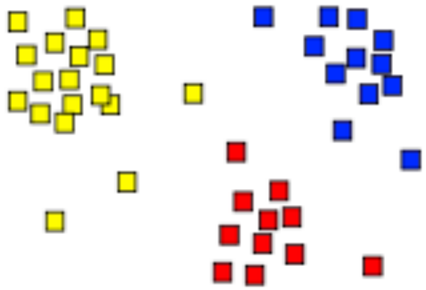
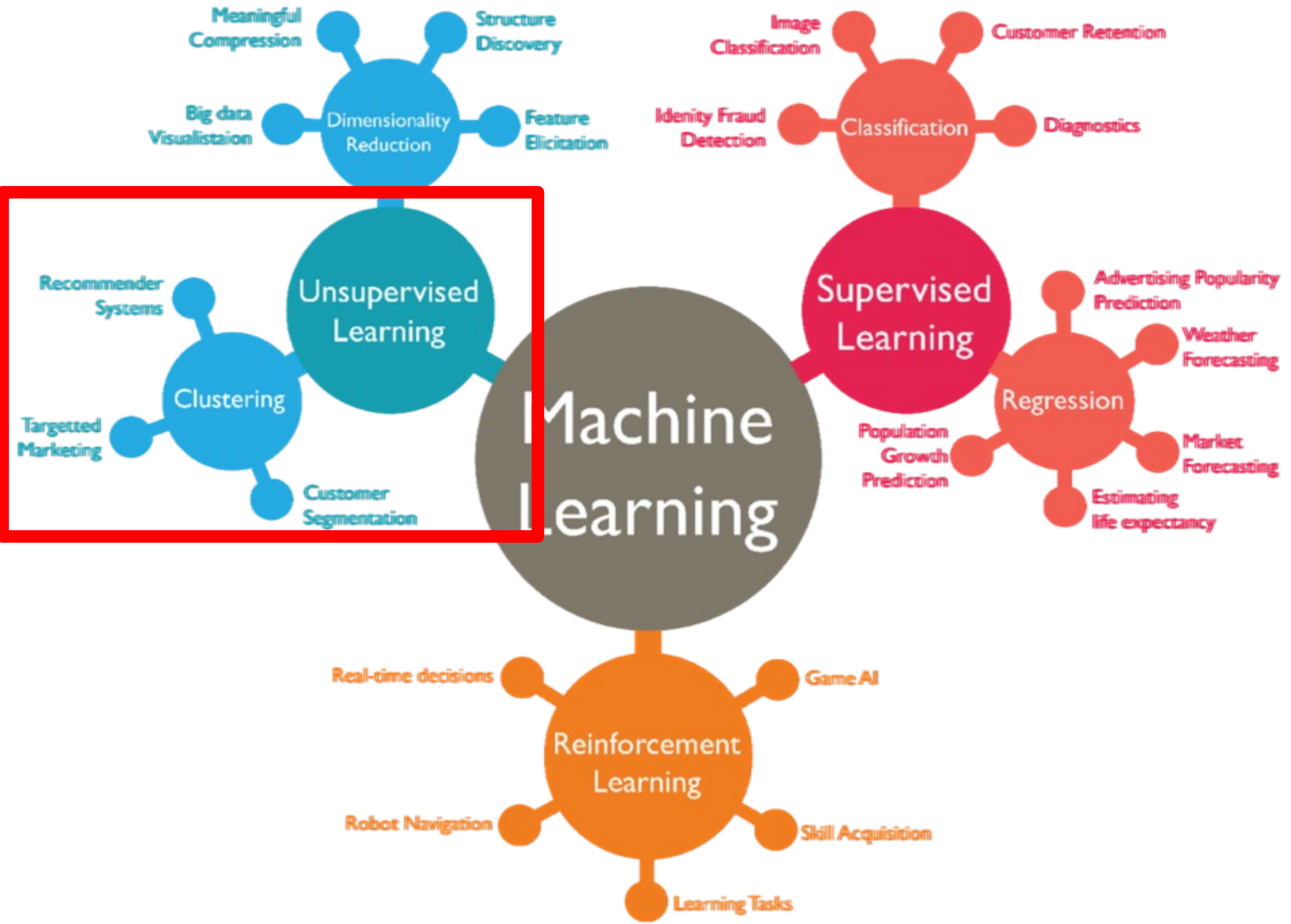
14.6



Unsupervised Learning: Clustering

Beyond K-means

Some material adapted from slides by Andrew Moore, CMU



Meaningful Compression
Structure Discovery
Big data Visualistaion
Dimensionality Reduction
Feature Elicitation
Image Classification
Customer Retention
Idenity Fraud Detection
Classification
Diagnostics
Recommender Systems
Unsupervised Learning
Supervised Learning
Advertising Popularity Prediction
Weather Forecasting
Clustering
Regression
Targetted Marketing
Machine Learning
Population Growth Prediction
Market Forecasting
Customer Segmentation
Estimating life expectancy
Real-time decisions
Game AI
Reinforcement Learning
Robot Navigation
Skill Acquisition
Learning Tasks

