Adventures in Ultra-Small-Space Clustering

Adam Meyerson [1]   Liadan O'Callaghan [2]
Serge Plotkin [3]

December 2002
CMU-CS-02-193

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We give a sampling-based algorithm for the k–Median problem, with running time $O(\frac{k^2}{\epsilon}\log\frac{k}{\epsilon})$, where k is the desired cluster number and ϵ is a confidence parameter. This is the first k–Median algorithm with fully polynomial running time that is independent of n, the size of the data set. It gives a solution that is, with high probability, an $O(1)$ approximation if each cluster in the optimal solution has $\Omega(\frac{nk}{\epsilon})$ points. We give near-matching lower bounds showing that this assumption about cluster size is necessary. We also present a related algorithm for finding a clustering that excludes a small number of outliers.

[1] Aladdin Project, Carnegie-Mellon University. Research supported by NSF grant CCR-0122581 and ARO grant DAAG55-98-1-0170. Email: `adam@cs.cmu.edu`

[2] Department of Computer Science, Stanford University. Research supported by NSF grants IIS-9811904, EIA-0137761, and IIS-0118173, an NSF Graduate Fellowship and an ARCS Fellowship. Email: `loc@cs.stanford.edu`

[3] Department of Computer Science, Stanford University. Research supported by ARO grant DAAG55-98-1-0170 and ONR grant N00014-98-1-0589. Email: `plotkin@theory.stanford.edu`

Keywords: algorithms, approximation, clustering, sampling, k-median, facility location

1 Introduction

Clustering, or grouping data into representative groups, can often make the manipulation of large data sets simpler. For example, in a large customer database a cluster might represent customers with similar characteristics. In a vision application, a cluster might be the group of pixels that form a single object in the viewed image. In the context of the web, a cluster could be a set of web pages with similar content. In census data a cluster might represent a population center. In each of these applications, computing a clustering makes the data easier to manipulate, and easier for a human to parse. Clustering algorithms have been the subject of a great deal of research [4, 5, 9, 10, 11, 15, 19, 22, 23].

Data sets in the applications mentioned are growing at an exponential rate. The web is the most striking example of this expansion, but exponential growth has been observed in many other domains as well. As the data sets grow, the need for fast clustering algorithms becomes more and more apparent. Most of the theoretical work in clustering has focused on producing polynomial-time approximation algorithms for NP-Hard clustering problems. While these algorithms work very well for moderately sized data sets, the pattern of exponential growth suggests that even a *linear*-time algorithm may soon be too slow in practice, and so we consider the possibility of *sublinear*-time clustering algorithms that maintain provable approximation bounds. We cannot read the entire data set within such a running time, and so we must make several reasonable assumptions about the nature of the data (essentially, that the best clustering cannot be greatly influenced by the presence of a relatively small number of points). We show that these assumptions are necessary and that our running times are within a polylogarithmic factor of the best possible.

First, we must formally define the notion of clustering. Given a data set, a clustering of the data is defined as a subdivision of the data into several large groups. The quality of a clustering depends upon the similarity of the elements within a single group, as well as the non-similarity of elements in distinct groups. Many different measures have been proposed to quantify the quality of clustering. This paper will focus on k–Median, one of the most popular and best-studied clustering measures.

An instance of k–Median consists of a data set N, a distance metric, and an integer k; we are to choose k members of N as "medians," and assign each member of N to its closest median. The assignment distance of a point $x \in N$ is the distance from x to the median to which it is assigned, and the k–Median objective function, which is to be minimized, is the sum of assignment distances. The *continuous* k–Median problem is the same except that medians can be arbitrary elements of the metric space in which N resides.

k–Median is NP-Hard, and in fact it is NP-Hard to approximate the best solution to within a factor of 1.4 [12]. Previous work has provided a number of constant approximations to the problem[6, 7, 13, 17]. In the geometric setting, Arora, Raghavan, and Rao [2] gave a PTAS for the k–Median problem in low dimensions. Charikar, Guha, Tardos, and Shmoys [7] gave the first constant-factor approximation for general metric spaces. Jain and Vazirani [17], gave a Primal-Dual approximation, which was subsequently improved by Charikar and Guha [6]. The best approximation known to the general problem is $3 + \epsilon$, given by Arya *et al.* [3].

All of these approximations have running times polynomial in the size of the input. In general the running time is $\Omega(n^2)$ to compute approximate k–Median on n points. This running time arises from the need to compute all pairwise distances, or to solve a linear program with variables for all pairs. Guha *et al.* [13] give a divide-and-conquer approach that improves the running time to $\tilde{O}(nk)$ with some loss in the approximation ratio. Indyk [16] gives a sampling-based bicriteria algorithm with running time $O(nk)$.

