CMSC 471

ML: Clustering

Unsupervised Learning

- Supervised learning used labeled data pairs (x, y) to learn a function f : X→y
- But, what if we don't have labels?
- No labels = **unsupervised learning**
- Only some points are labeled = **semi-supervised learning**
  - Getting labels is expensive, so we only get a few
- **Clustering** is the unsupervised grouping of data points based on similarity
- It can be used for **knowledge discovery**

Clustering algorithms

- Many clustering algorithms
- Clustering typically done using a **distance measure** defined between instances or points
- Distance defined by instance **feature space**, so it works with numeric features
  – Requires encoding of categorial values; may benefit from normalization
- We'll look at three popular approaches

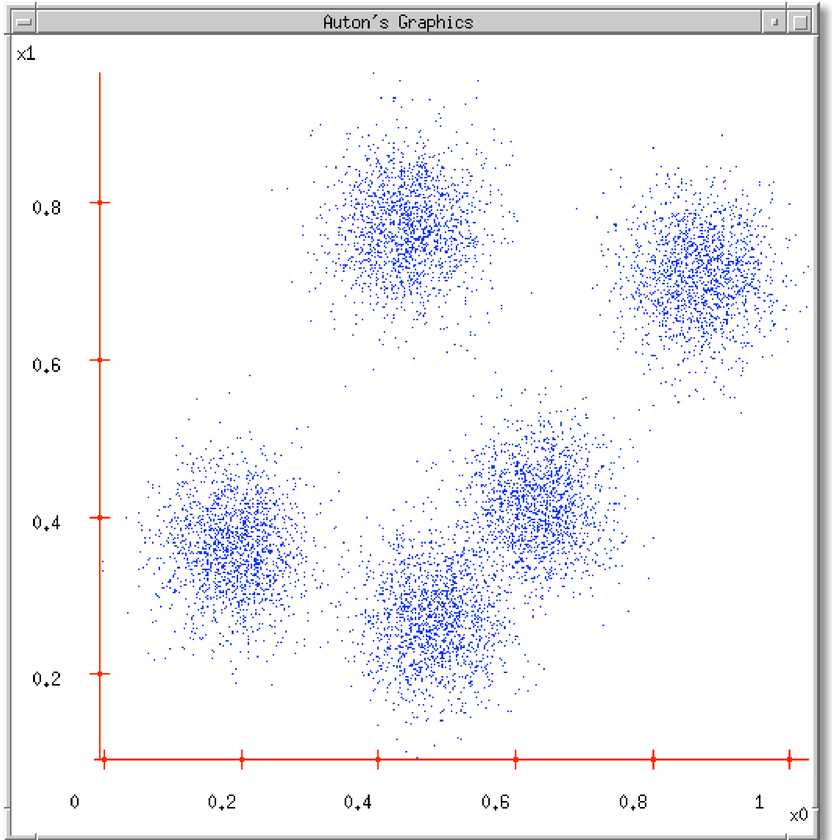
1. Centroid-based clustering
2. Hierarchical clustering
3. DBSCAN

Clustering Data

Given a collection of points (x,y), group them into one or more clusters based on their distance from one another

How many clusters are there?

