L9 -- Hierarchical Clustering
[Jeff Phillips - Utah - Data Mining]

What is clustering?
 one of the most ambiguous topics ever!
 - I'll ambiguously define it.
 - Then I'll formally define it.
 - Then I'll tell you why you maybe should *not* formally define it!

Let P be a data set. (perhaps in R^d, but maybe not)
let d : P x P -> |R be a metric distance on P

A cluster S is a subset of P.
Typically we find a set {S_1, S_2, ... S_k} subset P
  s.t. S_i disjoint S_j   and   union_i S_i = P

goal:
 - all for all points p_i, p_j in S
   d(p_i, p_j) is small
   "width"
 - all (most) points p_i in S_i, p_j in S_j and i!=j
   d(p_i, p_j) is large
   "split"

Want "split"/"width" large.

Draw points in plane.
Illustrate possible clusters.
Illustrate split/width.

Hierarchical/Agglomerative Clustering!

If two points are close --> put them in the same cluster.
Repeat.

Init: All points are 1 point clusters.
WHILE (2 clusters are "close enough")
  Find two "closest" clusters: S_i, S_j
  Merge clusters.

2 parts remain to be specified: "close" and "close enough"

What is "close"?

- distance between "centers" of clusters
  "center" = mean, center-point (median), center of MEB,
  some representative = min distance to other points "Non-Euclidean"
- distance between closest points
- distance between furthest points
- average distance between all pairs of points in different clusters
- lowest radius of MEB between joined cluster
- smallest average distance between point and center

** there are often ties **

What is "close enough"?

- diameter, radius of MEB, average from center beneath threshold?
  fixes scale (good/bad?)
- density beneath threshold.
  "density" = # points/volume, # points/radius^d
- joined density jumps too quickly since last time "elbow"
- when we have k clusters

Hierarchy --> Phylogenic Tree

Efficiency: (specific: closest to centroid, never stop)
  $O(n^3)$
  - $O(n)$ rounds
  - x $O(n^2)$ each round, check all pairs to find closest
  -   + $O(n)$ to recompute centroid

can reduce to $O(n^2 \log n)$: maintain priority queue of $O(n^2)$ distance
  - updates affect $O(n)$ distances, each takes $O(\log n)$ time
  - $O(n)$ rounds | updates

----

k-center clustering
  "Gonzalez Algorithm 85"

"HAC" one form of greedy. Different form of greedy.

```
  --> be greedy, but be smart and greedy :)


k-center clustering:
  Find k points C = {c_1, ..., c_k}, s.t.
   - each p \in P assigned mu(p) = arg min_{c in C} d(p,c)
   - minimize max_{p in P} d(p, mu(p))

(like k-means   minimize sum_{p in P} d(p,mu(p))^2  )
(     k-median  minimize sum_{p in P} d(p,mu(p))    )

k-center cluster optimally is NP-Hard.
   better than 2-appox  -->  also NP-Hard !!!

-----------
Choose first c_1 arbitrarily
  C_1 = {c_1}    (generally C_i = {C_1, C_2, ..., C_i}  \\ goal C_k)

Let c_{i+1} = arg max_{p in P \ C_i} d(p,mu(p))
   "always pick point furtherest from set of centers C_i"
----------

2-approx to optimal algorithm (worst case).  Often much better.

O(k^2 n)  O(k) rounds x O(kn) per round
  ---
O(kn) : maintain mu(p)
 O(k) rounds
  - maintain mu(P)
  - on new c_i, spend O(n) to check each point if closer,
    update t_j = max_{p \in P \ C_i} d(p,mu(p)) s.t. mu(p) = c_j
       for each c_j \in C_i
    update t = max_j t_j


*** Works for any metric.
*** Biases centers to "edge" of data set.
    - heuristic to recenter:  after run, find "clusteroid" of mu^{-1}(c_j) as
new c_j
```