Lecture 23

Clustering - a case study

and a bit more on

MST algorithms
NP - completeness & reductions
approximation algorithms

Announcement

Final: Tues Dec 18 1:30-4:30 Johnson Track
Can we find $K$ students to hold candy bowls so that each 6.046 student is within $r$ seats from a candy bowl?

This is an example of a clustering problem. Clustering is one of the most widely used tools for analyzing data sets:

- websites, search results, click stream data, customer behavioral patterns, community discovery in social networks,
- photographs, music, movies, genes +proteins, word behaviors, biological species, etc. (lots more examples!)
The scenario:
Given many objects

The goal:
Divide into groups so that
- objects in same group are close
- objects in different groups are further apart

Need to define:
- The objects
- Notions of distance (i.e., close, far)

Many formalizations

How would you cluster this?
Red clustering - might be good for computer vision to find connected objects.
Blue clustering - small number of groups
Green clustering - all elements in the same group are pretty close

What should a good clustering definition look like?
(There are millions of formulations!)

One possibility:

Hierarchical Agglomerative Clustering

A "spacings" of a $k$-clustering is the minimum distance between any pair of points lying in different clusters

Find a $k$-clustering with maximum possible spacing.

(assume $d(p_i, p_i) = 0$, $d(p_i, p_j) = 0$ for $i \neq j$ and $d(p_i, p_j) = d(p_1, p_2)$)
Hierarchical Agglomerative Clustering: Single link

Idea: "grow" a graph on the vertices (objects)
s.t. connected components \( \sim \) clusters
+ bring nearby points into same cluster quickly
  (don't want nearby points to be in different but nearby clusters) \( \Rightarrow \) seems like good idea to consider pairs in order of distance

Algorithm: (Initialization: each object is in its own cluster
\( + \) \( H \) is graph with node for each object, no edges)
1. sort pairs of objects by distance
   
   \[ (u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m) \]
   
   s.t. \( d(u_i, v_i) \leq d(u_m, v_m) \)

2. Go thru list of pairs in order from 1 to \( m \)
   - If \( u_x, v_x \) are not already in the same cluster, merge their clusters + record edge \( (u_x, v_x) \) in graph \( H \)

   \( \text{"Agglomerative" refers to merge} \)

   * Stop when there are only \( k \) clusters left

Comments:
- \( O(m \log n) \)
  * \( H \) is a union of trees (since never add edges between nodes in same cluster)
  + at end \( H \) is union of \( k \) clusters.

  * Algorithm is exactly the same as Kruskal's MST algorithm
    \[ \text{for } k=1 \] (for general \( k \): output MST minus \( k \)-th costliest edges)
Claim: Output is a k-clustering of max spacing.

Proof:
Fact: Output $C = \langle C_1 \ldots C_k \rangle$ has spacing $d^*$, which is
distance of first pair not considered
by algorithm.
Note: all edges in $H$ have distance $\leq d^*$

Consider $C' = \langle C'_1 \ldots C'_k \rangle$

$C \neq C' \Rightarrow \exists C_r$ st. not subset of any $C_s$ in $C'$

$\Rightarrow \exists p_i, p_j \in C_r$ such that $p_i \in C'_s \neq C'_r$
Output adds $p_i - p_0$ (red) path $V$ before stopping so each red edge $\leq d^*$ $\Rightarrow d(p, p') \leq d^*$ so spacing of $C' \leq d^*$ 

"... any other k-clustering has smaller spacing.

Is this the kind of clustering you want? Often used in practice..."
Another alternative:

**Min Radius Clustering**: Given \( k, R \).

Given a set of points \( P \) (assume \( P \subseteq \mathbb{R}^n \) in particular, satisfies \( \Delta^+ \)), is there a subset of points \( G \subseteq P \) of size \( k \) such that each point in \( P \) is within distance \( R \) of some element of \( G \)?

