Understanding the K-Medians Problem

Christopher Whelan, Greg Harrell, and Jin Wang
Department of Mathematics and Computer Science
Valdosta State University, Valdosta, Georgia 31698, USA

Abstract *- In this study, the general ideas surrounding the k-medians problem are discussed. This involves a look into what k-medians attempts to solve and how it goes about doing so. We take a look at why k-medians is used as opposed to its k-means counterpart, specifically how its robustness enables it to be far more resistant to outliers. We then discuss the areas of study that are prevalent in the realm of the k-medians problem. Finally, we view an approach to the problem that has decreased its time complexity by instead performing the k-medians algorithm on small coresets representative of the data set.*

Keywords: K-medians; K-means; clustering

1. Introduction

The clustering problem is one well researched in the computer field due to its incredible variety of applications, be it unsupervised learning, geographic positioning, classifying, data mining, or other. K-medians and k-means are two widely popular approaches to performing this clustering task. Both involve finding k cluster centers for which the sum of the distance between a center and all points in that cluster is minimized. Where the two methods differ is in what they consider the "center" of the cluster to be. As one could infer by their names, k-means uses the mean (minimizing the 2-norm distances) while k-medians uses to median (minimizing the 1-norm distance). This paper will focus on the k-medians variation.

2. K-Medians

As mentioned above, the k-medians approach to clustering data attempts to minimize the 1-norm distances between each point and its closest cluster center. This minimization of distances is obtained by setting the center of each cluster to be the median of all points in that cluster. This section discusses why this is such a powerful method of clustering data, shows why it is a good alternative to the k-mean approach, and provides a brief overview of the k-medians algorithm to procure a better knowledge base concerning this topic.

2.1 Benefits over K-Means

The k-means problem was conceived far before the k-medians problem. In fact, k-medians is simply a variant of k-means as we know it. Why would k-medians be used, then, instead of a more studied and further refined method of locating k cluster centers? K-medians owes its use to robustness of the median as a statistic [1]. The mean is a measurement that is highly vulnerable to outliers. Even just one drastic outlier can pull the value of the mean away from the majority of the data set, which can be a high concern when operating on very large data sets. The median, on the other hand, is a statistic incredibly resistant to outliers, for in order to deter the median away from the bulk of the information, it requires at least 50% of the data to be contaminated [1].

Through its use of the median as the determining factor in placement of cluster centers, k-medians is able to assimilate the robustness that the median provides. Implementing variations of the k-medians method can further reduce the minimal shifts resulting from the presence of one of more outliers. Such variations include k-medians with outliers, in which points that exhibit attributes common with outliers are handle in a manner in which their distances will have a smaller effect on the positioning of the center, and robust k-medians with m outliers, which attempts to discard up to m points that the algorithm determines to be outliers.

2.2 K-Medians Algorithm

Given a set of points, the k-medians algorithm attempts to create k disjoint cluster that minimize the following equation. This means that the center of each cluster center minimizes this objective function [2].

$$Q\left(\{\pi_j\}_{j=1}^{K}\right) = \sum_{j=1}^{K} \sum_{x \in \pi_j} \left\| x - c_j \right\|_1$$

This minimization is defined by the geometric median.

$$\underset{y \in R^n}{\arg\min} \sum_{i=1}^{m} \|x_i - y\|_2$$

In order to begin this process, k initialization points must be selected as the cluster centers. The logic code below is then performed:

K-Medians Algorithm

```
Q = infinity
do
    for point in dataset
        min = infinity
        index = 0
        for i in k
            dist = distance(point, center[i])
            if dist < min
                min = dist
                index = i
        disjoint-sets.add(index, point)
    for i in k
        center[i] = median(disjoint-set.get(i))
    sum = 0
    for i in k
        for point in disjoint-set.get(i)
            sum = sum + distance(point, center[i])
    oldQ = Q
    Q = sum
while (oldQ - Q) > eps
```

The above code follows these steps

1. Assign each point in the data set to its closest center. The points assigned to the same center are then said to be in the same cluster, therefore they are added to the same disjoint-set. Because each point has been assigned to its closest center, the value of Q will not increase.
2. With the new disjoint-sets as the clusters, calculate their median to determine the updated value of that cluster's center. Because the center is a minimization of 1-norm distances, Q cannot increase as a result of this step.
3. Sum all distances between each point and its respective cluster center. This is the new value for Q.
4. If the improvements made by this iteration are less than a previously determined epsilon, quit. Otherwise, repeat the process.