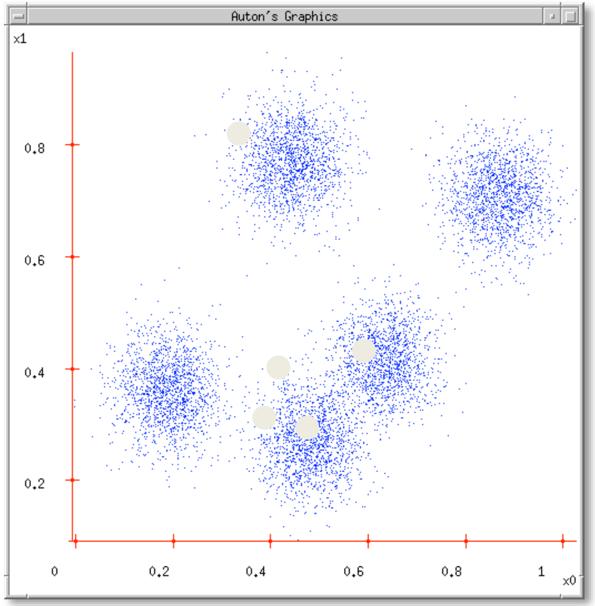
How can we find them

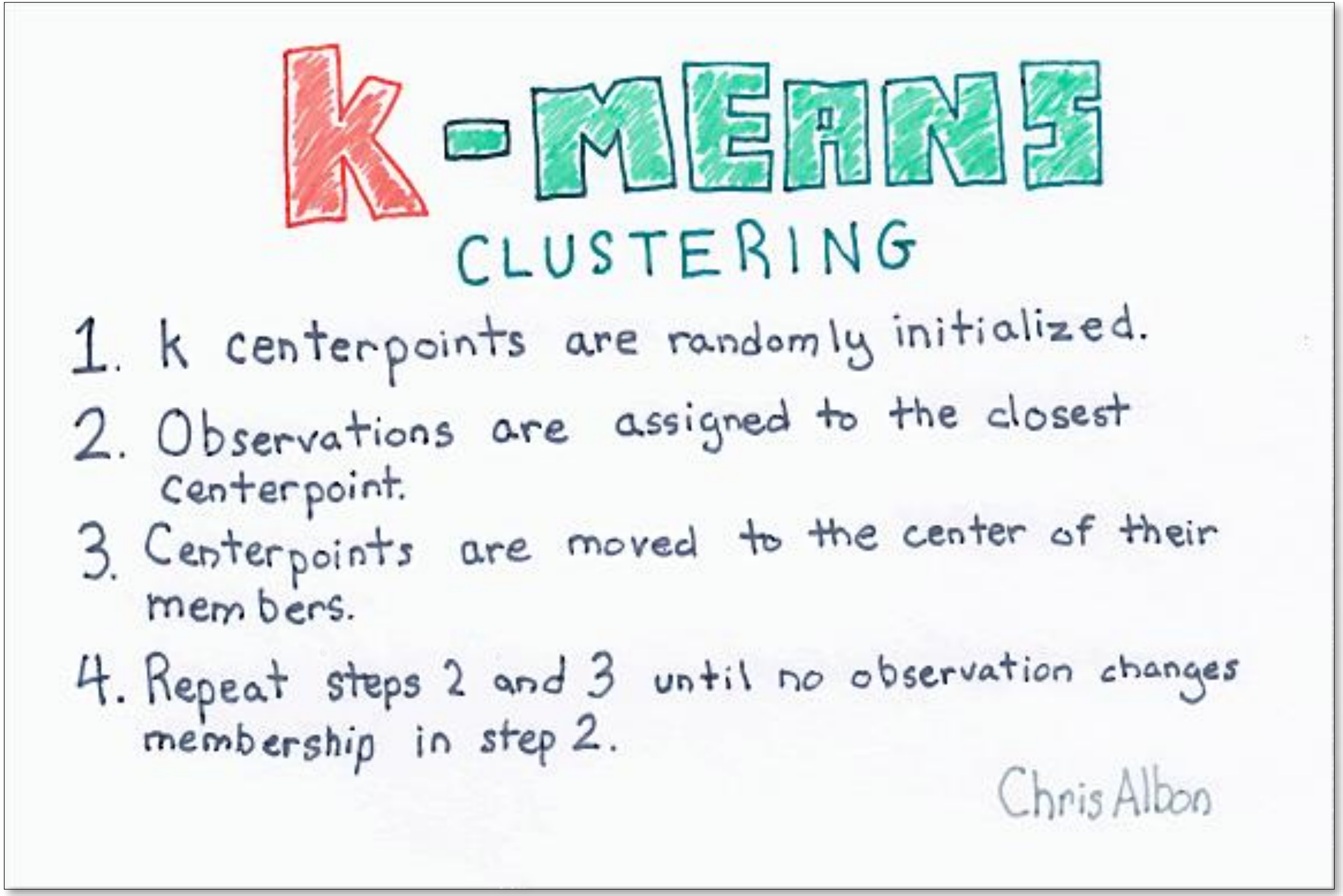


(1) K-Means Clustering

- Randomly choose k cluster center locations, aka **centroids**
- Loop until convergence
  - assign a point to cluster of closest centroid
  - re-estimate cluster centroids based on its data assigned
- Convergence: no point is re-assigned to a different cluster

k = 5




k-MEANS
CLUSTERING
1. k centerpoints are randomly initialized.
2. Observations are assigned to the closest centerpoint.
3. Centerpoints are moved to the center of their members.
4. Repeat steps 2 and 3 until no observation changes membership in step 2.
Chris Albon

distance, centroids

- Distance between points (X_0,Y_0,Z_0) and (X_1,Y_1,Z_1) is just $\mathrm{sqrt}((X_0- X_1)^2 + (Y_0- Y_1)^2 + (Z_0- Z_1)^2)$
- In numpy

```
>>> import numpy as np
>>> p1 = np.array([0,-2,0,1]) ; p2 = np.array([0,1,2,1]))
>>> np.linalg.norm(p1 - p2)
3.605551275463989
```

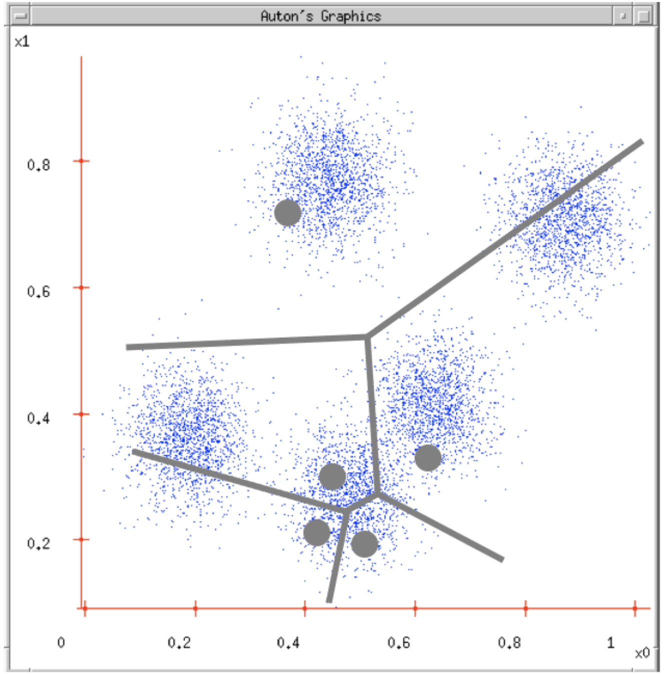
- Computing centroid of set of points easy

```
>>> points = np.array([[1,2,3], [2,1,1], [3,1,0]])    # 3D points
>>> centroid = np.mean(points, axis=0)        # get mean across columns
>>> centroid
array([2.0, 1.33, 1.33])
```