When the data set is very large, even linear time can be prohibitive. We therefore turn to sampling algorithms, which select only a subset of the data to use in computing a clustering. Mishra *et al.* [20] give a sampling-based algorithm that, with probability at least $1-\delta$, produces a k–Median clustering of cost at most $2\alpha\text{OPT}+\epsilon$, where OPT is the optimum k–Median cost, α is the approximation ratio guaranteed by the k–Median approximation algorithm used as a subroutine, and ϵ is an input parameter. The running time depends on $(M/\epsilon)^2$, where M is the diameter of the data set, and on $\log(n/\delta)$. In the special case of Euclidean space, the dependence of the running time on $\log(n/\delta)$ is avoided, and the clustering cost is at most $\alpha\text{OPT}+\epsilon$. Alon *et al.* [1] present a sampling-based property-testing algorithm for the k–Center problem, in which the aim is to minimize the maximum assignment distance, rather than the sum of assignment distances. Sampling has also been used successfully in the applied community for various clustering problems [14, 18, 21].

We immediately recognize several challenges that will arise if sampling is to be used. First, we must be able to sample the data uniformly at random. If the selection of the sample were adversarial, we could "miss" significant portions of the data set and thereby have no chance of producing a good clustering. Second, it is possible for a small number of data points to have a very significant effect on the clustering. For example, suppose there is a very small (say $o(\sqrt{n})$) set S of points that are extremely far from all the other points. If any of these points are assigned to groups containing points *outside* of S, the clustering quality will be poor. Unfortunately, a small sample will be very unlikely to include *any* points from this set, and so it will be impossible to produce a good solution for this instance unless we examine more points. Third, this approach precludes outputting an explicit assignment of each point to a group, since performing this assignment will require at least linear time.

In order to deal with these problems, we will make several assumptions. First, we will assume that we can select a data point chosen uniformly at random in constant time. We can think of the data as being stored in some random-access memory device, so that we can select a random location and access that data point in effectively constant time. We observe that if we do not have random access, and the data are ordered adversarially on a device that must be read in-order, then the data are effectively streaming and the best we can hope for is linear time. Second, we assume that each of the clusters in the *optimum* solution, against which we compare our clustering, is large. The "average" size of a cluster would be n/k, and we will assume each cluster to have $\Omega(n/k)$ size. Equally well, we can allow our solution to disregard a small fraction of "outlying" points. Third, we will output k group medians, which are "representative" of the data set. We consider this output sufficient without assigning each point explicitly to its nearest median (indeed, if the data set is large, it is probably better to output the median set rather than an explicit assignment of every point).

In general, the goal of our algorithms is to produce several broad classes that are representative of most data points, and clustering a small number of outliers is not as important. Often the data itself has "noise" due to observational or recording errors; this noise may lead to small numbers of extreme outliers which should in fact be disregarded by the clustering solution.

Under the above assumptions, our algorithm will produce a set of medians in $O(\frac{k^2}{\epsilon}\log k)$ time. Here k is the target number of medians and ϵ is the parameter of our second assumption: we assume each optimum cluster contains at least $\frac{\epsilon n}{k}$ data points (or equivalently that we can disregard ϵn points). Observe that this running time is *independent* of n, the number of data points. Our algorithm guarantees that, with high probability, the medians we produce are within $O(1)$ of the optimum medians for which each cluster contains at least $\frac{\epsilon n}{k}$ points. We give near-matching lower bounds relating the size of the sample to the minimum cluster size requirement such that a constant approximation can be produced. Our methods also provide a technique for *evaluating* candidate clusterings in sublinear time, under the same assumptions about the nature of the data set.

2 Overview of Our Results

Our algorithm consists of two steps. In the first step, described in section 4, we generate several candidate median sets. To generate each set, we draw a uniform sample from the data and compute an approximately-best clustering for the sample using the algorithm of Arya *et al.* [3]. We show that, with constant probability, the cost of a candidate set is in fact within $O(1)$ of that of the best possible set of medians. By increasing the constants in the approximation factor and sample size, we could produce an algorithm that would with high probability produce a good clustering; the required increases would be unacceptably large, however. We can obtain better results by repeatedly sampling and clustering, and taking the best of our candidate median sets.

Determining the best set of candidate medians is a nontrivial problem. The straightforward approach to testing would take linear time. Thus, the second part of our algorithm is a method for determining the best of several candidate clusterings. We show, given a number of candidates, that we are able to select one whose cost is within $O(1)$ of the cost of the best candidate. This process can be performed, again using sampling, in time independent of n. The basic technique is to build a number of random samplings and measure the quality of each set of candidate medians on each random sample. We will return the set of medians that have the best observed quality (lowest minimum value over all samples). This evaluation of candidates is discussed in section 5.

Both steps of our algorithm require that the clusters in some optimal solution be large, and that we can select a data point uniformly from the input in constant time.

Combining the steps outlined above gives us an algorithm that produces constant approximations to the best k–Median solution, in running time independent of n, the size of the data set (under certain assumptions about the nature of the input). We present this overall result in section 6, and near-matching lower bounds in section 8.

