Performance guarantees for hierarchical clustering

Sanjoy Dasgupta

University of California, Berkeley
dasgupta@cs.berkeley.edu

Abstract. We show that for any data set in any metric space, it is possible to construct a hierarchical clustering with the guarantee that for *every* k, the induced k-clustering has cost at most eight times that of the optimal k-clustering. Here the cost of a clustering is taken to be the maximum radius of its clusters. Our algorithm is similar in simplicity and efficiency to common heuristics for hierarchical clustering, and we show that these heuristics have poorer approximation factors.

1 Introduction

A *hierarchical clustering* of n data points is a recursive partitioning of the data into $2, 3, 4, \ldots$ and finally n, clusters. Each intermediate clustering is made more fine-grained by dividing one of its clusters. Figure 1 shows one possible hierarchical clustering of a five-point data set.

Such hierarchical representations of data have long been a staple of biologists and social scientists, and since the sixties or seventies they have been a standard part of the statistician's toolbox. They require no prior specification of the number of clusters, they permit the data to be understood simultaneously at many levels of granularity, and there are some very simple greedy heuristics which can be used to construct them.

The wide popularity of this kind of data analysis has fueled, and been reinforced by, a large body of theoretical work (e.g., Hartigan, 1985, and the references therein). However, there is still a fundamental and nontrivial existence question which needs to be addressed: must there always exist a hierarchical

Figure 1: A hierarchical clustering of five points.

clustering in which, for *every* k, the induced k-clustering (grouping into k clusters) is close to the optimal k-clustering under some reasonable cost function? After all, it could be that the optimal cost-based k-clustering cannot be obtained by merging clusters of the optimal $(k+1)$-clustering, and that in fact the two are so far removed that they cannot be reconciled even approximately into a hierarchical structure. The conventional wisdom is that a hierarchical clustering cannot be optimal at *every* level, and this is true; however, we show that the story is quite different if approximate optimality is sufficient.

Theorem 1. *Take the cost of a clustering to be the largest radius of its clusters. Then, any data set in any metric space has a hierarchical clustering in which, for each k, the induced k-clustering has cost at most eight times that of the optimal k-clustering.*

We present an algorithm for constructing such a hierarchy which is similar in simplicity and efficiency to standard heuristics for hierarchical clustering. It is based upon the *farthest-first traversal* of a set of points, introduced by Hochbaum and Shmoys (1985) as an approximation algorithm for the closely-related *k-center* problem. Their use of this traversal for clustering is ingenious, and in fact just a cursory examination of its properties is necessary for their result. For hierarchical clustering, we examine it in far greater detail and need to built upon it. Specifically, the farthest-first traversal of n data points yields a sequence of "centers" $\mu_1, \ldots, \mu_n$ such that for any k, the first k of these centers define a k-clustering which is within a factor two of optimal. However, the n clusterings created in this way are not hierarchical. Our main contribution is to demonstrate a simple and elegant way of using the information found by the traversal to create a hierarchical clustering. The approximation factor we achieve might not be optimal; however, it is possible to construct cases showing that a factor of at least two is inevitable.

Unlike our algorithm, common heuristics for hierarchical clustering work *bottom-up*, starting with a separate cluster for each point, and then progressively merging the two "closest" clusters until only a single cluster remains. The different schemes are distinguished by their notion of closeness. In *single-linkage* clustering, the distance between two clusters is the distance between their closest pair of points. In *complete-linkage* clustering, it is the distance between their farthest pair of points (and thus complete-linkage is explicitly trying to minimize our cost function). *Average-linkage* has many variants; in the one we consider, the distance between clusters is the distance between their means (Hartigan, 1985; Eisen *et al.*, 1998).

We analyze the worst-case behavior of these three heuristics, and find that their approximation ratios are unbounded.

Theorem 2. *For any k, single-linkage can produce induced k-clusterings which are a multiplicative factor k from optimal, while average- and complete-linkage can be off by a multiplicative factor of* $\log_2 k$.

The problems of single-linkage clustering are already well understood by statisticians; our lower bound on its performance can be seen as a convenient

quantification of this common knowledge. On the other hand, our bad cases for the other two heuristics yield new insights into their behavior.