K-Means Clustering

K-Means (k , data)

- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid.
  - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster

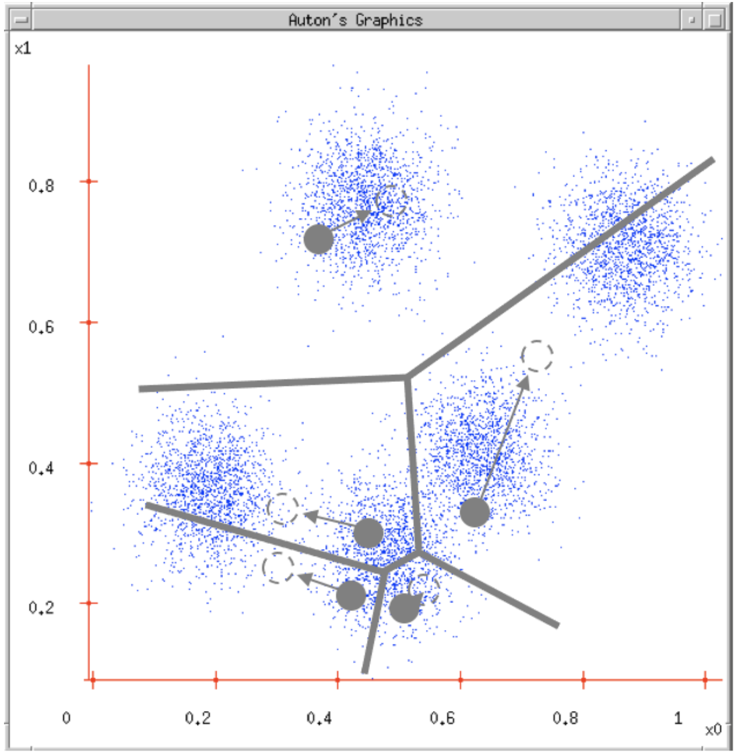


veroni diagram: add lines for regions of points closest to each centroid

K-Means Clustering

K-Means (k , data)

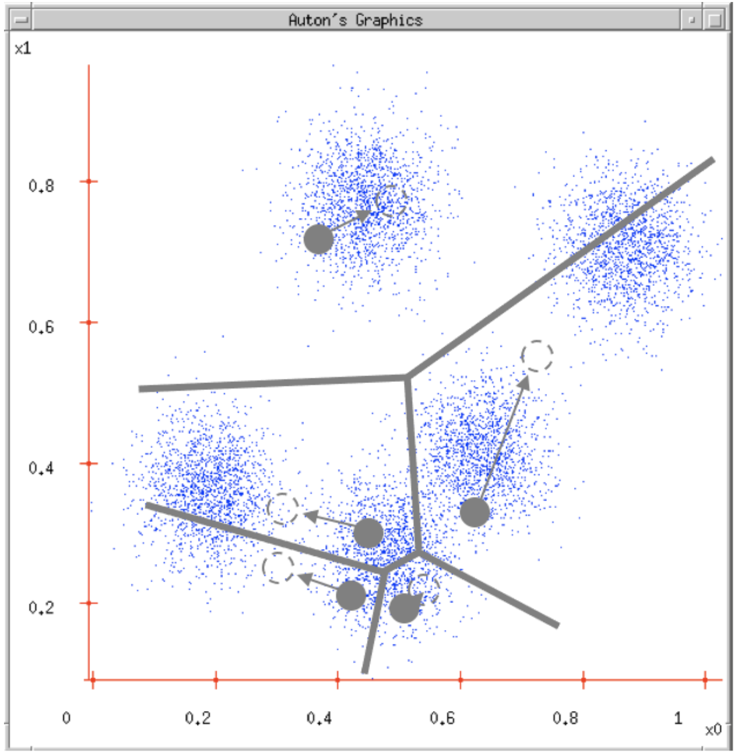
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each
- Convergence: no point is assigned to a different cluster