Because both steps 1 and 2 can only decrease the overall value of Q, k-medians is a local optimization algorithm minimizing the sum of the 1-norm distances throughout the dataset [3].

3. Areas of Study

This section will cover a handful of topics that go into the k-medians algorithm that were abstracted away for simplicity. These topics include: median calculation, distance specifications, determining a value for k, and the initialization process for k cluster centers.

These topics have in no way been studied to completion, and many of the problems surrounding these topics have yet to be solved [3]. These subsections that follow will merely introduce these areas and current methods on how to approach them.

3.1 Finding the Median

K-medians uses the median as the statistic to determine the center of each cluster. It has been proven, however, that there exists no closed form that can determine the geometric median in every dimension. Because of this, methods of finding the median have turned to a more heuristic approach. We will now take a look at two of these methods, one that uses a simple simulated annealing algorithm, the other the more commonly implemented Weiszfeld's algorithm in order to understand the ideas surrounding median calculation [4].

3.1.1 Simulated Annealing

The simulated annealing approach is this general method

Simulated Annealing

```
step = arbitrary value
median = mean(points)
min = sum_distances(points, median)
improved = false
while step > eps
    for direction in directions
        temp_median = median, + (direction*step)
        d = sum_distances(points, temp_median)
        if d < min
            min = d
            median = temp_median
            improved = true
            break
    if !improved
    step = step / 2
```

In the above algorithm, "directions" is a list of all of the unit vectors of the dimension in the points are found in. The addition and multiplication shown are, by extension, vector addition and scalar multiplication respectively.

This method follows these steps

1. Initialize an arbitrarily large value for steps. Alternatively, this value could correspond to some statistical measurement of the points, be it variance, standard, deviation, median absolute deviation, or something else of the sort.

2. Initialize median to be the mean of the data set. This is just a starting approximation.
3. Check to see if moving the median in any direction by the value of step provides a better approximation for the median. If it does, update the median and continue.
4. If none of the movements improve our median, half the step size.
5. Stop once step has decreased below a predetermined value of epsilon

This method of approximation, while slow due to the constant calculation of the sum of distances (2d where d is the dimension in which the problem resides), will find a good median approximation with accuracy based on epsilon. The convex nature of the median guarantees that this method will not be trapped at a local minimum.

3.1.2 Weiszfeld's Algorithm

Weiszfeld's algorithm is as follows

Weiszfeld's Algorithm

```
median = mean(points)
for j, k
    m1 = {0}
    m2 = 0
    for point in points
        dist = distance(point, median)
        for i in m1.length
            m1[i] = m1[i] + point[i]
            m2 = m2 + (1/dist)
    median = m1/m2
```

1. Initialize median to mean. This is just an initial approximation.
2. Set up a variable to represent the sum of the points divided by the distance from that specific point to the approximated median (m1).
3. Initialize a variable to keep track of the sum of the inverted distances from each point to the approximated median (m2).
4. For each point, add the value of the point divided by the distance between it and the current median to m1. Add the inverse of the distance between it and the current median to m2.
5. The new approximation for the median is equal to the scalar division of m1 divided by m2

This method takes advantage of the alternate definition of the median [4]. We can see the similarities between this and this representation of Weiszeld's algorithm.

3.2 Different Forms of Distance

While Euclidean distance is the measure most commonly used when the k-medians algorithm is applied to a k-clusters problem, it is not always the appropriate choice to correctly model what the k-clustering is attempting to achieve. Many times, models of real world scenarios require certain restraints in distance measurement. One such deterrent may be the existence of obstacles prohibiting straight-line travel from one point to the next. In a situation such as this, Euclidean measurements are simply far too inaccurate, and other forms of distance, such as Manhattan distance, must be considered instead.

Manhattan distance is a measurement based on a grid system in which the points in question are placed. The concept is that in order to move from start to end point, one of four directions must be chosen for the point to advance: up, down, left, or right. Each decision will move the start point one unit in the chosen direction. The Manhattan distance is determined by the number of decisions required for the start point to reach the end point [5]. This method of distance can be extended to account for obstacles that may stand in between the two points. Luckily, this method is well suited for k-medians. The goal of k-medians is to minimizing the absolute deviations, which is in fact equivalent to the Manhattan distance.

These two measurements do not cover all scenarios, however. When using clustering to organize more categorical data, minimizing divergences, such as Bregman and Kullback-Leibler divergences [5], can be the desired outcome. Because of these different ways to determine the quantity of weight to be assigned to the edge between two points, the implementation of the k-medians algorithm must be shaped to appropriately handle the type of distance measurement best suited to model the problem at hand.