Some data sets contain "noise" or "outliers" that make the data inherently difficult to cluster, in the following sense: the best possible k–Median solution is "much more expensive" than the best solution that excludes a small number of points (i.e., does not group them and does not pay their distance costs). Using a natural extension of the algorithms already described, we can produce a constant approximation to the best clustering that excludes ϵn outliers, provided our algorithm is allowed to exclude a slightly higher $(O(1)\epsilon n)$ number of outliers. The approach used is identical, except that we need to use the algorithm of Charikar *et al.* [8] instead of that of Arya *et al.* [3] to cluster the sample. We discuss this extension in section 7.

3 Notation

Before giving the proof of approximation factor of the algorithm, we first introduce some notation.

Definition 1 *N is the data set, with $n = |N|$.*

Definition 2 *S is a sample, with $s = |S|$.*

Definition 3 *k is the desired number of medians; K^* represents an optimum median set, and K will represent a candidate set of medians. Here $|K| = |K^*| = k$.*

Definition 4 *ϵ is such that every optimum cluster contains at least $\frac{\epsilon n}{k}$ points.*

Definition 5 *$f^N(K)$ represents the cost of medians K as a clustering of the data set N. That is, $f^N(K) = \sum_{x \in N} d(x, C(x))$, where $d(\cdot, \cdot)$ is the distance function and $C(x)$ is the closest member of K to x. $f^S(K)$ is the quality of the same medians K when used to cluster the sample S.*

4 Generation of a Good Candidate Set

We will select a random sample S with $|S| = s = \Omega(\frac{k}{\epsilon} \log k)$. We solve the continuous k–Median problem to a g-approximation on this sample using the algorithm[1] of Arya *et al.* [3]. The value of g depends on the version of Arya's algorithm we use; by modifying the running time of the algorithm we can obtain any $6 < g \leq 10$.

Theorem 1 *With constant probability we will produce medians K such that $f^N(K) \leq O(1) f^N(K^*)$.*

Proof: The main idea of the proof is to show that each of the optimum medians has one of our medians nearby. This claim depends upon various probabilistic events, and requires the assumption that the optimum clusters are large.

Consider an optimal median set $K^* = \{K_1^*, \ldots, K_k^*\}$. For $1 \leq i \leq k$, let C_i^* be the cluster of points about K_i^*. Let $n_i = |C_i^*|$. Let n_i^S be the number of points in the ith optimal cluster that were also picked by the sample. For all $x \in N$, let $C^*(x)$ denote the closest median in K^* and $C(x)$ the closest median in K. Let $Q_i = \sum_{x \in C_i^*} d(x, C^*(x))$ and $R_i = \sum_{x \in C_i^*} d(x, C(x))$. That is, Q_i is total assignment cost under the optimal centers K^* for the points in the ith optimal cluster, and R_i is the total assignment cost of the same points to our centers K. We immediately observe that $\sum_{1 \leq i \leq k} Q_i = f^N(K^*)$ and that $\sum_{1 \leq i \leq k} R_i = f^N(K)$. Let Q_i^S and R_i^S represent the same summations restricted to cover only the points in the sample.

We observe that $\sum_{1 \leq i \leq k} Q_i^S = f^S(K^*)$ and that $\sum_{1 \leq i \leq k} R_i^S = f^S(K)$. The triangle inequality ensures a bound of $\min_{x \in C_i^* \cap S} (d(K_i^*, x) + d(x, C(x))) \leq \frac{1}{n_i^S}(Q_i^S + R_i^S)$ on the distance from optimum median K_i^* to the nearest K_j. We can then bound the cost of using our medians to cluster the entire data set N. The distance from a point $x \in N$ to the nearest median in K is

[1] We could reduce running time for a larger approximation ratio using the algorithm of Guha *et al.* [13] instead.

bounded by the sum of the distance to its optimum median and the distance from this optimum median to the nearest median in K. $f^N(K)$ is therefore bounded by

$$f^N(K) \leq \sum_{x \in N} d(x, C^*(x)) + \sum_{i=1}^{k} \frac{n_i}{n_i^S}(Q_i^S + R_i^S).$$

For each i, the expected value of $\frac{n_i}{n_i^S}$ is $\frac{n}{s}$, but the actual value may differ. Linearity of expectations does not apply, because the values of Q_i^S and R_i^S depend upon the choice of sample and are not independent of the value of n_i^S. We therefore apply Chernoff Bounds and make use of the assumption that the clusters are not large, to show that with constant probability all the $\frac{n_i}{n_i^S}$ are close to their expected values. By Chernoff Bounds, $\forall i \ \mathbf{Pr}[n_i^S < Bn_i(s/n)] < e^{\frac{-sn_i(1-B)^2}{2n}}$, where $0 < B < 1$ is a chosen parameter that trades the sample size against the approximation factor. We would like this probability to be smaller than $\frac{1}{k}$ so that there is a constant probability that *every* cluster is well represented in the sample. After some algebra, and making use of our assumption that $n_i \geq \frac{\epsilon n}{k}$, we can determine that $s = \frac{2}{(1-B)^2}\frac{k}{\epsilon}\log(32k)$ is sufficient to guarantee that $\mathbf{Pr}[\exists i \ n_i^s < Bn_i(s/n)] < \frac{1}{32}$. The important point is that, with constant probability, *all* of the optimum clusters will be well represented in the sample. Provided all optimum clusters are well represented, we can simplify the earlier expression given for the quality of clustering using the medians derived from the sample, obtaining

$$f^N(K) \leq f^N(K^*) + \frac{n}{sB}\sum_{i=1}^{k}(Q_i^s + R_i^s) = f^N(K^*) + \frac{n}{sB}(f^S(K) + f^S(K^*)).$$

