Lecture 24

- More clustering
- Sublinear time algorithms
- Beyond 6.046
Last time:

\[ \text{def } P \subseteq \mathbb{R}^n \text{ is } (K,R)-\text{radius closetable} \]

if there is size \( K \)-subset \( C \subseteq P \)
such that each point in \( P \) is
within distance \( R \) of some element of \( P \).

**Fact 1** \( (K,R)-\text{r.c.} \) is \( \text{NP-} \) complete
(not shown in class)

**Fact 2** \( (K,R)-\text{r.c.} \) \( \leq \) Set Cover?

( note, Set Cover? is \( \text{NP-} \) complete
but still, this \( \leq \) does not imply Fact 1
due to the direction of the reduction)

**Fact 3** Set Cover + hence \( (K,R)-\text{r.c.} \) has an approximation algorithm.

Namely:

- If Input is \( (K,R)-\text{r.c.} \) then algorithm
  outputs a \( (K \log n, R)-\text{r.c.} \)

What if we need a small number of

Clusters?
Another approximation algorithm for clustering:

New guarantee: If points are \((K, R)\)-radius clusterable

then algorithm outputs a \((K, 2R)\)-radius clustering

(recall that we have assumed points are in \(\mathbb{R}^n\))

Algorithm:

\[
\begin{align*}
& i = 0 \\
& S \leftarrow \text{all points} \\
\text{while } & \text{S is not empty} \\
& \quad i \leftarrow i + 1 \\
& \quad \text{pick an arbitrary point } v \in S \\
& \quad \text{let the new cluster } C_i \text{ be all points within distance } 2R \text{ of } v \\
& \quad S \leftarrow S - C_i \\
\end{align*}
\]

Output \(C_1, \ldots, C_i\)

Runtime: \(O(n^2)\)

Actually - can get a better analysis
Claim After K times thru loop, S will be empty

Corollary \( O(nK) \) runtime
Output satisfies guarantee

**Pf of claim**

Let \( C^* = \langle C_1^*, ..., C_K^* \rangle \) be a \((k, R)\)-r.c.

with centers \( p_1, ..., p_k \) respectively.

For each \( C_i \),

pick any pair of pts \((u, v) \in C_i^*\)

then \( d(u, v) \leq d(u, p_i) + d(p_i, v) \)

\[ \leq 2R \quad \text{(by \( \Delta \))} \]

Once algorithm picks a \( v \in C_i^* \),
it places all other points \( u \) in \( C_i^* \) into the cluster (since \( d(u, v) \leq 2R \))

So at each round, must pick pt from a new \( C_i^* \) cluster \( \Rightarrow \) after \( K \) rounds,
no points are left! \( \square \)
What if number of objects to be clustered is huge?

(ESTIMATES)

World population 6.6 Billion

Number of grams of sand on Earth $7.5 \times 10^{18}$

Number of stars in our galaxy $O(10^{11})$

Number of stars $O(10^{21})$

Number of atoms in universe (observable) $O(10^{80})$

And you need a quick answer?

Even linear time algorithms will take a while ...

Could we hope for algorithms running in

Sublinear time?

But such algorithms can view only a small portion of the input, so they can't give a correct answer!
e.g. for clustering—1 point can make it unclusterable!

How can you find that point in sublinear time?

you can't 😞

But you can change your goal!

the idea is to allow some outliers

\[ P \text{ is } \varepsilon\text{-close to } (K,R)\text{-r.c. if can delete } \varepsilon |P| \text{ points to make it } (K,R)\text{-r.c. plus } \varepsilon\text{-far from } (K,R)\text{-r.c. otherwise.} \]
Clustering "Property Tester"

Input \( P, k, R, \epsilon \)

Output

- If \( P \) is \((k, R)\) - Rclus. output "Yes"
- If \( P \) is not \((k, 2R)\) - r.c.
- \( \epsilon \)-far from \((k, 2R)\) - r.c.

Output "NO" with probability \( \geq 2/3 \)

Comments

- Output not specified on all inputs! e.g.,
- If not \((k, R)\) - r.c. but is \((k, \frac{3}{2}R)\) - r.c. after deleting 2 points, then OK to output "Yes" or "No"
- (both answers are reasonable...)