K-Means Clustering

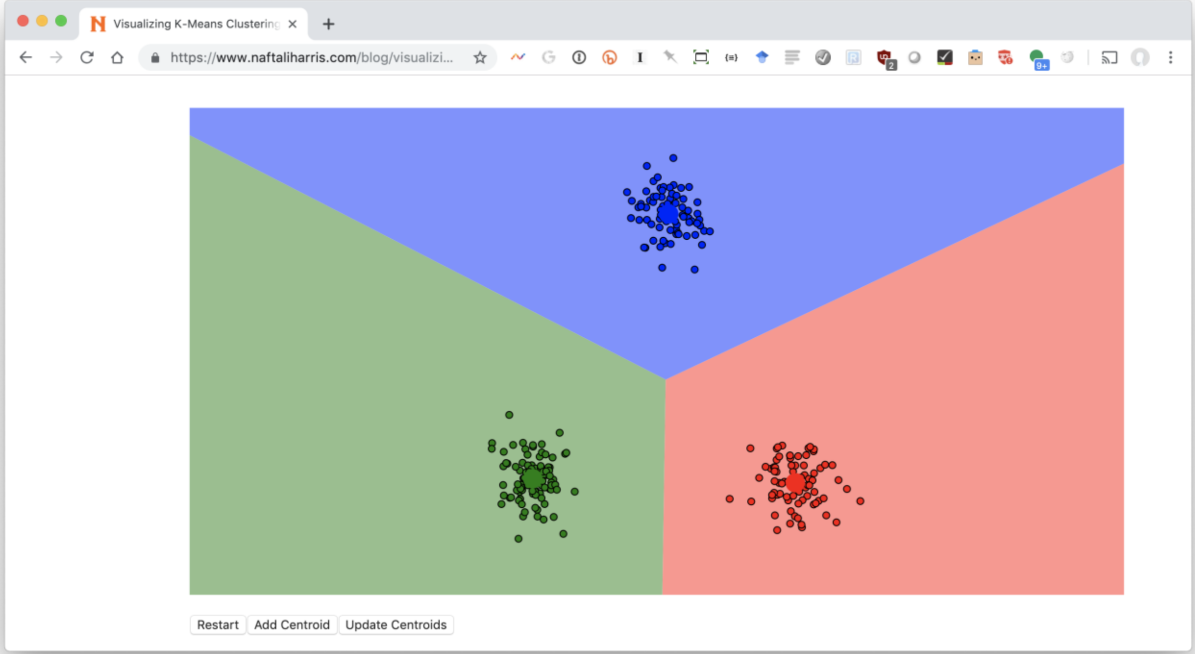
K-Means (k , data)

- Randomly choose k cluster center locations (centroids)
- Loop until convergence
    - Assign each point to the cluster of the closest centroid
    - Re-estimate the cluster centroids based on the data assigned to each
- **Convergence: no point is assigned to a different cluster**



Visualizing k-means:

http://bit.ly/471kmean



Problems with K-Means

- Only works for numeric data (typically reals)
- ***Very*** sensitive to the initial points
  - **fix:** Do many runs, each with different initial centroids
  - **fix:** Seed centroids with non-random method, e.g., **farthest-first** sampling
- Sensitive to outliers
  - **E.g.: find three**
  - **fix:** identify and remove outliers
- Must manually choose k
  - Learn optimal k using some performance measure

(2) Hierarchical clustering

- **Agglomerative**
  - **Bottom-up** approach: elements start as individual clusters & clusters are merged as one moves up the hierarchy
- **Divisive**
  - **Top-down** approach: elements start as a single cluster & clusters are split as one moves down the hierarchy

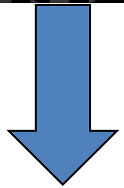
Hierarchical clustering advantages

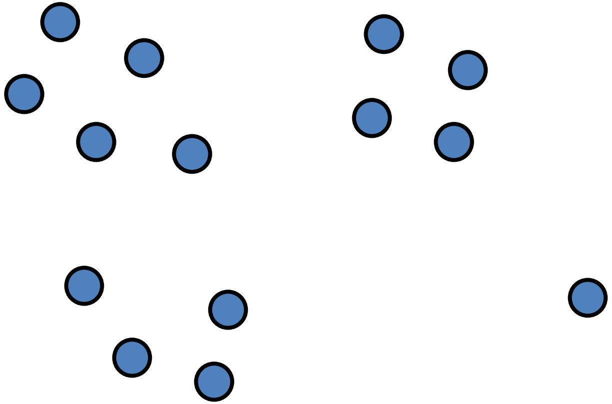
- Need not specify number of clusters
- Good for data visualization
  - See how data points interact at many levels
  - Can view data at multiple granularity levels
  - Understand how all points interact
- Specifies all of the K clusterings/partitions

Divisive hierarchical clustering

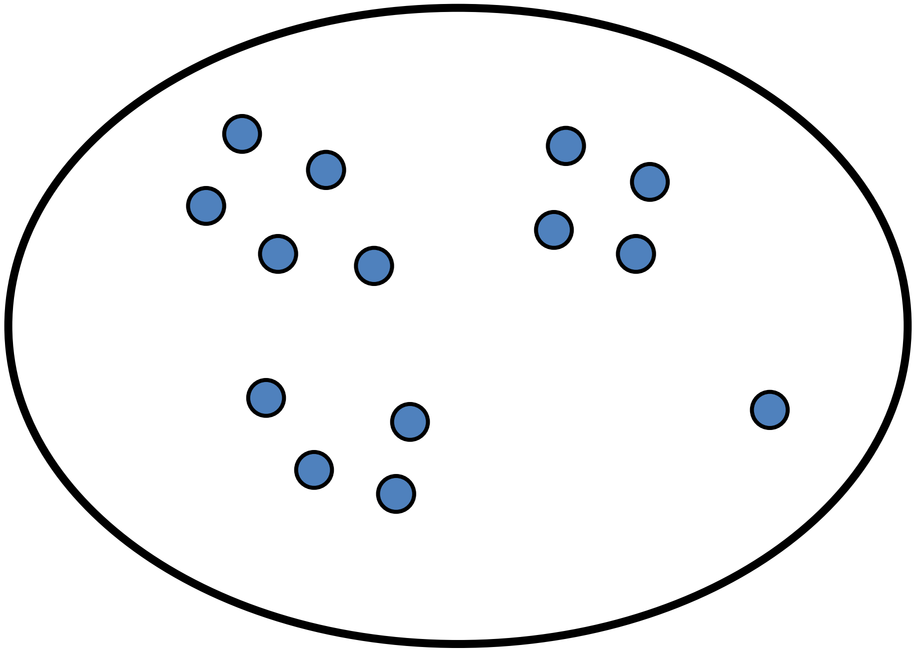
- Top-down technique to find best partitioning of data, generally exponential in time
- Common approach:
  - Let **C** be a set of clusters
  - Initialize **C** to be a one-clustering of data
  - While there exists a cluster *c* in **C**
    - remove *c* from **C**
    - partition *c* into 2 clusters (c_1 and c_2) using a flat clustering algorithm (e.g., k-means)
    - Add to c_1 and c_2 **C**
- Bisecting k-means