For an illustrative application of hierarchical clustering, the reader is directed to recent analyses of gene expression data (for instance, Alizadeh *et al.*, 2000, and Eisen *et al.*, 1998). These high-dimensional data sets contain well-separated groups at various levels of detail, and the use of hierarchical clustering allows the data, and the cluster information, to be displayed with stunning clarity.

2 Some background on clustering

2.1 Approximation algorithms for clustering

The most widely-used clustering algorithms – k-means, EM, and (hierarchical) agglomerative clustering – have received almost no attention from theoretical computer scientists. Some exceptions include work by Kearns, Mansour, and Ng (1997), and by Dasgupta and Schulman (2000). On the other hand, there has been a lot of theoretical work on the *k-center* and *k-medians* problems. In either case, the input consists of points in Euclidean space (or more generally, in a metric space) as well as a preordained number of clusters k, and the goal is to find a partition of the points into clusters $C_1, \ldots, C_k$, and also cluster centers $\mu_1, \ldots, \mu_k$, so as to minimize some cost function which is related to the radius of the clusters.

(a) *(geometric) k-center*: Maximum radius of clusters

$$\max_j \max_{x \in C_j} \|x - \mu_j\|$$

(b) *(geometric) k-medians*: Average radius of clusters

$$\sum_j \sum_{x \in C_j} \|x - \mu_j\|$$

Both problems are NP-hard but have simple constant-factor approximation algorithms. For k-center, a 2-approximation was found by González (1985) and by Hochbaum and Shmoys (1985), and this is the best approximation factor possible (Feder and Green, 1985). For k-medians there has been a series of results, of which the most recent (Charikar and Guha, 1999) achieves an approximation ratio of eight on this geometric version of the problem.

What does a constant-factor approximation mean for a clustering problem? Consider the scenario of Figure 2. The solid lines show the real clusters, and the three dots represent the centers of a bad 3-clustering whose cost (in either measure) exceeds that of the true solution by a factor of at least ten. This clustering would therefore not be returned by the approximation algorithms we mentioned. However, EM and k-means regularly fall into local optima of this kind, and practitioners have to take great pains to try to avoid them. In this sense, constant-factor approximations avoid the worst, and might provide good starting points for local search procedures (like EM) which can then fine-tune the solution.

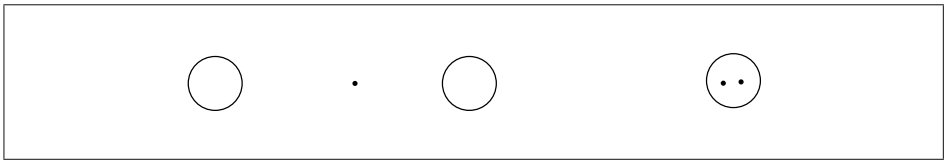

Figure 2: The circles represent an optimal 3-clustering; all the data points lie within them. The dots are centers of a really bad clustering.

Input: n data points with a distance *metric* $d(\cdot, \cdot)$.

Pick a point and label it 1.
For $i = 2, 3, \ldots, n$
Find the unlabeled point furthest from $\{1, 2, \ldots, i-1\}$, and label it i. Use the standard notion of distance from a point to a set: $d(x, S) = \min_{y \in S} d(x, y)$.
Let $\pi(i) = \arg\min_{j<i} d(i, j)$.
Let $R_i = d(i, \pi(i))$.

Figure 3: Farthest-first traversal of a data set.

2.2 Previous work on hierarchical clustering

There has been a lot of work on hierarchical clustering in the theoretical statistics community, and we will describe just one typical kind of analysis (Hartigan, 1975). Fix some underlying distribution P, and from it define some "natural" hierarchical clustering T (based, say, on level sets of P). Also fix an algorithm, and let the random variable T_n denote the hierarchical clustering returned by this algorithm when it is given as input n i.i.d. points from P. Is it true that as n approaches infinity, T_n approaches T almost surely? And if so, what is the variance of this estimator, asymptotically?