Our medians K were chosen to minimize $f^S(K)$. Since the algorithm of Arya *et al.* provides a constant approximation, we conclude that $f^S(K^*)$ cannot be better by more than a constant factor of g. It follows that $f^N(K) \leq f^N(K^*) + \frac{n}{sB}(1+g)f^S(K^*)$. The final step is estimating the cost of the optimum medians on the sample. The expected value is $E[f^S(K^*)] = \frac{s}{n}f^N(K^*)$. Of course, there is some possibility that our sample has a much higher cost, but we can apply Markov's Inequality to bound the probability of this event. We observe that $\mathbf{Pr}[f^S(K^*) \leq \frac{16}{15}\frac{s}{n}f^N(K^*)] \geq \frac{1}{16}$. We conclude that $\mathbf{Pr}[f^N(K) \leq 1 + (1+g)\frac{16}{15B}f^N(K^*)] \geq 1 - \frac{1}{32} - \frac{15}{16} = \frac{1}{32}$. The required sample size was $s = \frac{2}{(1-B)^2}\frac{k}{\epsilon}\log(32k)$. If we consider $B = \frac{4}{5}$ we will obtain an approximation ratio of $1+(1+g)\frac{4}{3}$ with probability at least $\frac{1}{16}$, and a sample size of $50\frac{k}{\epsilon}\log(32k)$. By modifying the constants involved, we can make our approximation ratio arbitrarily close to $2+g$. ■

We have shown that a given candidate set, generated as described, has constant probability of being a good approximation to k–Median. One approach to clustering would be to change the constants so as to guarantee a high probability of success. We observe, however, that our proof depends upon the claim that $f^S(K^*) = \frac{s}{n}f^N(K^*)$. This claim holds in an expected sense but not with high probability. It might be the case that a small number of points contribute significantly to the cost of clustering; imagine a set of $\frac{n}{s}$ points which account for nearly all the cost of the optimum clusters on the set N. Our sample expects to see only one of these points, but there is reasonably good probability we will see two, thus doubling the value of $f^S(K^*)$ relative to $f^N(K^*)$. Guaranteeing a good result with high probability would therefore cost us at least a factor of two in the approximation ratio.

Instead, we will independently generate m candidate sets, where m is a constant. The probability that one of them produces a good approximation tends towards $1 - (\frac{31}{32})^m$. This process allows us to guarantee with high probability that we can generate a set of candidates such that *one of them* is a good approximation to k–Median. That is, if the candidate sets are $K_1, \ldots, K_m$, and K_{BEST} is the candidate set with the lowest f^N cost, then with probability at least $1 - (31/32)^m$, $f^N(K_{BEST}) \leq (1 + \frac{4}{3}(1+g))f^B(K^*)$. Next we give a procedure that will identify a good median set from $K_1, \ldots, K_m$.

5 Close Estimation of Candidate Quality

Suppose we are given m sets of candidate medians K_1 through K_m. We would like to determine which of these sets minimizes $f^N(K_i)$, in sublinear time. Clearly it is possible to measure $f^N(K_i)$ exactly for each i, by calculating, for each K_i, the assignment distances of *all* n points in N to the medians K_i. This process would take $\Omega(nkm)$ time, however, defeating the purpose of using a sampling-based algorithm. Instead, we will *estimate* each $f^N(K_i)$.

The estimation procedure will be as follows. l independent, uniform random estimation samples $S_1, \ldots, S_l$ will be drawn from N. The number l and the size of these samples will be discussed shortly. On each of these estimation samples S_j, for each K_i, $f^{S_j}(K_i)$ will be computed *exactly*. For each K_i, the estimated cost $f^{EST}(K_i)$ will be calculated as $\min_{1 \leq j \leq l} (\frac{n}{s} f^{S_j}(K_i))$. The candidate set with minimum f^{EST} will be returned as the final k–Median solution.

First we will show that K_{BEST}, the candidate set with lowest f^N cost, will with high probability have $f^{EST} \leq \alpha f^N(K_{BEST})$, for some small constant α. It will follow that with high probability there will be some K_i with a small $f^{EST}(K_i)$, and therefore that the median set returned by the algorithm will at least have a low *estimated* cost. Finally we will show the soundness of the estimation procedure: that for every candidate set K_i it will be unlikely that $f^{EST}(K_i)$ is much *lower* than $f^N(K_i)$. It will follow that the median set returned has not only a low *estimated* cost but also a low *actual* (f^N) cost.