Divisive clustering



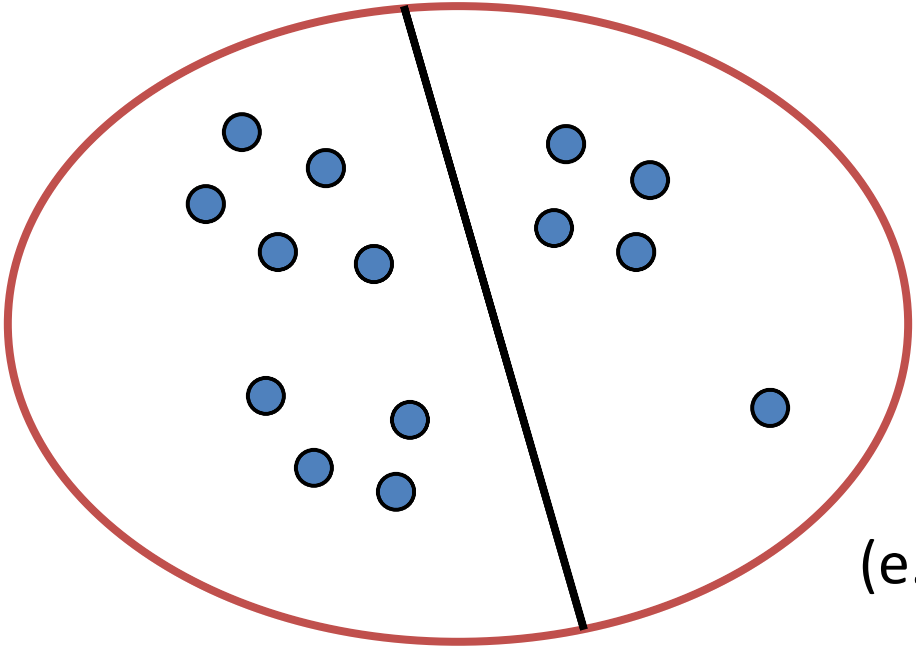


Divisive clustering



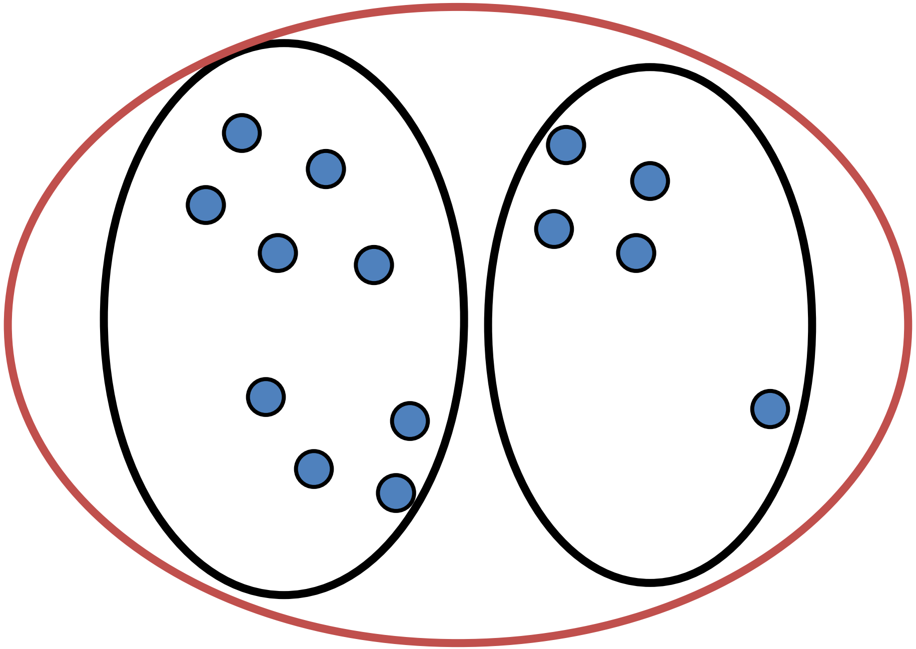
start with one cluster

Divisive clustering

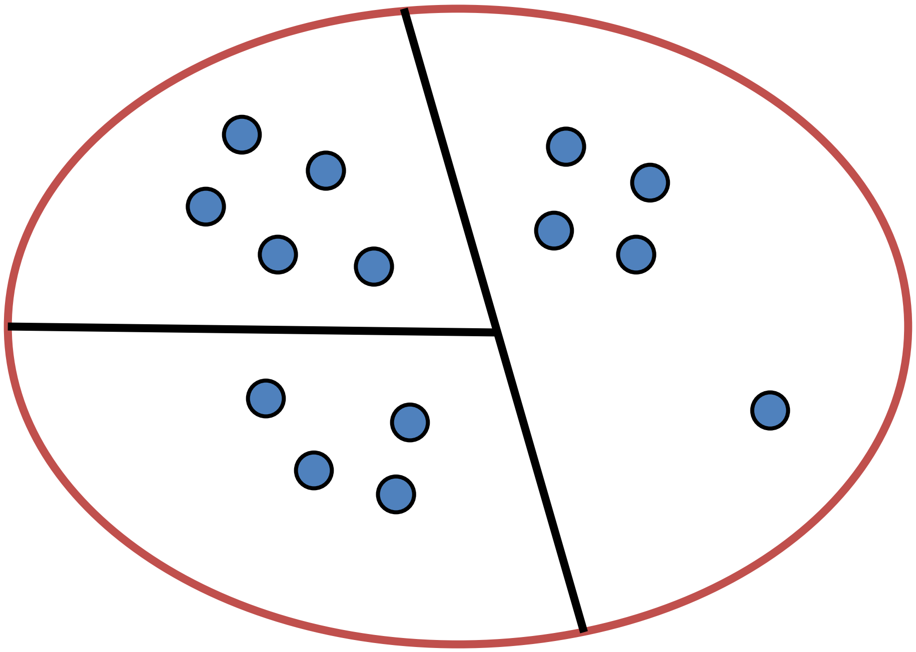


use flat clustering to split into two clusters (e.g., using K-means with k=2)

