, then build another circle and repeat the process.

What we want is that after repeating this process for some x times, we claim that the point with the lowest cost is the k-optimal center. The problem then becomes to find an expected value of x, and to check if we are bound to get to get the k-optimal center ultimately, regardless of other factors.

3.3.3 Probability of reaching the center from n steps away

Let P_n represent this probability. Then we choose the optimal point on the circle with probability 1/40 and the rest 39/40 is the probability of choosing the non optimal points. This brings us to the recurrence relation:

$P_n = 1/40P_{n-1} + 39/40P_{n+1}$, with the boundary conditions :
$P_0 = 1$ and $P_\infty = 0$ [5]

Solving this relation, we get the corresponding probability P_n

$P_n = (1/39)^n$

[4]Note that this is not the random walk algorithm, that comes in the next section
[5]Technically we can write P_b, $b \to \infty$

Note that not all optimal points are taking us away from the center. The optimal point is surely the best, but there are other points that can bring us closer to the center as well. Therefore, it is better for us to write $P_n > (1/39)^n$

3.3.4 Expected number of steps to reach the center

Let X_n be the expected number of steps to reach the center when the point is n steps away. Then, we have the following recurrence relation (The 1 is added for the current step)[6]

$X_n = 1 + (1/40)X_{n-1} + (39/40)X_{n+1}$ with the boundary conditions
$X_0 = 0$, $X_\infty = \infty$

Solving, we get $X \to \infty$! (One of the co-efficients get to infinity on providing the boundary condition)

An issue maybe raised here, that although not the optimum, but if the random walk chooses points that bring us closer to the center, would the expected amount of steps be still infinity? Indeed it will still be infinity. Let us consider that the circle is simply divided into two parts, one half of the circle closer to the center, and the other half away from it. Let any point chosen in the optimum half be equivalent to moving one step closer to the center, and any point chosen on the other half be equivalent to a negative step. Note that no point in the non-optimum half brings us closer to the center.

With this setting, our probability of going one step ahead towards the center is equal to the probability of going away one step from the center. However, even here, we find that the expected time goes to infinity, using similar recurrence relations.

However, we have a crucial piece of information, the center must lie in the convex hull of the intersection points. If we modify our boundary conditions, that once the walk moves outside the convex hull, we stop the random walk, then technically we get a finite expected number of steps. However this expected number is the time before the random walk stops, which may be because the point moves outside the convex hull.

If d is the largest distance (in number of steps) to the boundary of the convex hull from the center, then we have a modified condition:

$X_d = 0$

[6]The following result is derived from total Expectation and linearity of expectation

Using these modified boundary conditions: $X_n = O(n)$[7]

Note again, that this is not an indication of the expected amount of time taken to reach the center, rather for the walk to end.

Note here that d is the largest distance from the center to the convex hull, so the actual expected time of the ending of the walk should be less than the value calculated here. This also solves another problem, every step that is not optimum is a step towards the convex hull, so all of them count as failures.

3.3.5 Scaling for higher dimensions

Flats in $\mathbb{R}^d$ may not necessarily intersect, therefore, wherever the boundary of convex hull was used, they will not hold any longer. However, all other calculations hold. Everything should also hold fine for hyperplanes in $\mathbb{R}^d$, because hyperplanes will necessarily intersect and give us a convex polytope.

3.4 Coreset For 1-center for flats and hyperplanes in $\mathbb{R}^d$

We are now prepared for an algorithm for a coreset in $\mathbb{R}^d$ for $1-center$ only. As usual, we provide an algorithm for lines in $\mathbb{R}^d$, and present a scalable algorithm, that can be extended to flats in $\mathbb{R}^d$

Algorithm 5 (Coresets for lines for $1 - center$)

```
Input: L, m
1. p ← Algorithm 4 $(L, r, t, k)$
2. $P \leftarrow \phi$, $O \leftarrow \phi$
3. for every line $l \in L$
        Draw perpendicular from p to l . Let $o_i$ be the point where the perpendicular from p meets $l_i$
        $P \leftarrow P \cup \{o_i\}$
4. B ←Algorithm 2 $(P, 1)$
5. O ←Algorithm 3 $(P, 1, B, m)$
6. for each point o in O
        Output ← Output∪ line corresponding to o
7. Return Output
```

[7]The solution is exactly analogous to the unfair Gambler's ruin.

We clearly see what is being done here. We choose certain appropriate points from each of the lines, then proceed to build the coreset for the points, following which we map the points in the coreset back to the lines they came from.