- If repeat \( \frac{\epsilon/\ln \frac{1}{\beta}}{\ln \frac{1}{\beta}} \) times, pass only if passes each time

get that: \( P \) if \( P \) \( (k, R) \) - r.c. output Yes
- if \( P \) \( \epsilon \)-far from \((k, 2R)\) - r.c. output No
- with probability \( \geq 1 - (1 - \frac{3}{2})^\frac{\epsilon}{\ln \frac{1}{\beta}} \)
  \( \geq 1 - e^{-\ln \frac{1}{\beta}} = 1 - \beta \)

very standard trick
def Xpoint, $S$ is subset \( d(x, s) = \min d(x, y) \)

Algorithm

\[ i = 0 \]
\[ C \leftarrow \emptyset \]

Do \( k+1 \) times

\[ i \leftarrow i + 1 \]

pick sample \( S \) of size \( s = \frac{\ln 2(k+1)}{\varepsilon} \) pts from \( P \)

if there is \( x \in S \) such that \( \text{dist}(x, C) > 2R \)

then \[ C \leftarrow C \cup \{x\} \]

If \( |C| > k+1 \) output "No"

else output "Yes" + \( C \)

Runtime \( O\left(\frac{k^2 \ln k}{\varepsilon}\right)\)

Claim Algorithm satisfies property testing requirements

Proof outputs "No" only if finds \( k+1 \) points

with pairwise distance \( \geq 2R \),

which means it cannot be \((k, R)\)-r.c.

equivalently (contrapositive):

if \( P \) is \((k, R)\)-r.c. the algorithm outputs "Yes"
If \( P \) is \( \varepsilon \)-far from \( (k, 2R) \)-r.c., then will show algorithm outputs "no" with probability \( \geq \frac{2}{3} \): 

**Idea:** in each of \( k+1 \) iterations, will increase \( C \) 
\[ \Rightarrow |C| = k+1 \text{ at end} \]

\[ \text{def: } x \in P \text{ is a candidate if } d(x, C) > 2R \]

Claim if \( |C| \leq K \), must have \( > \varepsilon \cdot n \) candidates.

**Proof** assume not.

Remove all candidates.

all other pts are within \( 2R \) of some member of \( C \Rightarrow (k^2R^2) \)-r.c.

so \( P \) is \( \varepsilon \)-close to \( (k^2R^2) \)-r.c.

\[ \Rightarrow (\text{initial assumption}) \]

\[ \therefore \Pr[\text{don't find candidate in one iteration}] < (1 - \frac{\varepsilon \cdot n}{n})^2 \]

\[ < \frac{1}{e} \ln \frac{3(k^2)}{e} = \frac{1}{3k+1} \]

\[ \Rightarrow \Pr[\text{don't find candidate in some iteration}] < \frac{k^2 + \frac{1}{3k+1}}{3k+1} = \frac{1}{3} \]

if this doesn't happen, you output "no"

\[ \square \]

**Conclusion:** If clusterable, can find pretty good clustering in time independent of \( n \).
Lots of approximations can be performed in sublinear time:

properties -
linear
monotone
clusterable
bipartite
high connectivity
3-colorable

approximations -
MST
Number of connected components
diameter of point set
cluster number

Take "Sublinear Algorithms"
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.854</td>
<td>Advanced Algorithms</td>
<td>(fall)</td>
</tr>
<tr>
<td>6.856</td>
<td>Randomized Algorithms</td>
<td>(odd springs)</td>
</tr>
<tr>
<td>6.850</td>
<td>Geometric Computing</td>
<td>&quot;</td>
</tr>
<tr>
<td>6.851</td>
<td>Advanced Data Structures</td>
<td>&quot;</td>
</tr>
<tr>
<td>6.852</td>
<td>Distributed Algorithms</td>
<td>(even springs)</td>
</tr>
<tr>
<td>6.047</td>
<td>Computational Biology</td>
<td>(spring)</td>
</tr>
<tr>
<td>6.581</td>
<td>Algorithms in Systems Biology</td>
<td>(spring)</td>
</tr>
<tr>
<td>6.857</td>
<td>Network &amp; Computer Security</td>
<td>(fall)</td>
</tr>
<tr>
<td>6.896</td>
<td>Sublinear Algorithms</td>
<td>(?)</td>
</tr>
<tr>
<td>6.845</td>
<td>Automata, Computability, Complexity</td>
<td>spring</td>
</tr>
<tr>
<td></td>
<td>Theory of Computation</td>
<td>full</td>
</tr>
<tr>
<td>6.840</td>
<td>Advanced Complexity Theory</td>
<td>alternate springs</td>
</tr>
<tr>
<td>6.841</td>
<td>Randomness &amp; Computation</td>
<td></td>
</tr>
<tr>
<td>6.842</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>