Divisive clustering



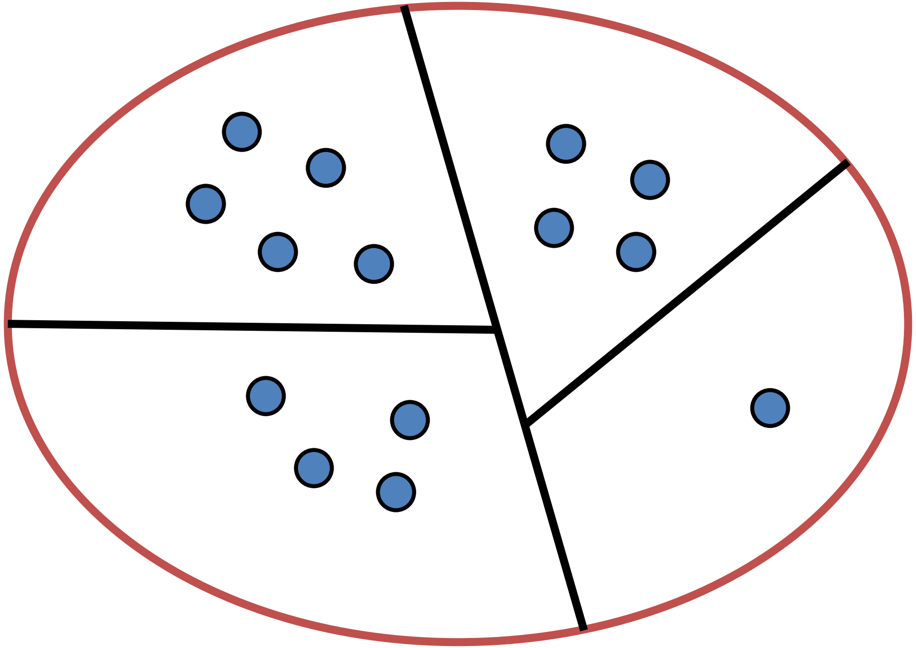
Divisive clustering



split using flat clustering, e.g., Kmeans

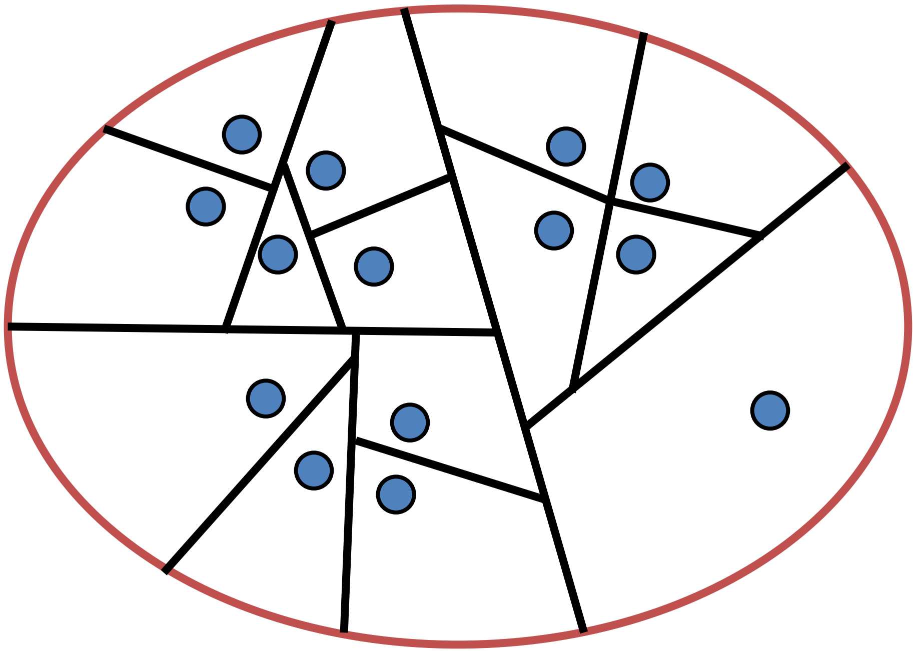
Divisive clustering

split using flat clustering



split using flat clustering, e.g., Kmeans

Divisive clustering



Stop when clusters reach some constraint

AGGLOMERATIVE CLUSTERING

All observations start as their own cluster. Clusters meeting some criteria are merged. This process is repeated, growing clusters until some end point is reached.

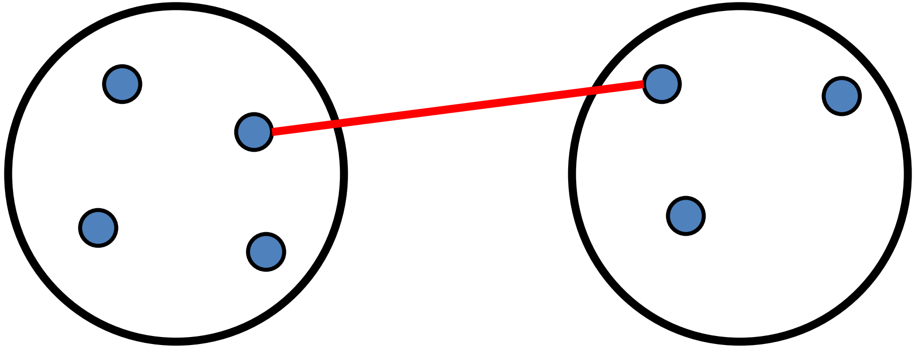
Chris Albon

Hierarchical Agglomerative Clustering

- Let **C** be a set of clusters
- Initialize **C** to all points/docs as separate clusters
- While **C** contains more than one cluster
  - find c_1 and c_2 in **C** that are **closest together**
  - remove c_1 and c_2 from **C**
  - merge c_1 and c_2 and add resulting cluster to **C**
- Merging history forms a binary tree or hierarchy
- **Q: How to measure distance between clusters?**

Distance between clusters

Single-link: Similarity of the *most* similar (single-link)