The algorithm is clearly scalable in dimensions. For d dimensions, and $k-flats$, one simply needs to get the appropriate points for each of the flats.

3.4.1 Analysis

Correctness

Theorem 8
The algorithm "Coresets for lines for 1*-mean" returns, for a defined value of ϵ, the ϵ-coreset for the set of lines L in $\mathbb{R}^d$. Furthermore, the time complexity of the algorithm exceeds that of the time complexity of the algorithm to determine ϵ-coresets for points by only the time complexity to determine the 1-center of the set of lines in L.*

It is evident that when we choose a point perpendicular to the line l from the center p (let it be o), the distance between l and p is the same as the distance between o and p.

Note that when the coreset is built, the approximate center is away from the actual center, so the distance from the points change, and hence in the line set, the distance from the lines change too. However, for all lines, this change in distance from each line must be bounded by $|r|$, if r is the amount the approximate center is shifted from the optimum center. The change in cost in the line set must therefore, be less than or equal to the change in cost for the corresponding point set. Therefore, the property of ϵ-coresets hold.

A problem may arise, because the center we developed for the line set is not the exact optimal, rather within a range of k cost from the optimal. Therefore, we know that the coreset we built can have at most $\epsilon Cost(L, C) + k$ error. This can be easily overcome, by using an $\epsilon' < \epsilon$-coreset for the points, to get an ϵ-coreset for the lines.

Time Complexity and Overhead

The only overhead for building the coreset is getting to the approximate center of the set of the lines. The rest is exactly same as building the coreset for points. The excess time complexity too, is only due to this overhead.

3.4.2 Discussion and Scalability

Clearly, since Algorithm 4 is scalable in the number of dimensions, and can be easily modified for both hyperplanes and flats, and since the argument of getting the points of intersection from the point to the flat being the set of points whose ϵ-coreset gives us the ϵ-coreset of the set of flats holds, we can assert that the algorithm is scalable in dimensions.

Similar to Algorithm 4, Algorithm 5 has a better time complexity if the coreset is built for hyperplanes rather than flats, since Algorithm 4 converges faster for hyperplanes.

Future Work

While we have been able to build the 1-center coreset for flats and hyperplanes in any dimension, the problem of generalizing this to K-means still remain. The k-center problem is NP-Hard; a divide and conquer heuristic approach by us to get the approximate k-center so that we could use a similar algorithm to get the coreset did not yield any significant result.

We were looking at using the Frank-Wolfe algorithm for this purpose; however, we could not make any progress.

While the dissertation is being submitted at this stage of the work, we will continue to work on the actual problem of generalizing the problem of coresets to any number of clusters.

Bibliography

Mihai Badoiu and Kenneth L. Clarkson. Optimal core-sets for balls. *Comput. Geom.*, 40(1):14–22, 2008. doi: 10.1016/j.comgeo.2007.04.002. URL `https://doi.org/10.1016/j.comgeo.2007.04.002`.

Michael Langberg and Leonard J. Schulman. Universal epsilon-approximators for integrals. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 598–607. SIAM, 2010. doi: 10.1137/1.9781611973075.50. URL `https://doi.org/10.1137/1.9781611973075.50`.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM, 2007. URL `http://dl.acm.org/citation.cfm?id=1283383.1283494`.

David Haussler. Sphere packing numbers for subsets of the boolean n-cube with bounded vapnik-chervonenkis dimension. *J. Comb. Theory, Ser. A*, 69(2): 217–232, 1995.

Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *J. Comput. Syst. Sci.*, 62(3):516–527, 2001. doi: 10.1006/jcss.2000.1741. URL `https://doi.org/10.1006/jcss.2000.1741`.

Andreas Krause Olivier Bachem, Mario Lucic. Practical coreset constructions for machine learning. 2017. URL `https://arxiv.org/abs/1703.06476`.

Yair Marom and Dan Feldman. k-means clustering of lines for big data. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 12797–12806, 2019. URL `http://papers.nips.cc/paper/9442-k-means-clustering-of-lines-for-big-data`.

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. *CoRR*, abs/1106.1379, 2011. URL `http://arxiv.org/abs/1106.1379`.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3-4):231–357, 2015. doi: 10.1561/2200000050. URL `https://doi.org/10.1561/2200000050`.

Almir Mutapic Stephen Boyd, Lin Xiao. Subgradient methods, 2003. URL `https://web.stanford.edu/class/ee392o/subgrad_method.pdf`.