We introduce two new parameters α and c. The parameter α trades off the number of samples against the accuracy of the estimation, while c trades off the number of samples against the probability of failure. We independently draw $\frac{c\alpha}{\alpha-1}$ samples S_j from N, each of size $s = \Theta(\frac{k}{\epsilon} \log \frac{k}{\epsilon})$. For each median set i, we will compute $f^{EST}(K_i) = \min_j \frac{n}{s} f^{S^j}(K_i)$. We will show that the medians that minimize f^{EST} produce a good approximation to the minimum f^N.

5.1 Low-Estimate Property of Candidate Sets

Lemma 1 *For any candidate set K, if we compute $f^{EST}(K)$ using $\frac{c\alpha}{\alpha-1}$ samples ($\alpha \geq 2$), then with probability at least $1 - e^{-c}$, $f^{EST}(K) \leq \alpha f^N(K)$.*

Proof: The proof follows directly from Markov's Inequality, and the fact that for any random sample S, $E[f^S(K)] = (s/n)f^N(K)$. ■

As $f^N(K_{BEST})$ is with high probability close to $f^N(K^*)$, with high probability $f^{EST}(K_{BEST})$ is close to $f^N(K^*)$. Therefore, with high probability some candidate set will have an f^{EST} close to $f^N(K^*)$.

5.2 Soundness of the Estimation Method

For each median set K we can consider the ordering of all the points $x \in N$ by their distance $C(x)$ to the nearest median[2] Further, imagine dividing all the points of N into $b = \frac{kD}{\epsilon}$ "bins" each containing $\frac{\epsilon n}{kD}$ points (see Figure 1). The first bin will have the "cheapest" points (i.e., those with the smallest assignment distances) and the last bin will have the most expensive points. We will denote by $\textsc{cost}(i)$ the cost of the ith bin, that is, the sum of assignment distances of points in this bin.

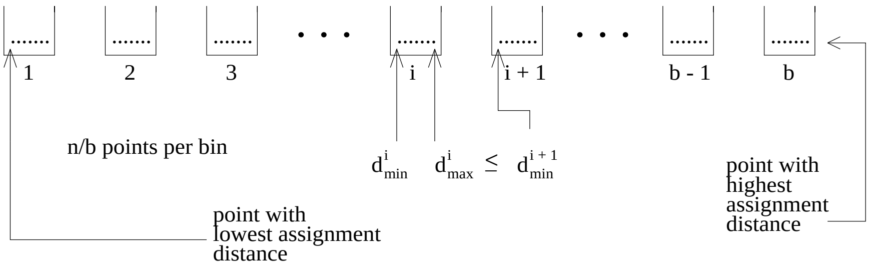
....
1
2
3
. . .
i
i + 1
b - 1
b
n/b points per bin
d^{i}_{min} d^{i}_{max} $\leq$ d^{i+1}_{min}
point with lowest assignment distance
point with highest assignment distance

Figure 1: Ordering of points by assignment distances under K.

By selecting our sample size to be sufficiently large, we will guarantee that with high probability each bin is "well-represented" in each sample for each median. We will then show how to bound the cost $f^N(K)$ in terms of $f^{EST}(K)$ under the assumption that all bins are well-represented.

We will set $s = \frac{4}{b(1-\beta^2)} \log\left(mb\frac{c\alpha}{\alpha-1}\right) = \Theta(\frac{k}{\epsilon}\log\frac{k}{\epsilon})$. Here β is a parameter trading off between the sample size and the accuracy of the estimation method, m is the number of samples, $b = \frac{kD}{\epsilon}$ is the number of bins, and α, c are the parameters defining the number of samples.

Lemma 2 *With high probability, for every triple of (bin, median, sample), we have at least $\frac{\beta s}{b}$ points from this bin in the sample.*

Proof: The expected number of points is $\frac{s}{b}$ since points from N are equally divided among the bins. Using Chernoff Bounds, the probability that a particular bin for a particular median has too few points in a particular sample is at most $e^{\frac{-s(1-\beta)^2}{2b}}$. Substituting the value of s yields a probability of at most $(mb\frac{c\alpha}{\alpha-1})^{-2}$. There are m median sets, b bins, and $\frac{c\alpha}{\alpha-1}$ samples, so the probability that there exists a bad triple remains very small (and can be reduced by increasing the constants). ■

Lemma 3 *With high probability, $f^{EST}(K) \geq \beta \sum_{i=1}^{b-1} \textsc{cost}(i)$.*

Proof: Let us denote by d_i^{max} the assignment distance of the last point, i.e., the point with largest assignment distance, in bin i. Let d_i^{min} be the assignment distance of the first point in bin i. Then $f^{EST}(K) \geq \frac{n}{s}\sum_{i=1}^{b}\frac{\beta s}{b}d_i^{min} \leq \frac{\beta n}{b}\sum_{i=1}^{b-1} d_i^{max} \leq \beta\sum_{i=1}^{b-1}\textsc{cost}(i)$. Here the first inequality holds with high probability because of lemma 2; the second inequality holds because of the manner in which points were assigned to bins (in order of increasing assignment costs). ■

[2] This ordering and division into bins are not computed by the algorithm, but are merely analytical tools in this proof.