3 An approximation algorithm for hierarchical clustering

3.1 Farthest-first traversals

Hochbaum and Shmoys (1985) introduced the *farthest-first traversal* of a data set as an approximation algorithm for what is sometimes called the *k-center* problem, that of finding an optimal k-clustering under our cost function, maximum cluster radius. The idea is to pick any data point to start with, then choose the point furthest from it, then the point furthest from the first two (the distance of a point x from a set S is the usual $\min\{d(x, y) : y \in S\}$), and so on until k points are obtained. These points are taken as cluster centers and each remaining point is assigned to the closest center. If the distance function is a metric, the resulting clustering is within a factor two of optimal. For hierarchical clustering, we will study the farthest-first traversal in detail, and will need to build upon it.

Starting with n points in a metric space, number the points in it using a farthest-first traversal (Figure 3). For any point i, describe its closest neighbor

among $1, 2, \ldots, i-1$ as its *parent*, $\pi(i)$. Let R_i be its distance to this parent,

$$R_i \;=\; d(i, \pi(i)) \;=\; d(i, \{1, 2, \ldots, i-1\}).$$

The Hochbaum-Shmoys algorithm uses points $1, 2, \ldots, k$ as centers for a k-clustering. Let $\mathbb{C}_k$ be this clustering.

We begin by showing that the distances R_i are decreasing, and that the cost of each k-clustering $\mathbb{C}_k$ is exactly R_{k+1}.

Lemma 1. *Adopt the convention that $R_1 = \infty$ and $R_{n+1} = 0$.*
(1) For all i, $\pi(i) < i$.
(2) $R_1 \geq R_2 \geq R_3 \geq \cdots \geq R_n$.
(3) For all k, cost$(\mathbb{C}_k) = R_{k+1}$.

Proof. By the manner in which any point i is chosen, we know that for all $j > i$,

$$\begin{aligned} R_j &= d(j, \pi(j)) \\ &= d(j, \{1, 2, \ldots, j-1\}) \\ &\leq d(j, \{1, 2, \ldots, i-1\}) \\ &\leq d(i, \{1, 2, \ldots, i-1\}) \\ &= R_i. \end{aligned}$$

This immediately gives us (2). To see (3), notice that for all $i > k$,

$$d(i, \{1, 2, \ldots, k\}) \;\leq\; d(k+1, \{1, 2, \ldots, k\}) \;=\; R_{k+1}.$$

∎

For easy reference, and to illustrate our notation, we repeat here the result of Hochbaum and Shmoys.

Lemma 2 (Hochbaum-Shmoys). *For any k,*

$$\textit{cost}(\mathbb{C}_k) \leq 2 \cdot \textit{cost(optimal } k\textit{-clustering).}$$

Proof. Let $S_1, \ldots, S_k$ denote the clusters of the optimal k-clustering. Suppose one of these clusters contains two or more of the points $\{1, 2, \ldots, k\}$. These points are at distance at least R_k from each other. Therefore the clusters must have radius at least $\frac{1}{2}R_k \geq \frac{1}{2}R_{k+1} = \frac{1}{2}\text{cost}(\mathbb{C}_k)$.

On the other hand, if each optimal cluster S_j contains exactly one of the points $\{1, 2, \ldots, k\}$, then (for any j) every point in S_j is within $2 \cdot \text{radius}(S_j)$ of $\{1, 2, \ldots, k\}$. Therefore

$$\text{cost}(\mathbb{C}_k) \;=\; R_{k+1} \;=\; d(k+1, \{1, 2, \ldots, k\}) \;\leq\; 2 \cdot \max_j \text{radius}(S_j),$$

and again we're done. ∎

3.2 Constructing a hierarchical clustering

The Hochbaum-Shmoys result gives us an ordering of the points such that for any k, the first k points constitute the centers of a near-optimal k-clustering $\mathbb{C}_k$. Unfortunately, the n clusterings defined in this manner are not hierarchical.

We will stick with the same choice of centers, but we will no longer be able to assign each remaining point to the closest center. As a first try at a hierarchical clustering,

- construct a tree T^π on nodes $\{1, 2, \ldots, n\}$, rooted at 1, and with an edge between each point i and its parent $\pi(i)$ (see Figure 4 for an example); and
- for any k, let $\mathbb{C}_k^\pi$ be the clustering with centers $1, 2, \ldots, k$, in which each remaining point is associated with the center which is its closest ancestor in T^π.