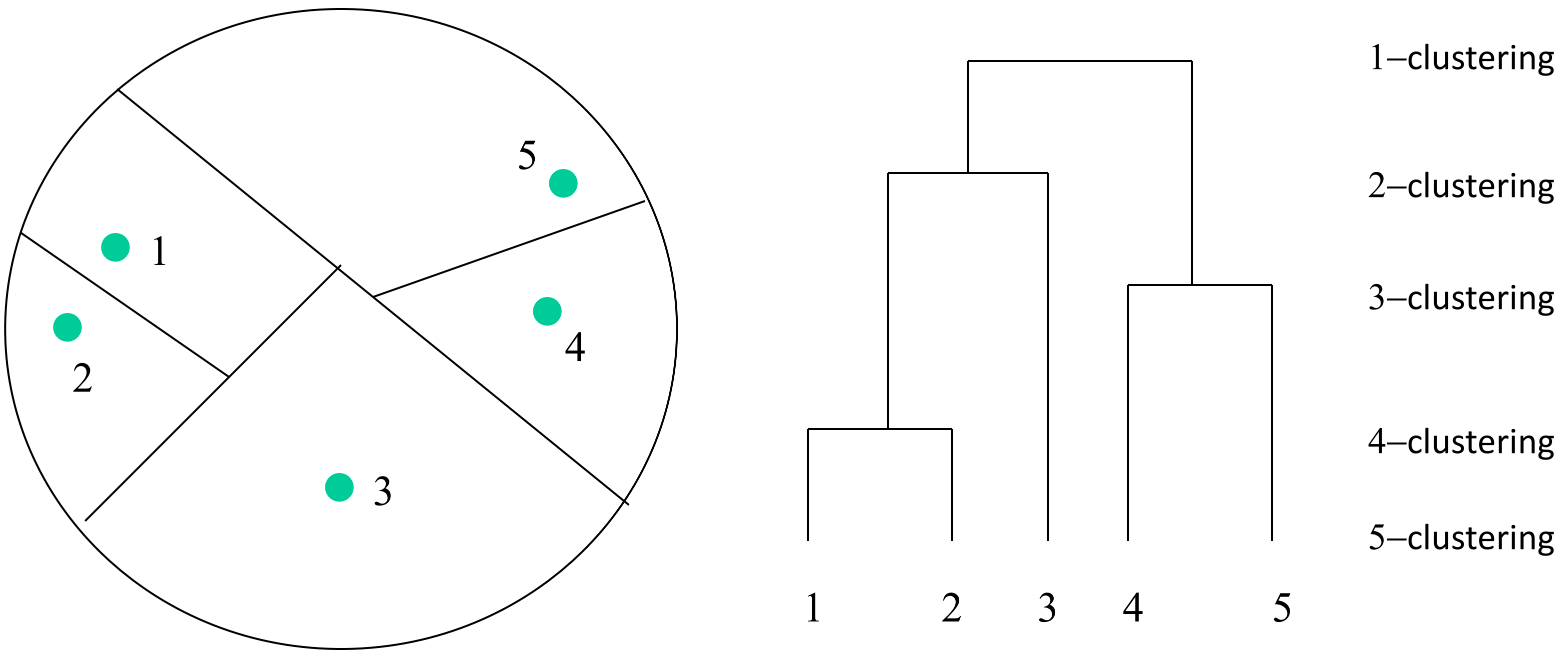
(2) Hierarchical clustering

Two approaches:

- **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

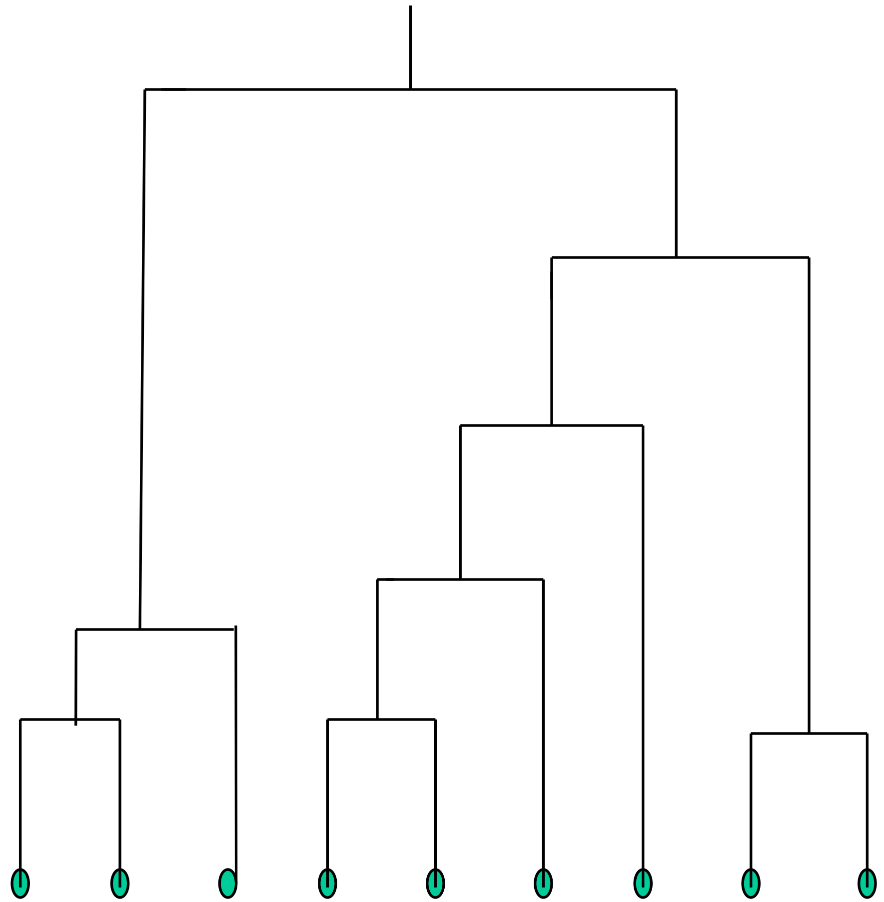
Hierarchical Clustering

The approaches do a recursive partitioning / merging of a data set



Dendogram

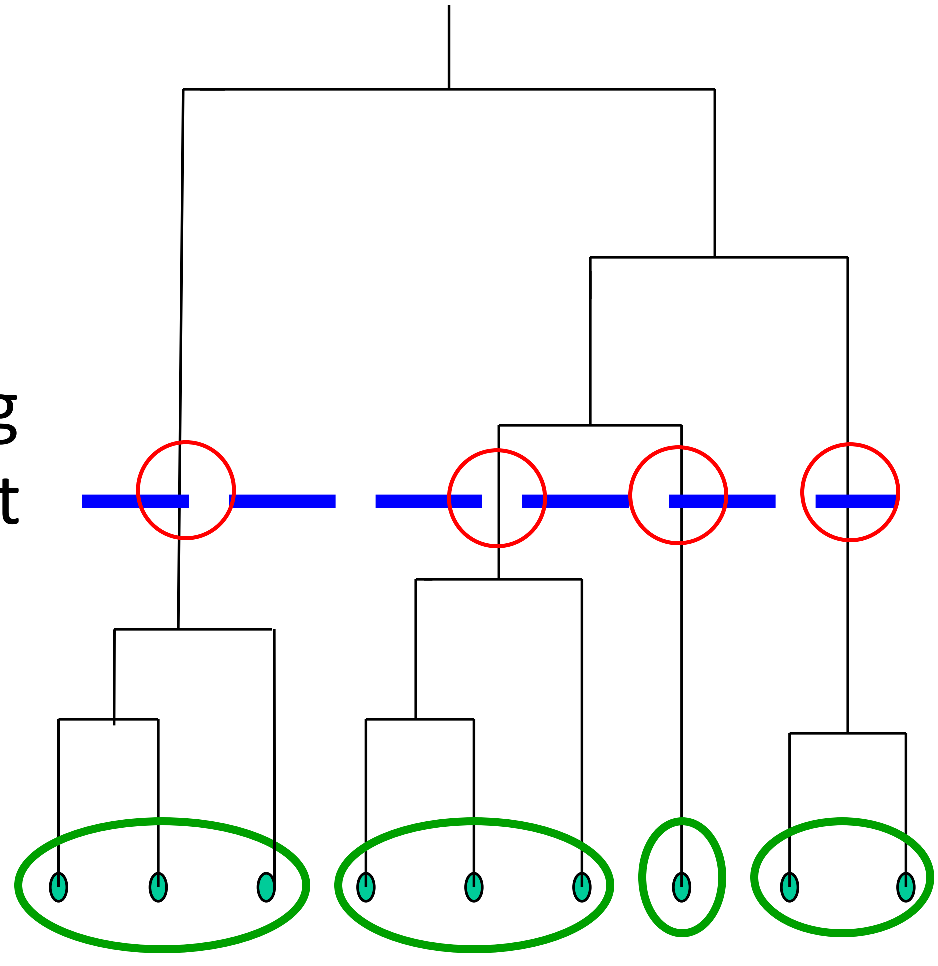
- Tree structure representing all data partitionings
- Constructed as clustering proceeds



Nine items

Dendogram

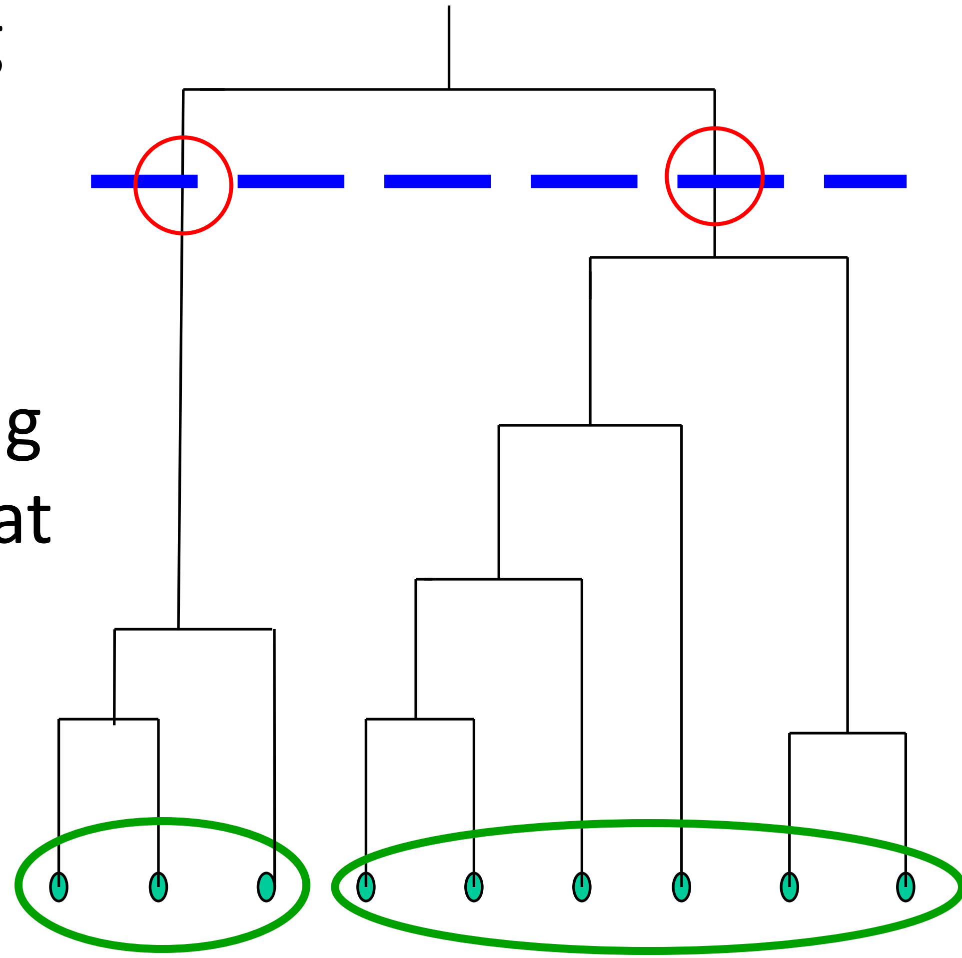
- Tree structure representing all data partitionings
- Constructed as clustering proceeds
- Get a **K-clustering** by looking at **connected** components at any given level
- Often binary dendograms, but n-ary ones easy to get with minor algorithm changes



Four clusters at this level

Dendogram

- Tree structure representing all data partitionings
- Constructed as clustering proceeds
- Get a K-clustering by looking at **connected** components at any given level
- Often binary dendograms, but n-ary ones easy to get with minor algorithm changes