**Cost of clustering according to \( G \)**

\[
F[G] = \max_{G} \min_{p \in P} \min_{c \in G} d(p, c)
\]

Best way to assign \( P \)

Any ideas for an algorithm?
One idea:

for every \( v \) in \( P \), let

\[ S_v \leftarrow \text{all points in } P \text{ within distance } R \text{ of } v \]

\( \mapsto \) points that \( v \) would cover if chosen to be a center

gives collection of sets

\[ S_1, S_2, \ldots, S_n \]

is there a subcollection of size \( k \)

\[ S_{i_1}, S_{i_2}, \ldots, S_{i_k} \]

such that \( \bigcup_{i \in \{1, j, k\}} S_{i,j} = P \) ?

This is actually the Set Cover Problem (SC?)

Input \( k, S_1, \ldots, S_n \) s.t. \( S_i \subseteq P \)

Output Yes if \( \exists S_{i_1}, S_{i_2}, \ldots, S_{i_k} \) s.t. \( \bigcup_{i \in \{1, j, k\}} S_{i,j} = P \)

A big problem: Set Cover is \( NP \)-complete!

Does that mean \((k_r)\)-radius clustering is \( NP \)-complete?

Careful! The reduction is in the wrong direction!

(but clustering via this definition is also known to be \( NP \)-complete)
Set Cover: A short detour

Why is Set-Cover NP-complete?

Recall Vertex Cover?

\( VC \)?

Input: graph \( G \), integer \( k \)

Output: "yes" if there is a size \( \leq k \) cover \( C \)

\[ C \subseteq V \text{ s.t.} \]
\[ \forall (i,j) \in E \quad i \in C \text{ or } j \in C \]

Vertex Cover is NP-complete (lecture 22)

Fact \( VC \) \( \leq \) \( SC \) ?

Proof need \( f \) mapping \( (G,k) \) to \( (S_1,\ldots,S_m, k') \)

\( \text{s.t.} \) \( f \) has \( VC \) of size \( k \)

iff \( S_1,\ldots,S_m \) has \( SC \) of size \( k \)

How? \( (G,k) \to (S_1,\ldots,S_m, k') \)

\( S_i = \) set of edges adjacent to vertex \( i \)

Why? \( C \) is a vertex cover in \( G \)

iff \( C \) is a set cover of \( P \)
Set Cover: (More on the detour)
An approximation algorithm for set-cover:

input: $S_1 \ldots S_n$ such that $\bigcup S_i = P$

guarantee:
If there is a set cover of size $k$
then output is a cover of size $\leq k \log |P|$

algorithm:

```
while $P$ isn't covered
    add set covered max number of remaining elements
```

Why does it work?
Let $U_i \leftarrow$ uncovered elements after $i$th set added

Fact: there is always a set that covers at least $\frac{|U_i|}{k}$ new elements

(Why? $P$ can be covered by $\leq k$ sets
so $U_i \cup P$ can be covered by $\leq k$ sets
so, one of those sets must cover at least $\frac{|U_i|}{k}$ elements of $U_i$)
Using our fact:

\[ |u_{i+1}^j| \leq |u_i| - \frac{|u_i|}{k} = |u_i|(1 - \frac{1}{k}) \]

by induction, show

\[ |u_{i+1}| \leq (1 - \frac{1}{k})^i |p| \]

so if \( \lambda \geq k \log |p| \)

\[ |u_{i+1}| \leq \left(1 - \frac{1}{k}\right)^K \log |p| \cdot |p| \leq 1 \]

\[ \frac{1}{|p|} \]

so all elements covered.

Note: reduction is the wrong way for proving NP-hardness,

but the right way for using the approximation algorithm for SetCover.
An approximation algorithm for clustering:

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

then algorithm outputs a \((k\log n, r)\)-radius clustering

algorithm:

use \(O(n\log n)\)-approximation algorithm for set cover!

Note: approximation algorithm gives right radius,
but approximates \(K\) (number of centers)

What if you care about \(K\) but not \(r\)?

Coming up next...
Other clustering measures:

- diameter
- average distance to center (as opposed to max)
- not requiring point in P to be a center
- graph-based – min/sparse cuts