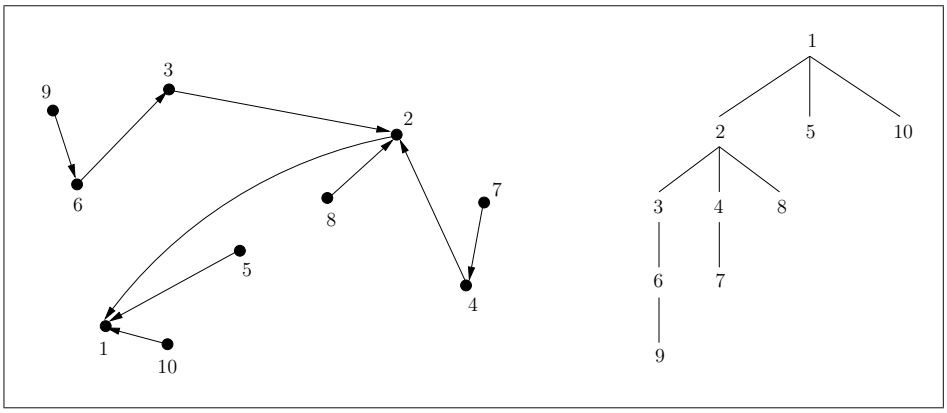

Figure 4: A first try at a hierarchical clustering of ten points in $\mathbf{R}^2$, under the Euclidean metric. The graph on the left has been numbered by a farthest-first traversal, and has an edge from each point i to its parent $\pi(i)$. The tree on the right is T^π. The numbering and the tree are completely determined by the choice of point number one (and by the method of breaking any ties that arise).

These clusterings $\{\mathbb{C}_1^\pi, \mathbb{C}_2^\pi, \ldots, \mathbb{C}_n^\pi\}$ *are* hierarchical; the $(k+1)$-clustering $\mathbb{C}_{k+1}^\pi$ is obtained from the k-clustering $\mathbb{C}_k^\pi$ by splitting the cluster centered at point $\pi(k+1)$. However, if the tree T^π is very deep (taking distances into account), then the cost of an induced k-clustering could be a lot larger than $\text{cost}(\mathbb{C}_k)$. We therefore need to shrink the tree somehow. We will do so by adjusting the parent function $\pi(\cdot)$ to get a $\pi'(\cdot)$ for which $\text{cost}(\mathbb{C}_k^{\pi'}) \leq 4\text{cost}(\mathbb{C}_k)$.

The parent function $\pi(\cdot)$ has two salient properties:

- $\pi(i) < i$; and
- $d(i, \pi(i)) = d(i, \{1, 2, \ldots, i-1\})$.

Input: n data points with a distance metric $d(\cdot, \cdot)$.

By a farthest-first traversal, number the points and obtain parent function $\pi(\cdot)$.
For each $i = 2, 3, \ldots, n$
$\pi'(i) = \min\{j < i : \ d(i, j) \leq 2 \cdot d(i, \pi(i))\}$.
Return the hierarchical clustering corresponding to tree $T^{\pi'}$.

Figure 5: A hierarchical clustering procedure.

We will now consider parent functions π' in which the second property is relaxed somewhat. We will require only that

$$d(i, \pi'(i)) \ \leq \ 2 \cdot d(i, \{1, 2, \ldots, i-1\})$$

and this will enable us to shrink the tree. The resulting hierarchical clustering algorithm is shown in Figure 5, and its effect on our earlier example is shown in Figure 6.

