Sept 27th

HIERARCHICAL CLUSTERING

OUTPUT: dendogram / tree of clusters showing closest related points.
NOTE: we can cut at a specific threshold to generate different clusters.

- agglomerative → every point in own cluster, each step merge two closest cluster. Stop when every point in own cluster
- divisive → keep splitting until every point in own cluster.

AGGLOMERATIVE CLUSTERING ALGO

1. each point in dataset be in its own cluster
2. Compute the distance b/w all pair of clusters
3. merge 2 closest clusters
4. Repeat 3 and 4 until all points are in the same cluster

cluster distance options
b/w centroid
shortest path.

NOTE: $d \rightarrow$ b/w points
$D \rightarrow$ b/w clusters.

SINGLE LINK DISTANCE

min of all pairwise distance b/w points from one cluster and one from another cluster.

$$D_{SL}(C_1, C_2) = \min\left(d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\right)$$

NOTE: depends on choice d.

- ✓ can handle clusters of diff size
- ✗ does not do well w/ overlaps
- ✗ does not handle noise

COMPLETE LINK DISTANCE

max of all pairwise distance b/w points from one cluster and one from another cluster.

$$D_{CL}(C_1, C_2) = \max\left(d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\right)$$

NOTE: depends on choice d.

- ✓ can handle noise
- ✗ doesn't do well w/ overlaps
- ✓ does well w/ noise

AVERAGE LINK DISTANCE

avg of all pairwise distance b/w a points from one cluster and one from another cluster.

$$D_{AL}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|}\left(d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\right)$$

NOTE: depends on choice d.

CENTROID DISTANCE

$$D_C(C_1, C_2) = d(\mu_1, \mu_2)$$

WARD'S DISTANCE

diff b/w spread/var of points merged cluster + unmerged clusters.

$$D_{WD}(C_1, C_2) = \sum_{p \in C_{12}} d(p, \mu_{12}) - \sum_{p_1 \in C_1} d(p_1, \mu_1) - \sum_{p_2 \in C_2} d(p_2, \mu_2)$$

DENSITY BASED CLUSTERING.

GOAL: cluster together points that are densely packed together

define density: given fixed radius ε, if it has more than a min number of points, we can categorize it as dense.

DBScan- looks at points at core (in ε neighborhood w/ min pts) + perimeter (border) (in ε neighbor of core) or noise point (neither of previous).

Create clusters by connecting data points

✓ handles oddly shaped clusters

DBSCAN ALGO

Given: ε, min_pts

1. Find ε neighborhood of each pt
2. Label core if ≥ min_pts
3. Not core, but in neighborhood = border
4. else noise
5. For each core, assign to same cluster all core points in its neighborhood
6. Assign border points to nearby clusters.

✓ can identify clusters of diff shape/size
✓ resistant to noise

✗ limit is varying density
✗ created of same density.
✗ highly problematic in high-dim space.