3.3 Selecting K

K-Medians clustering aims to separate a given data set into k clusters. The value of k, while important in the structure of the resulting k-cluster model, is given to the algorithm as input from the user. This value of k may be chosen based on assumptions of the data set, prior experience with like data set, or prior knowledge on the contents of the data set [6]. K, in this sense, is a fixed value that may or may not produce desired clustering, even if that clustering minimized the objective function. There do exist, however, approaches that can help approximate a better value for k. These approaches are discussed in this section.

One method takes advantage of the fact that in center based clustering, the clusters adhere to a unimodal distribution. In this case, that will be the Normal, or Gaussian, distribution [6]. The idea is that k will be initializing to an arbitrarily small value, for simplicity we will say 1. Are clusters are then analyzed for their distribution. If a cluster is shown to have a unimodal normal distribution, the cluster remains unchanged. If not, the current center is replaced with two centers. After all clusters are analyzed, the clusters are recalculated using k-medians, and the process continues. The stopping point is reached when no cluster centers have to be broken into two [6].

Another approach find the value of k in a similar manner, but rather than splitting the centers until the desired distribution is found for each cluster, this method is supplied a range of potential k values and determines the best choice among the given options. It does so by creating a model scoring system based on Bayesian Information Criterion (BIC) [7]. This selection criterion provides a way to compare the distributions of the clusters resulting from a certain k selection to a different k selection. This method increases the value for k in a similar manner to that of the previous approach in that it splits a center into two distinct points. The center is chosen by comparing which split will most benefit the BIC score [7].

3.4 Initializing Centers

The k-medians approach to clustering locates a local optimal solution that minimizes the 1-norm distances from each point to its respective cluster center. It is important to note that this finds a local, rather than global, optimum. Because of this, k-medians is very sensitive to the initialization points of its k centers, each center having the tendency to remain roughly in the same cluster in which it is first placed [8]. Different ideas have therefore been proposed in order to find better initial placement for the k centers in hopes that they will converge to the global optimum. A handful of these propositions are as follows:

- **Random Initialization**- K points are generated at random. The k centers are initialized to these points.
- **Density Analysis**- K points are selected from viewing the distribution of the data set and isolating high density areas.
- **Single Dimension Subsets**- Column vectors are looked at independently in order to select the dimension with the largest variance. The points in this dimension are then divided into k subsets, and the centers are initialized by the median values of these subsets.
- **Diagonal Initialization**- The data set is divided in to k rows and k columns, creating a 2d grid. The cells on the diagonals of this grid are then weighted by their density. The k centers are then randomly selected from these cells, taking their weight into account for the selection.
- **Sampling**- Samples are taken from the data set. The k-medians algorithm is then applied to each sample using random initialization. The k centers are then randomly selected from the solutions generated [9].

These are only a few of the proposed concepts to generate better initialization points. While many of them show promise theoretically, however, their performance is often worse than or roughly equivalent to those result generated by random initialization. There is currently no method of initialization that will produce centers that will always converge to the global optimum.

4. References

[1] D. Feldman and L. J. Schulman (2012) Data reduction for weighted and outlier-resistant clustering Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (pp. 1342-1354)

[2] K.Chen (2006) On K-Median Clustering in High Dimensions Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithm (pp.1177-1185)

[3] B. Anderson, D. Gross, D. Musicant, A. Ritz, T. Smith, L. Steinberg (2006) Adapting K-Medians to Generate Normalized Cluster Centers Proceedings of the Sixth SIAM International Conference on Data Mining (pp.165-175)

[4] G. Hamerly and C. Elkan (2003) Learning the K in K-Means NIPS

[5] M. Ackermann, J. Blomer, C. Sohler (2010) Clustering for Metric and Non-Metric Distance Measures ACM Transactions on Algorithms 6:4

[6] G. Hamerly and C. Elkan (2003) Learning the K in K-Means NIPS

[7] D. Pelleg and A. Moore (2000) X-means: Extending K-means with Efficient Estimation of the Number of Clusters In Proceedings of the 17th International Conf. on Machine Learning (pp. 727-734)

[8] S. Bubeck, M. Meila, U. Von Luxembourg (2012) How the Initialization Affects the Stability of the K-Means Algorithm ESAIM: Probability and Statistics (pp.436-452)

[9] Mohammad F. Eltibi Wesam M. Ashour (2011) Initializing K-Means Clustering Algorithm using Statistical Information International Journal of Computer Applications (pp.51-55)