Two clusters at this level

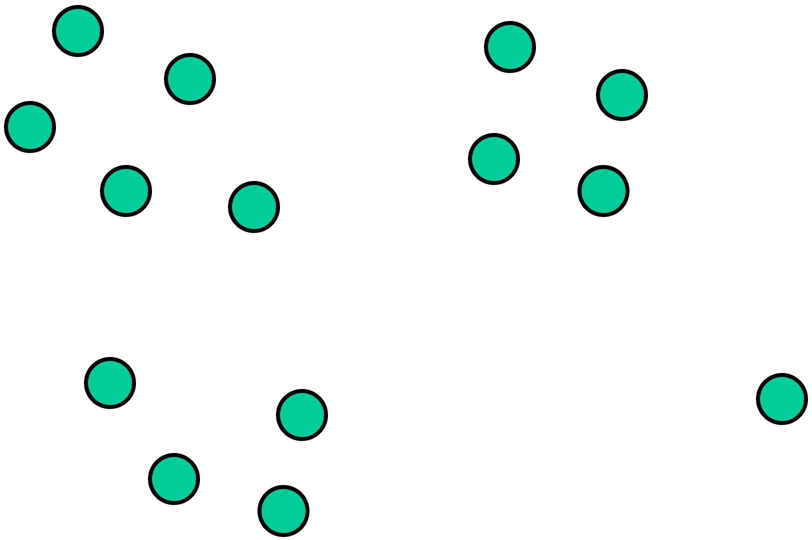
Hierarchical clustering advantages

- Need not specify number of clusters
  - You can get from 1 to n given n data points
- Good for data visualization
  - See how data points interact at many levels
  - Can view data at multiple granularity levels
  - Understand how all points interact
- Can generate all the K clusterings/partitions
- But which is the best clustering?
  - Algorithms using homogeneity measures of the clusters are often used

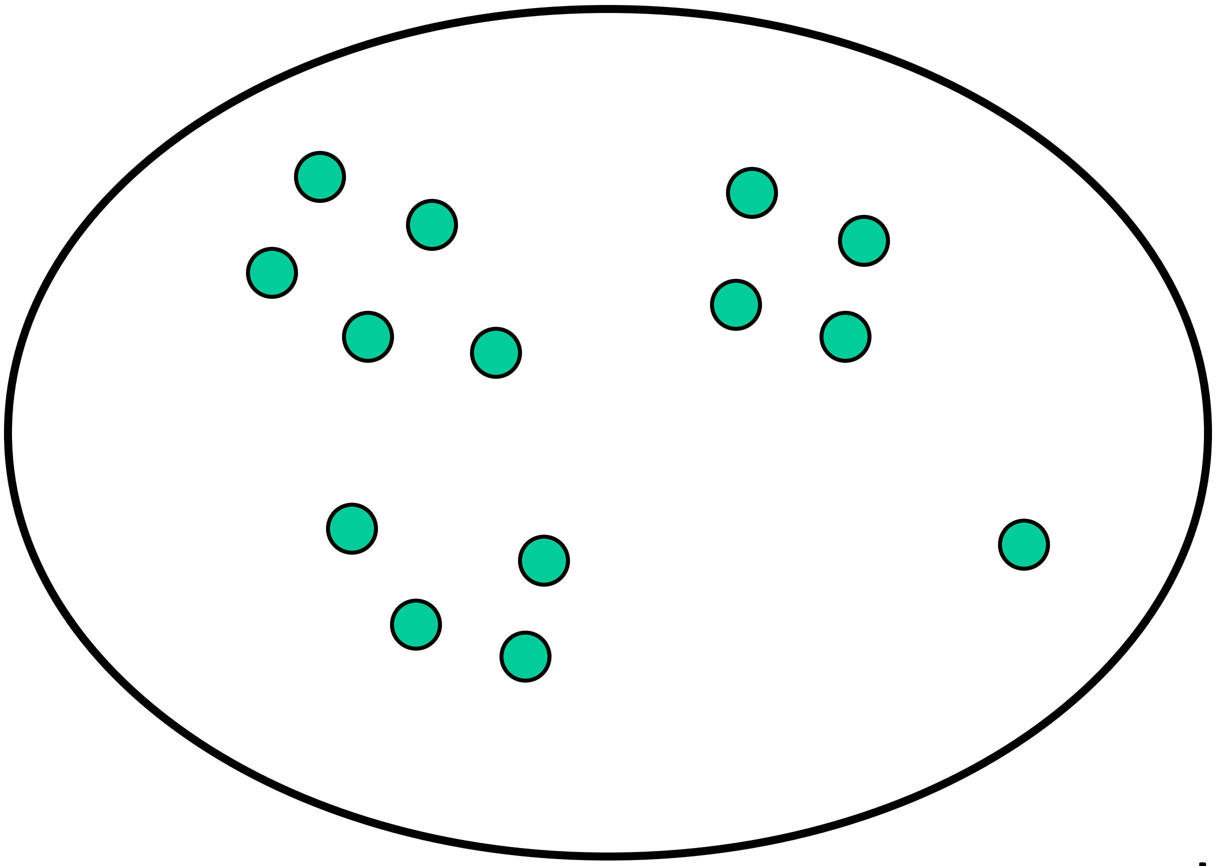
Divisive hierarchical clustering

- 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 with k=2)
    - Add to c_1 and c_2 **C**