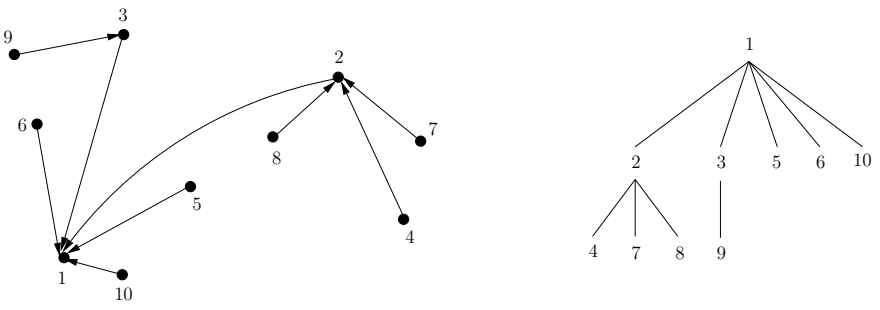

Figure 6: A continuation of the example of Figure 4. On the left is the new parent function π'; on the right, the new tree $T^{\pi'}$, which defines the final hierarchical clustering.

3.3 A performance guarantee

We now show a strong guarantee for the hierarchical clustering induced by π'. To this end, fix any k, and define the sequence $k_0, k_1, k_2, \ldots, k_J$ by

- $k_0 = k + 1$
- if $R_{k_j} > 0$ then $k_{j+1} = \min\{i : R_i < \frac{1}{2} R_{k_j}\}$ (note $k_{j+1} > k_j$);
  otherwise end the sequence, that is, $J = j$.

For the last element k_J we know $R_{k_J} = 0$; recall our convention that $R_{n+1} = 0$. The sequence definition ensures that R_{k_j} decreases geometrically (or faster).

Lemma 3. *For the sequence k_j defined above,*

(1) If $k_j \le i < k_{j+1}$, then

$$R_{k_j} \ge R_i \ge \tfrac{1}{2} R_{k_j} > R_{k_{j+1}}.$$

(2) $R_{k_j} < R_{k_0} \cdot \frac{1}{2^j}$.

We will next see that for any point i in the interval $[k_j, k_{j+1})$, its new parent $\pi'(i)$ must lie in $[1, k_j)$. Therefore, tree $T^{\pi'}$ has height at most J.

Lemma 4. *For any i in the range $k_j \le i < k_{j+1}$, we have*

(1) $\pi'(i) < k_j$, and

(2) $d(i, \pi'(i)) \le 2R_{k_j}$.

Proof. Let's start with (1). Pick such an i; then

(i) In the farthest-first traversal, since $i \ge k_j$, we know

$$d(i, \{1, 2, \ldots, k_j - 1\}) \;\le\; d(k_j, \{1, 2, \ldots, k_j - 1\}) \;=\; R_{k_j}.$$

(ii) As observed in Lemma 3, $R_{k_j} \le 2R_i = 2d(i, \pi(i))$.

Combining these two gives

$$d(i, \{1, 2, \ldots, k_j - 1\}) \le 2d(i, \pi(i)),$$

whereupon $\pi'(i) < k_j$.

Now for (2): $d(i, \pi'(i)) \le 2d(i, \pi(i)) = 2R_i \le 2R_{k_j}$. ∎

Lemma 5. *If $d(\cdot, \cdot)$ is a metric, then the cost of the k-clustering induced by π' is at most four times that of $\mathbb{C}_k$,*

$$cost(\mathbb{C}_k^{\pi'}) \;\le\; 4cost(\mathbb{C}_k).$$

Proof. Pick any point x_0, and say it gets associated with center $i \le k$ in clustering $\mathbb{C}_k^{\pi'}$. That is, of the points $\{1, 2, \ldots, k\}$, i is the closest ancestor of x_0 in $T^{\pi'}$. Consider the path in this tree which ascends from x_0 to i:

$$x_0 \;—\; x_1 \;—\; \cdots \;—\; x_{l-1} \;—\; x_l = i,$$

where each $x_j = \pi'(x_{j-1})$, and $x_0 > x_1 > \cdots > x_{l-1} > k$. By the previous lemma, these points $x_0, \ldots, x_{l-1}$ touch each interval $[k_j, k_{j+1})$ at most once, and for any x in such an interval, $d(x, \pi'(x)) \le 2R_{k_j}$. Therefore, since $d(\cdot, \cdot)$ obeys the triangle inequality,

$$\begin{aligned} d(x_0, i) &\le d(x_0, x_1) + d(x_1, x_2) + \cdots + d(x_{l-1}, x_l) \\ &\le \sum_{j=0}^{\infty} 2R_{k_j} \\ &\le \sum_{j=0}^{\infty} 2R_{k_0} \cdot \frac{1}{2^j} \\ &= 4R_{k_0} \quad = \quad 4R_{k+1}. \end{aligned}$$

where the third step uses Lemma 3. We finish up the proof with $R_{k+1} = \text{cost}(\mathbb{C}_k)$ (Lemma 1). ∎

This holds for all k; with Lemma 2 we can then clinch Theorem 1.