Lemma 4 $\textsc{cost}(b) \le f(K^*) + \frac{1}{D-1}\sum_{i=1}^{b-1}\textsc{cost}(i)$.

Proof: We will show an upper bound on the cost of the most expensive bin, by showing a way of "moving" points in this bin to their centers[3]. We will call a point "bad" if it is in the most expensive bin, and we will call it "good" if it is not in any of the $D-1$ most expensive bins. Figure 2 shows the bins re-labelled as to whether they contain good, bad, or non-good points.

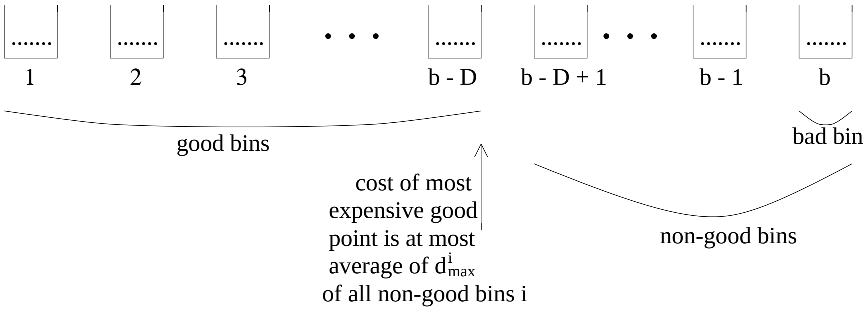
1
2
3
b - D
b - D + 1
b - 1
b
good bins
bad bin
cost of most expensive good point is at most average of d^i_{max} of all non-good bins i
non-good bins

Figure 2: Good, bad, and non-good bins.

Consider the optimal clusters $C_1^*, C_2^*, \ldots, C_k^*$ (those induced by the optimal medians K^*). Some of these contain bad points. We will imagine moving each bad point first to its optimal median and then to the location of a good point in the same optimal cluster, in such a way that each bad point has a unique new (good) location and so that the sum of new assignment distances of the (formerly) bad points is bounded. For each optimal cluster containing one or more bad points, we can find for each bad point a unique good point in the same cluster (because each optimal cluster contains at least $\frac{\epsilon n}{k}$ points, and there are at most $(D-1)\frac{\epsilon n}{kD}$ points in N which are not good). If we move each bad point to the location of a good point in the same optimal cluster, then we know that the sum of the movement distances is at most $f^N(K^*)$. This is because each bad point is moved at most $d_1 + d_2$ where d_1 is its own assignment distance under the optimal centers, and d_2 is the assignment distance of a different point in that optimal cluster; adding up a d_1 and a d_2 for each bad point, we get a sum of only some of the optimal-median assignment distances, which is then at most $f^N(K^*)$. Figure 3 illustrates this process.

Now that the bad points are all at good locations, we need to show that the remaining distance to their medians in K is not large. In the worst case, each good point has cost $\frac{1}{D-1}\sum_{i=b-D+1}^{b-1} d_i^{max}$. This is the average, over the $D-1$ most expensive bins, of the cost of the most expensive point in the bin, which is certainly an upper limit on the cost of any good point. We pay at most $\frac{\epsilon n}{kD}\frac{1}{D-1}\sum_{i=b-D+1}^{b-1} d_i^{max} \le \frac{n}{b}\frac{1}{D-1}\sum_{i=1}^{b-1} d_i^{max} = \frac{1}{D-1}\sum_{i=1}^{b-1}\textsc{cost}(i)$ for the remaining distance. ■

Theorem 2 *With high probability,* $f^N(K) \le f^N(K^*) + \frac{1}{\beta}\frac{D}{D-1}f^{EST}(K)$.

Proof: This follows directly from the two lemmas. ■

By adjusting the constants, we can cause the above expression to converge so that $f^N(K)$ is bounded by an expression arbitrarily close to $f^N(K^*) + f^{EST}(K)$. Combining this result with lemma 1, we have $f^N(K) - f^N(K^*) \le f^{EST}(K) \le \alpha f^N(K)$.

[3] The idea of "moving" points is only a tool for analyzing the assignment distances: if there is a path over which a point x can travel to its center $C(x)$, then the length of this path is certainly an upper bound on the assignment distance of x.