Divisive clustering

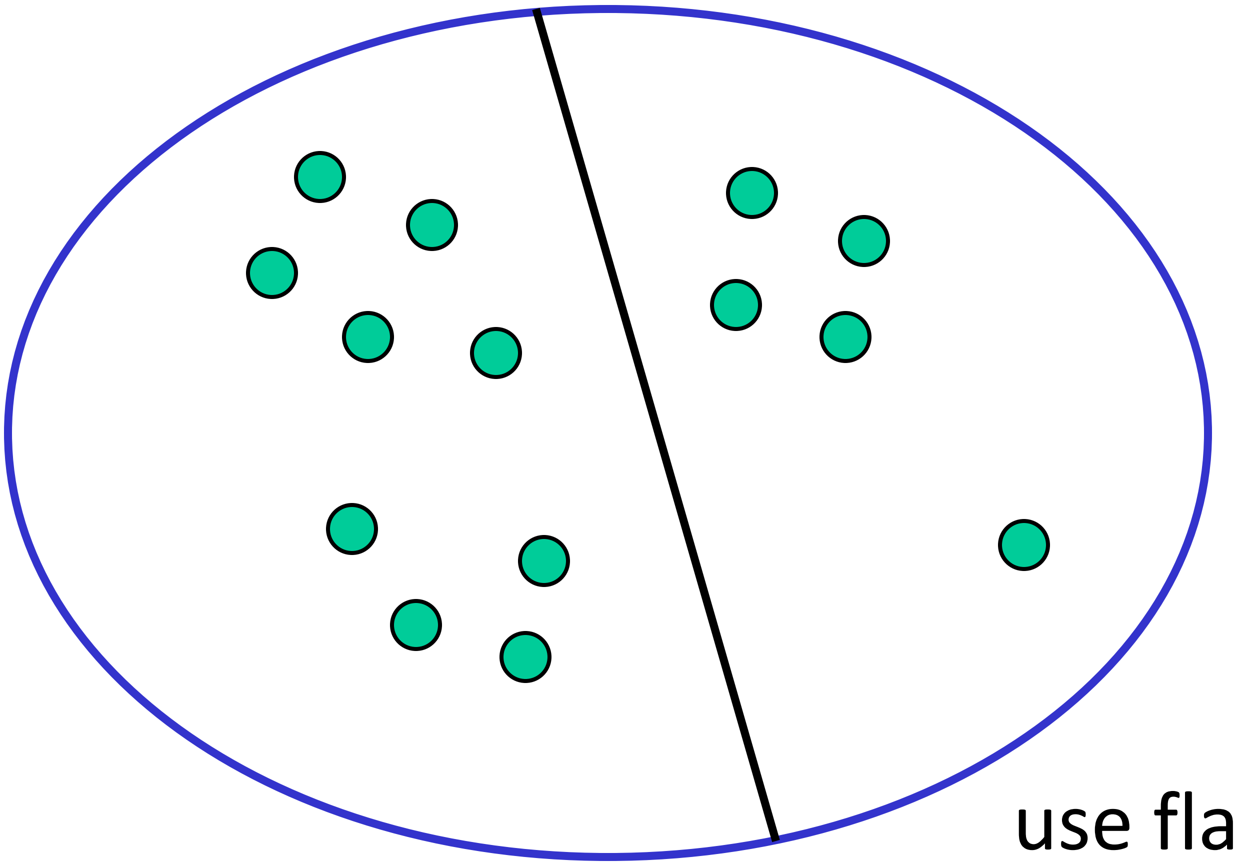


Divisive clustering



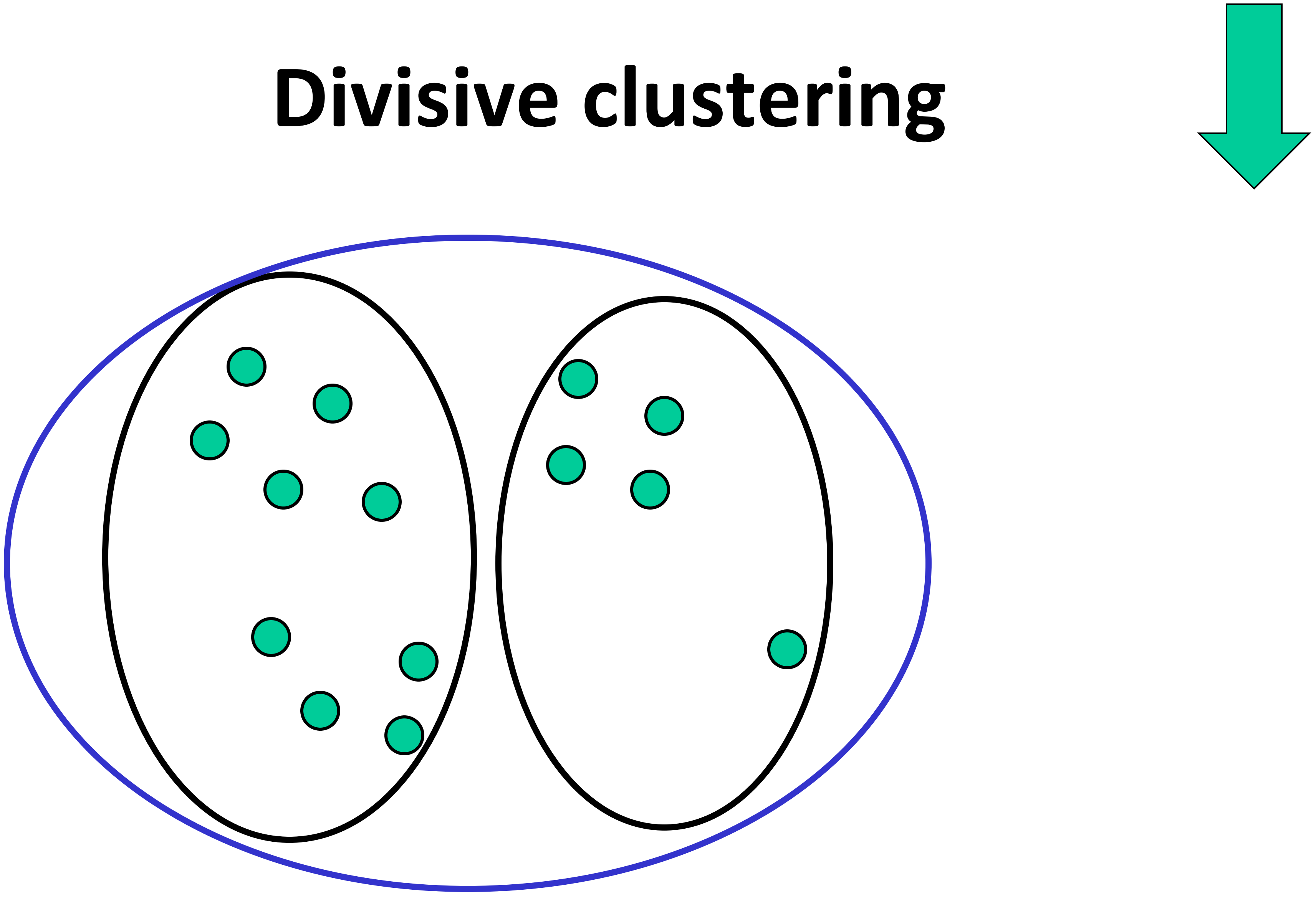
start with one cluster