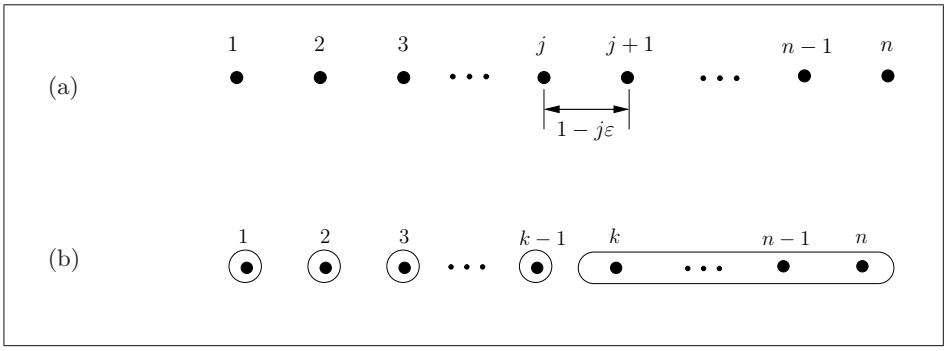

Figure 7: (a) Data points on a line. (b) The k-clustering induced by a single-linkage heuristic.

4 Worst-case performance of standard agglomerative clustering heuristics

4.1 Single linkage

Single-linkage clustering has a performance ratio of at least k for intermediate k-clusterings, even in the simple case when the data points lie on a line. This particular kind of bad behavior is known to statisticians as "chaining" (Hartigan, 1985).

Consider the set of points shown in Figure 7(a). Let the distance between any two adjacent points j and $j+1$ be $1-j\varepsilon$, for some tiny $\varepsilon > 0$. Then the intermediate k-clustering found by single linkage is as shown in the bottom half of the figure. It consists of one large cluster containing $n-k+1$ points, and $k-1$ singleton clusters. The diameter of the big cluster can be made arbitrarily close to $n-k$ by setting ε small enough. On the other hand, the optimal k-clustering has clusters of diameter at most $\lceil \frac{n}{k} \rceil - 1$, for vanishingly small ε. Therefore the approximation ratio is at least k.

4.2 Average linkage

The average-linkage heuristic can create intermediate k-clusterings of cost $\log k$ times optimal. Fix any k which is a power of two. Our bad example in this case involves points in $(\log k)$-dimensional space under an L_1 metric. Again we will make use of a tiny constant $\varepsilon > 0$.

- For $j = 1, 2, \ldots, \log k$, define $B_j = (1-\varepsilon j)\cdot\{-1, +1\} = \{-(1-\varepsilon j), +(1-\varepsilon j)\}$.
- Let $S = B_1 \times B_2 \times \cdots \times B_{\log k}$ be the set of vertices of a $(\log k)$-dimensional cube whose side lengths are just slightly less than two.
- Arbitrarily label the points of S as $x_1, \ldots, x_k$. Now define points $S' = \{x'_1, \ldots, x'_k\}$ as follows. For $j = 1, 2, \ldots, k$,
  - if x_j has a positive first coordinate, then let x'_j be identical to x_j, but with first coordinate $+3 + j \cdot 2\varepsilon \log k$;