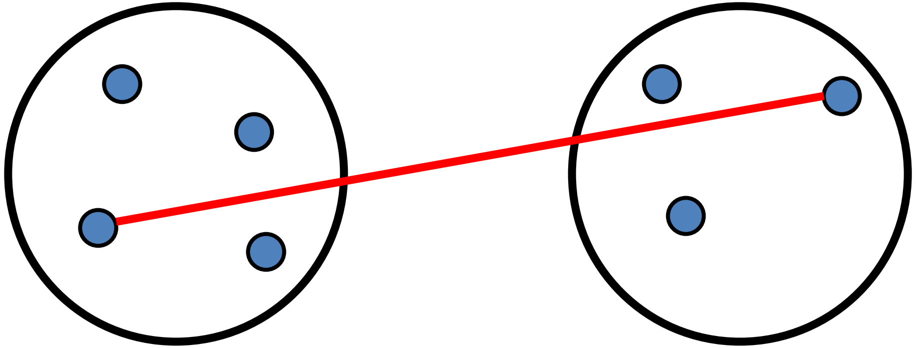
$$\max_{l \in L, r \in R} sim(l,r)$$

Weka: linkType=SINGLE

Distance between clusters

Complete-link: Similarity of the “furthest” points, the *least* similar

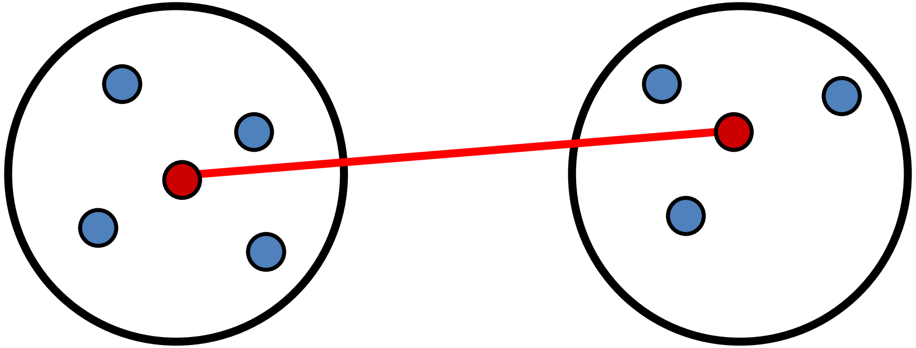
$$\min_{l \in L, r \in R} sim(l,r)$$



Weka: linkType=COMPLETE

Distance between clusters

Centroid: Clusters whose centroids (centers of gravity) are the most similar

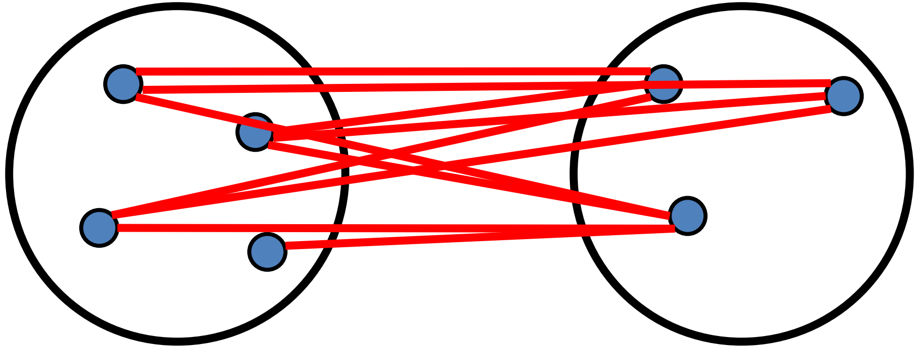


$$\left\| \mu(L) - \mu(R) \right\|^2$$

Weka: linkType=CENTROID

Distance between clusters

Average-link: Average similarity between all pairs of elements



$$\frac{1}{|L| \cdot |R|} \sum_{x \in L, y \in R} \|x - y\|^2$$

Weka: linkType=AVERAGE

Knowing when to stop

- General issue is knowing when to stop merging/splitting a cluster
- We may have a problem specific desired range of clusters (e.g., 3-6)
- There are some general metrics for assessing quality of a cluster
- There are also domain specific heuristics for cluster quality

(3) DBSCAN Algorithm