Divisive clustering

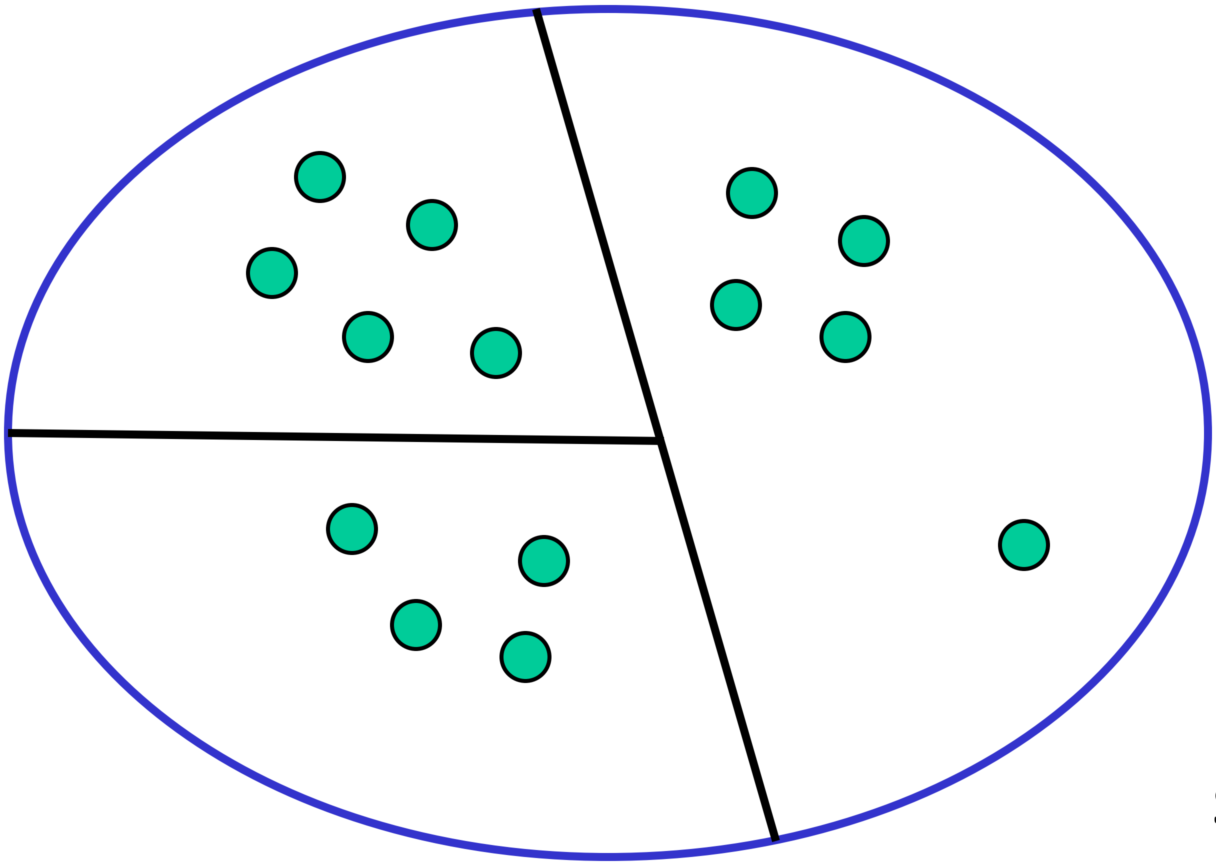


use flat clustering to
split into two clusters (e.g.,
using K-means with k=2)