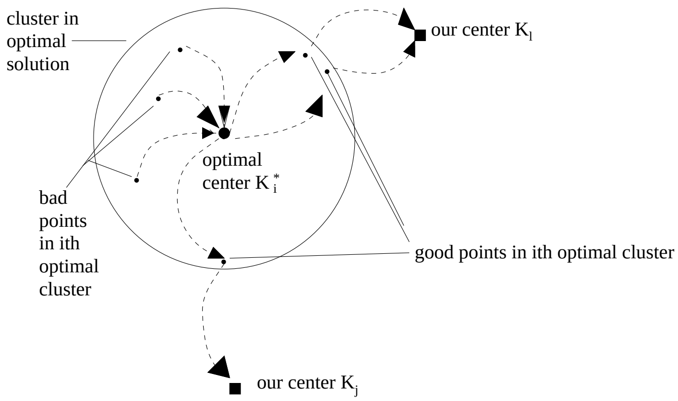
cluster in optimal solution
our center K_l
optimal center K_i^*
bad points in ith optimal cluster
good points in ith optimal cluster
our center K_j

Figure 3: Assignment distances of bad points.

6 Combining Generation and Estimation

We can now prove the following theorem.

Theorem 3 *Given an instance of k–Median, a data set N of n points with metric dist, and an integer k, with optimal solution K^* having all clusters of size at least $\frac{\epsilon n}{k}$, we can with high probability produce a $(9+O(\frac{1}{l}))$-approximation (where $l \in \{1,\ldots,k\}$), in running time $O((\frac{k^2}{\epsilon}\log k)^l)$.*

Proof: We first generate a large number of candidate median sets. By Theorem 1 the approximation ratio of the best candidate set is with high probability bounded above by a quantity arbitrarily close to $2+g$. We then perform the testing step, selecting a candidate K whose estimated cost $f^{EST}(K)$ is (by Theorem 2) with high probability between $f^N(K) - f^N(K^*)$ and $\alpha f^N(K)$. Therefore, the true approximation factor of the final medians is 3+g, with high probability. g is the approximation ratio of the continuous k–Median algorithm used to find the candidates; the algorithm of Arya *et al.* yields $g = 6 + 4/l$ and running time $O((\frac{k^2}{\epsilon}\log\frac{k}{\epsilon})^l)$ on a sample of size $\Theta(\frac{k}{\epsilon}\log\frac{k}{\epsilon})$. The estimation takes time $O(\frac{k^2}{\epsilon}\log\frac{k}{\epsilon})\frac{cm\alpha}{\alpha-1}$, where m is the $(O(1))$ number of candidate sets and $\frac{c\alpha}{\alpha-1}$ is the $(O(1))$ number of estimation samples. The theorem follows. ■

7 Clustering with Outliers

Instead of assuming all the optimum clusters are large, we can instead allow ourselves to discard some fraction δ of the demand points. These points are considered outliers (or noise) and their distances are not included in the cost of the solution. We will call the cost of a solution, discarding the most expensive δn points, the $(1-\delta)$–cost of the solution.

Claim 4.1 *There is a sampling-based algorithm which, for constant α, takes a point set N and finds k medians whose $(1-\alpha\delta)$–cost is (with high probability) within a constant factor of the optimum $(1-\delta)$–cost.*

Proof Sketch: Let $|N| = n$, and assume that some optimal $(1-\delta)$ solution for N has medians $K^* = \{K_1^*, \ldots, K_k\}$ and $(1-\delta)$–cost C.

We draw a sample S of s points; for large enough s, S should contain $\Theta(\frac{s}{n})|C_i^*|$ points from each optimum cluster C_i^* of at least $\frac{\delta n}{k}$ points. With high probability, fewer than $2\delta s$ of the δn points excluded by the K^* solution should land in S. Therefore, there should be a $(1-2\delta)$–solution for S, with cost $O(1)$ times the scaled optimum; the algorithm of Charikar *et al.* [8] will find such a solution while excluding $O(\delta)$ of the points in S.

For each i, let C_i^* denote the set of points assigned to K_i^* in the $(1-\delta)$ solution for N. For each i, there are three cases:

1. Fewer than half the points in $C_i^* \cap S$ were excluded in our solution; therefore, we have a center "close" to the corresponding K_i^*, and it follows from the earlier theorems that the assignment distances for these points should be low.
2. More than half the points in $C_i^* \cap S$ were excluded in our solution. Then $C_i^* \cap S$ is at most twice as large as the excluded set. As s was chosen to ensure S would (with high probability) have a good representation of all clusters larger than $\frac{\delta n}{k}$, we can exclude C_i from our $(1-\alpha\delta)--cost$.
3. C_i is underrepresented in our sample (has fewer than, say, $|C_i|s/2$ points in S). Then with high probability $|C_i| < \frac{\delta n}{k}$, and we can exclude C_i in finding our $(1-\alpha\delta)$–cost on N.

Therefore, the medians found on S should have a good $(1-\alpha\delta)$–cost on N. ■

8 Lower Bounds

The following example illustrates the trade-off between sample size and the number of points that must be excluded from the clustering (or the minimum cluster size). Assume we are trying to cluster using a sample of size s, and that we want a $(1-\delta)$–cost competitive with the best $(1-\delta')$–cost. (We will see how δ is a function of s, but it certainly must be at least δ'.)