- Density-Based Spatial Clustering of Applications with Noise
- Clusters close points based on a distance and a minimum number of points
  - Key parameters: eps=maximum distance between two points; minPoints= minimal cluster size
- Marks as outliers points in low-density regions
- Needn’t specify number of clusters expected
- Fast

UMBC

DBSCAN

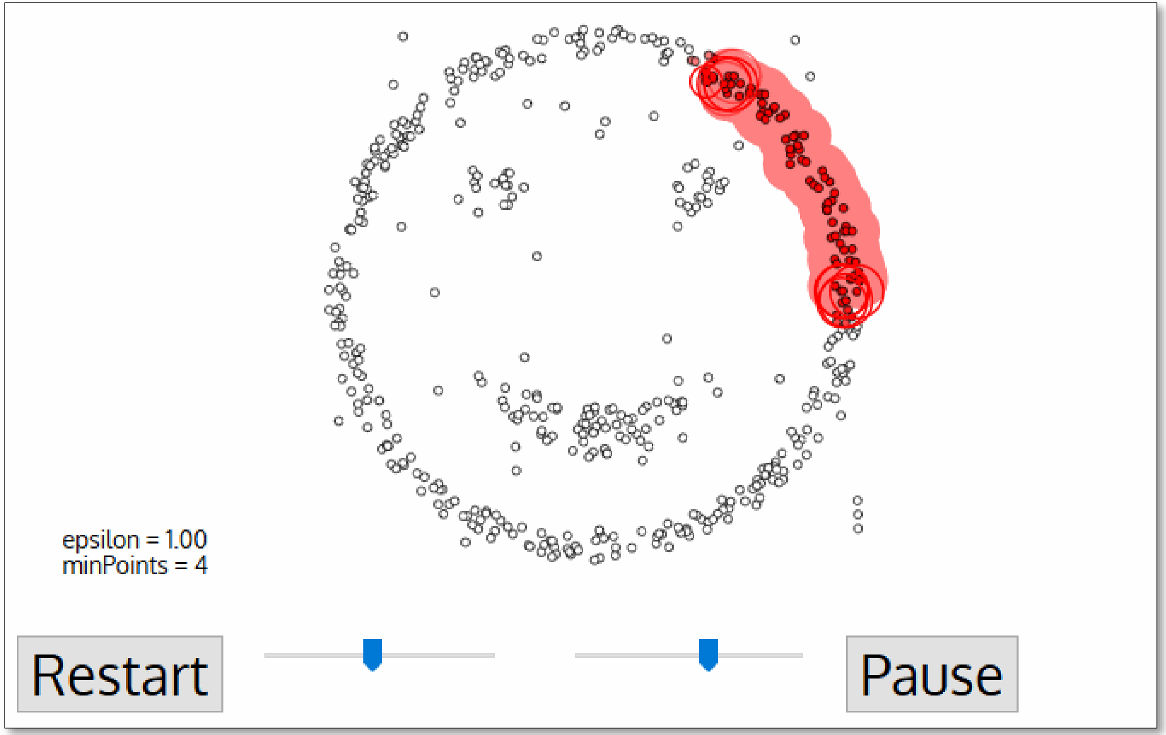
DBSCAN looks for densely packed observations and makes no assumptions about the number or shape of clusters.

1. A random observation, x_i, is selected
2. If x_i has a minimum of close neighbors, we consider it part of a cluster.
3. Step 2 is repeated recursively for all of x_i's neighbors, then neighbors' neighbors etc... These are the cluster's core members.
4. Once Step 3 runs out of observations, a new random point is chosen

Afterwards, observations not part of a core are assigned to a nearby cluster or marked as outliers.

ChrisAlbon

DBSCAN Example



This gif (in ppt) shows how DBSCAN grows four clusters and identifies the remaining points as outliers