Divisive clustering

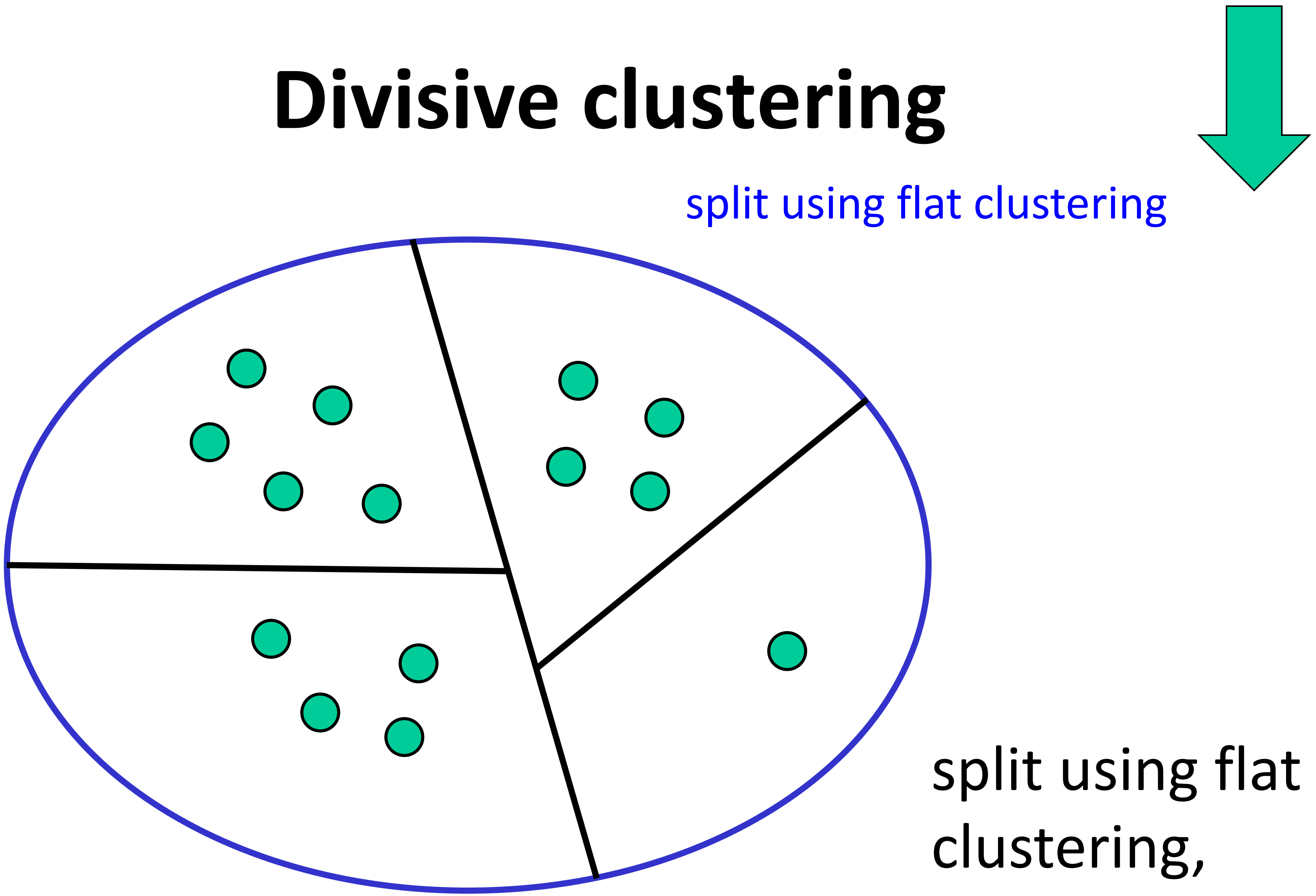


Divisive clustering



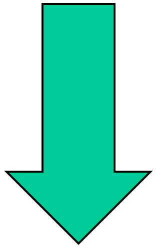
split using flat clustering, e.g., K-means

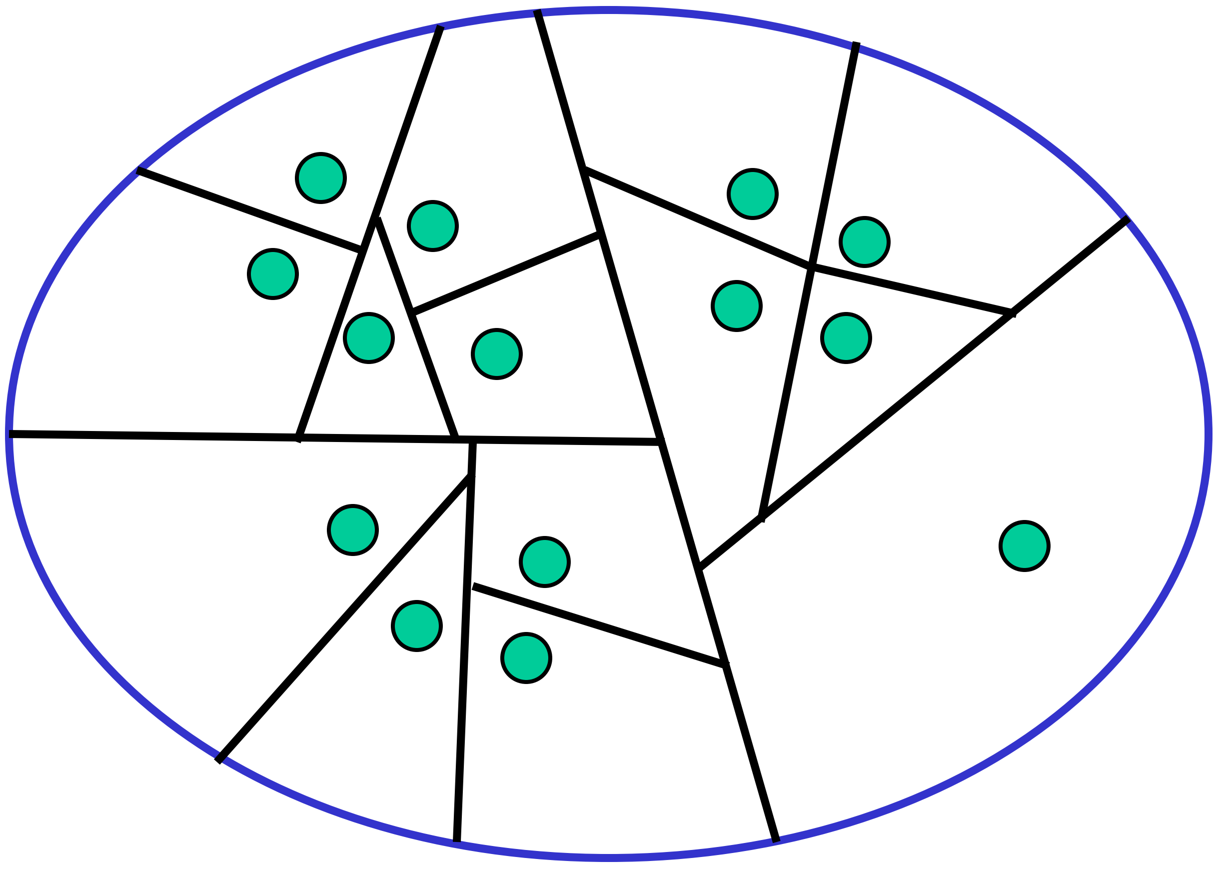
Divisive clustering



split using flat clustering, e.g., K-means

Divisive clustering





Stop when clusters reach some constraint, e.g., all of size 1

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.