Consider the class of data sets consisting of:

- $k-1$ "small" clusters, each consisting of n/s co-located points.
- $\delta' n$ singleton points.
- One "big" cluster of $n - \frac{(k-1)n}{s} - \delta' n$ points that are all co-located.

Assume all inter-point distances are infinite (or very large), except distances between pairs of co-located points, which are zero.

Then the optimal $(1-\delta')$–cost is 0. We will say our sample "misses" a cluster if the cluster has no members in the sample. For each small cluster that is missed, our solution will have to exclude an additional n/s points in order to have cost competitive with the best $(1-\delta')$–cost.

We expect each small cluster to contribute one point to the sample. Therefore, each cluster has constant probability of being missed, and so we expect the sample to miss $\Omega(k)$ small clusters. Even is we repeat our sampling a constant number of times, it is still likely that every sample will miss $\Omega(k)$ small clusters.

We conclude that we must either have $\delta \geq \delta' + \Omega(\frac{k}{s})$ or our optimum solution must disallow clusters of size $O(\frac{n}{s})$ or smaller.

Our upper bounds match these lower bounds to within a factor of $\Theta(\log \frac{k}{\epsilon})$. This factor ensures that clusters are represented in (roughly) equal proportion; this condition is required for our analysis and is naturally somewhat stronger than the lower bound condition that each cluster have at least one representative in the sample. It is straightforward to extend our results to consider samples whose size depends on n; for example if we use samples of size $\Theta(\sqrt{n}\log n)$, we can cluster a data set in which the optimum solution has no cluster of size smaller than $\sqrt{n}$.

References

[1] N. Alon, S. Dar, M. Parnas, and D. Ron. ”Testing of clustering.” In *Proc. FOCS*, 2000.

[2] S. Arora, P. Raghavan, and S. Rao. ”Approximation schemes for euclidean k–medians and related problems.” In *Proc. STOC*, 1998.

[3] V. Arya, N. Garg, R. Khandekar, and V. Pandit. ”Local search heuristics for k–median and facility location problems.” In *Proc. STOC*, 2001.

[4] P. S. Bradley, U. M. Fayyad, and C. Reina. ”Scaling clustering algorithms to large databases.” In *Proc. KDD*, 1998.

[5] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. “Incremental clustering and dynamic information retrieval.” In *Proc. STOC*, 1997.

[6] M. Charikar and S. Guha. ”Improved combinatorial algorithms for the facility location and k–median problems.” In *Proc. FOCS*, 1999.

[7] M. Charikar, S. Guha, E. Tardos, and D. Shmoys. ”A constant-factor approximation algorithm for the k–median problem (extended abstract).” In *Proc. STOC*, 1999.

[8] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. ”Algorithms for facility location problems with outliers.” In *Proc. SODA*, 2001.

[9] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1972.

[10] T. Feder and D. Greene. “Optimal algorithms for approximate clustering.” In *Proc. STOC*, 1988.

[11] T. F. Gonzalez. “Clustering to minimize the maximum intercluster distance.” *Theoretical Computer Science* 38(2-3):1985.

[12] S. Guha. “Approximation algorithms for facility location problems.” Stanford University Ph.D. Thesis, 2000.

[13] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan. “Clustering Data Streams.” In *Proc. FOCS*, 2000.

[14] S. Guha, R. Rastogi, and K. Shim. “CURE: An efficient clustering algorithm for large databases.” In *Proc. SIGMOD*, 1998.

[15] J. A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, 1975.

[16] P. Indyk. “Sublinear time algorithms for metric space problems.” In *Proc. STOC*, 1999.

[17] N. Jain and V. V. Vazirani. “Primal-dual approximation algorithms for metric facility location and k-median problems.” In *Proc. FOCS*, 1999.

[18] N. Joze-Hkajavi and K. Salem. “Two-phase clustering of large datasets.” 1998.

[19] J. Marroquin and F. Girosi. “Some extensions of the k–Means algorithm for image segmentation and pattern recognition.” In *AI Memo 1390, MIT AI Lab.*, 1993.

[20] N. Mishra, D. Oblinger, and L. Pitt. “Sublinear Time Approximate Clustering.” In *Proc. SODA*, 2001, pages 439–447.

[21] C. R. Palmer and C. Faloutsos. “Density biased sampling: an improved method for data mining and clustering.” In *Proc. SIGMOD*, 2000, pages 82–92.

[22] D. Pelleg and A. W. Moore. “Accelerating exact k–means algorithms with geometric reasoning.” In *Proc. KDD*, 1999.

[23] T. Zhang, R. Ramakrishnan, and M. Livny. “BIRCH: an efficient data clustering method for very large databases.” In *Proc. SIGMOD*, 1996, pages 103–114.