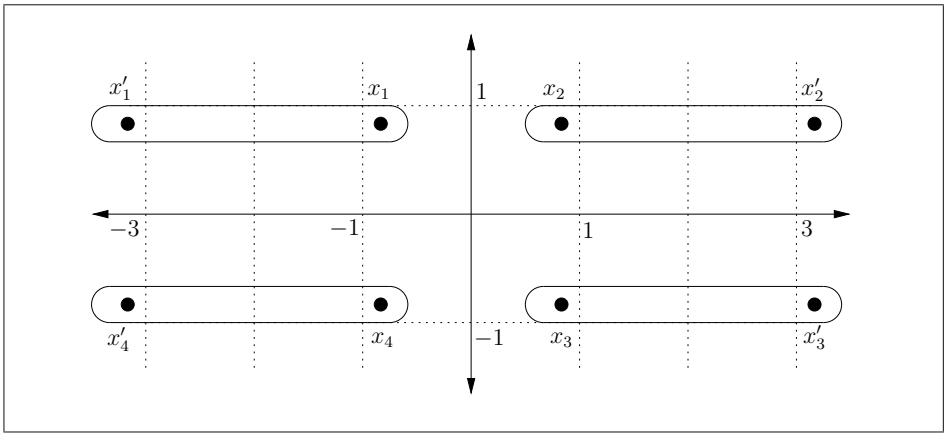

Figure 8: A two-dimensional bad case for average-linkage clustering. The optimal 4-clustering is shown, and has clusters of diameter ≈ 2. Average-linkage will put the central four points together; these have diameter ≈ 4. By increasing the dimension to $\log k$, we find that average-linkage can return k-clusterings whose component clusters have diameter $\log k$ times optimal.

- if x_j has a negative first coordinate, then let x'_j be identical to x_j, but with first coordinate $-3 - j \cdot 2\varepsilon \log k$.

– The data set so far consists of $S \cup S'$, a total of $2k$ points. Duplicate points as necessary to get the count up to n. Figure 8 shows this data set for $k = 4$.

Lemma 6. *For the data set just defined, under the L_1 metric,*

(1) The distance between any two distinct points of S' is at least two.

(2) The distance from any point in S' to $[-1, +1]^{\log k}$ is more than two.

(3) Any two points in S which disagree on the j^{th} coordinate have distance at least $2(1 - \varepsilon j)$ between them.

Proof. (1) Pick distinct $x'_a, x'_b \in S'$. If x_a, x_b disagree on the first coordinate then x'_a, x'_b differ by at least six on the first coordinate. If x_a, x_b differ on the j^{th} coordinate, then x'_a, x'_b differ by at least $2\varepsilon \log k$ on the first coordinate, and by $2(1 - \varepsilon j)$ on the j^{th} coordinate, giving a total of at least two.
(2) This can be seen by considering the first coordinate alone. ∎

The closest pairs of points (once duplicates get merged) are therefore those pairs in S which disagree only on the last coordinate. The distance between such pairs is $2(1 - \varepsilon \log k)$. They get merged, and in this way S is reduced to just $k/2$ clusters, with means $B_1 \times B_2 \times \cdots \times B_{\log k - 1} \times \{0\}$. Continuing in this way, eventually all the points in S get merged into one cluster centered at the origin, while the points of S' remain untouched. During this phase, clusters getting merged always have means which are within distance two of each other, and which lie in $[-1, +1]^{\log k}$.

The k-clustering therefore includes a large cluster containing all of S. The diameter of the cluster is at least the diameter of S, namely $2\log k - \varepsilon \log k - \varepsilon \log^2 k$.

There is a better k-clustering: $\{x_1, x_1'\}, \{x_2, x_2'\}, \ldots, \{x_k, x_k'\}$. These clusters have diameter at most $2 + 2\varepsilon k \log k + \varepsilon \log k$. Letting ε go to zero, we get an approximation ratio of $\log k$.

4.3 Complete linkage

Our negative result for complete linkage requires an unusual (non-L_p) metric. The data lie in $\mathbf{R} \times \mathbf{R}$, and the distance between two points (x, y) and (x', y') is defined as

$$d((x, y), (x', y')) \;=\; \mathbf{1}(x \neq x') + \log_2(1 + |y - y'|).$$

It can quickly be confirmed that this is a metric.

Assume k is a power of two for convenience. The data set consists of k clusters $S_1, S_2, \ldots, S_k$, each with k points (points can be duplicated to get the total up to n). Within each cluster, all points have the same y-coordinate, and have x-coordinates (say) $\{1, 2, \ldots, k\}$ (it doesn't matter what they are, as long as they are the same across clusters). Therefore the clusters all have diameter one. The y-spacing between clusters is shown in Figure 9 for the case $k = 4$. The y-distance between S_j and S_{j+1} is $1 - \varepsilon(\log_2 k - q)$, where 2^q is the largest power of two dividing j.

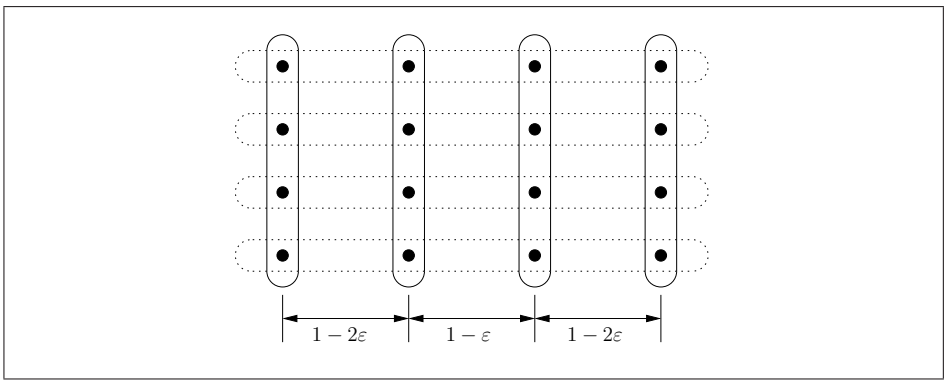

Figure 9: A bad case for complete-linkage clustering. The norm here is unusual; see the definition. The optimal 4-clustering is shown in bold and the 4-clustering found by complete-linkage is delineated by dots.

This example is set up so that the k-clustering found by complete linkage will have clusters which (each) touch every S_j, and which therefore have diameter $\log_2 k$, as ε goes to zero.

5 Practical issues and open questions

5.1 Small values of k

It is often sufficient to guarantee good k-clusterings just for small values of k, say in the hundreds or so. Therefore it would be quite heartening if it turned out that our $\Omega(\log k)$ lower bound on the approximation ratio of average- and complete-linkage were actually an upper bound as well.

5.2 Other cost functions

Can similar performance guarantees be obtained under other commonly-used cost functions, for instance that of k-means (average squared distance to nearest cluster center) or k-medians (average distance to nearest cluster center)?

5.3 Efficiency

Can our algorithm, or any of the standard heuristics we have considered, be implemented in $o(n^2)$ time for data sets of size n? Results of Borodin *et al.* (1999) and Thorup (2001) offer some hope here.

Acknowledgements

The author thanks Phil Long for suggesting this problem.

Literature cited

Alizadeh, A.A., *et al.* (2000) Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503-511.

Borodin, A., Ostrovsky, R., & Rabani Y. (1999) Subquadratic approximation algorithms for clustering problems in high dimensional spaces. *ACM Symposium on Theory of Computing*.

Charikar, M. & Guha, S. (1999) Improved combinatorial algorithms for facility location and k-median problems. *IEEE Foundations of Computer Science*.

Dasgupta, S. & Schulman, L.J. (2000) A two-round variant of EM for Gaussian mixtures. *Uncertainty in Artificial Intelligence*.

Eisen, M.B., Spellman, P.T., Brown, P.O., & Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95:14863-14868.

Feder, T. & Greene, D. (1988) Optimal algorithms for approximate clustering. *ACM Symposium on Theory of Computing*.

González, T.F. (1985) Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293-306.

Hartigan, J.A. (1985) Statistical theory in clustering. *Journal of Classification*, 2:63-76.

Hochbaum, D. & Shmoys, D. (1985) A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180-184.

Kearns, M., Mansour, Y. & Ng, A. (1997) An information-theoretic analysis of hard and soft assignment methods for clustering. *Uncertainty in Artificial Intelligence*.

Thorup, M. (2001) Quick k-median, k-center, and facility location for sparse graphs. *International Colloquium on Automata, Languages, and Programming*.