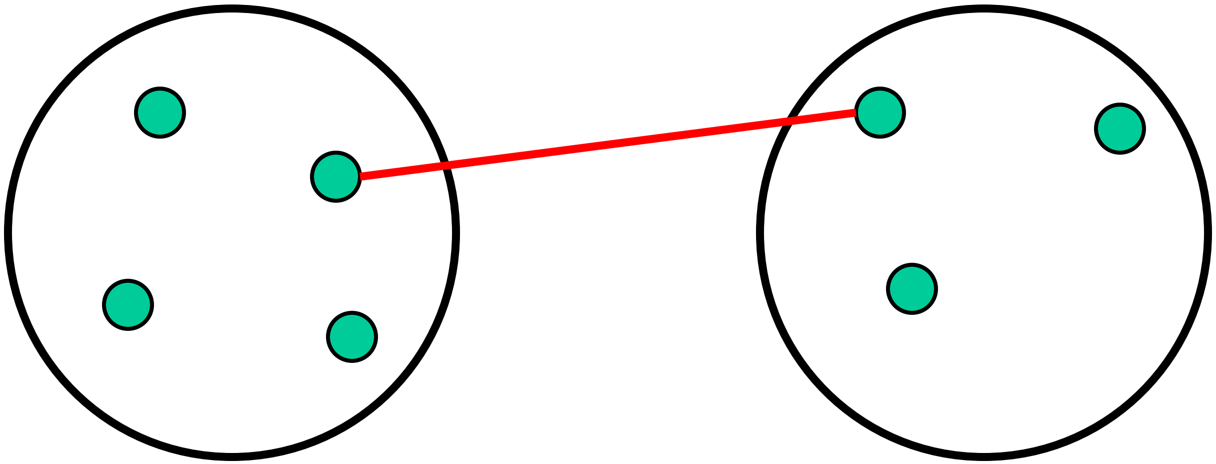
ChrisAlbon

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)



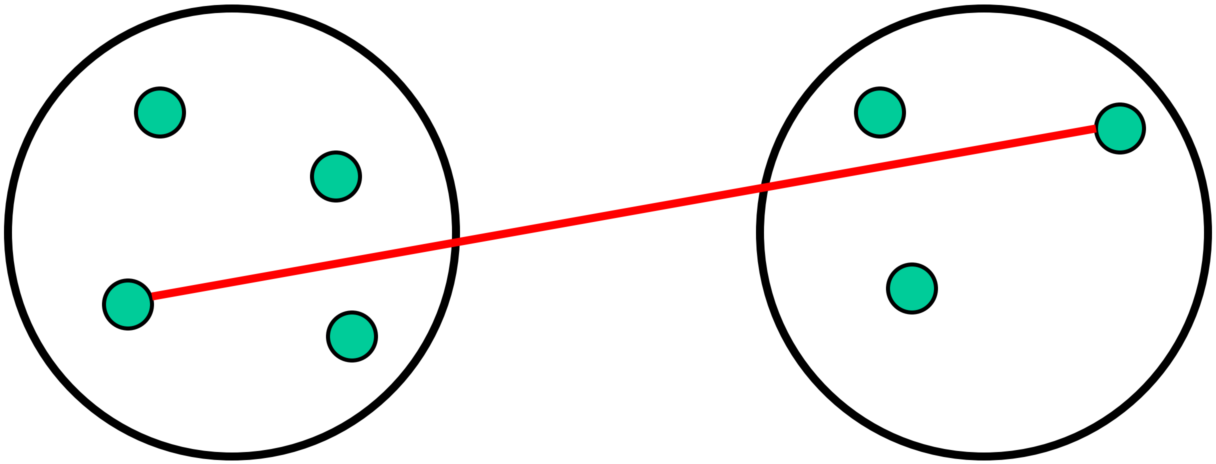
$$\max_{l \in L, r \in R} sim(l,r)$$

Weka: linkType=SINGLE

Distance between clusters

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

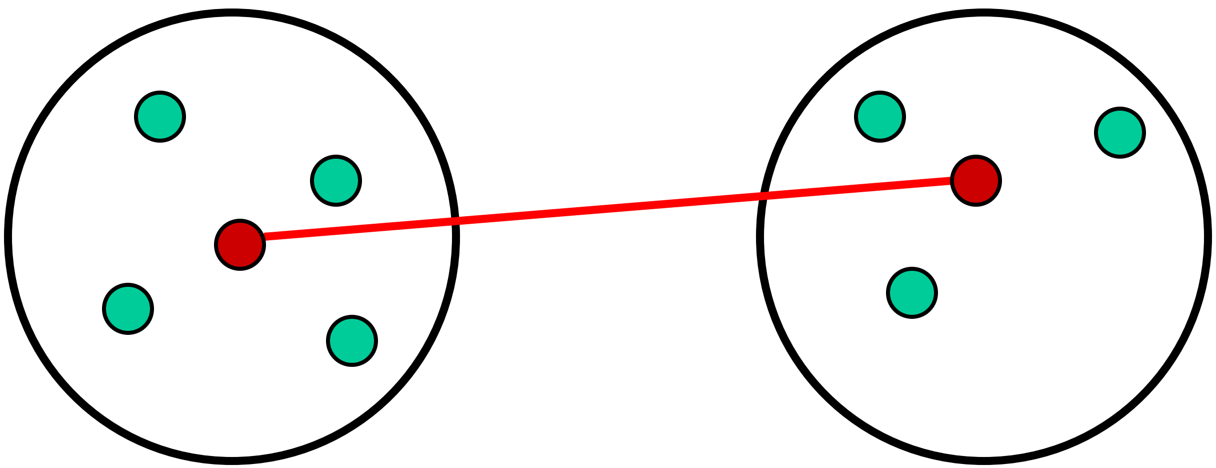
$$\min_{l \in L, r \in R} sim(l,r)$$



Weka: linkType=COMPLETE

Distance between clusters

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

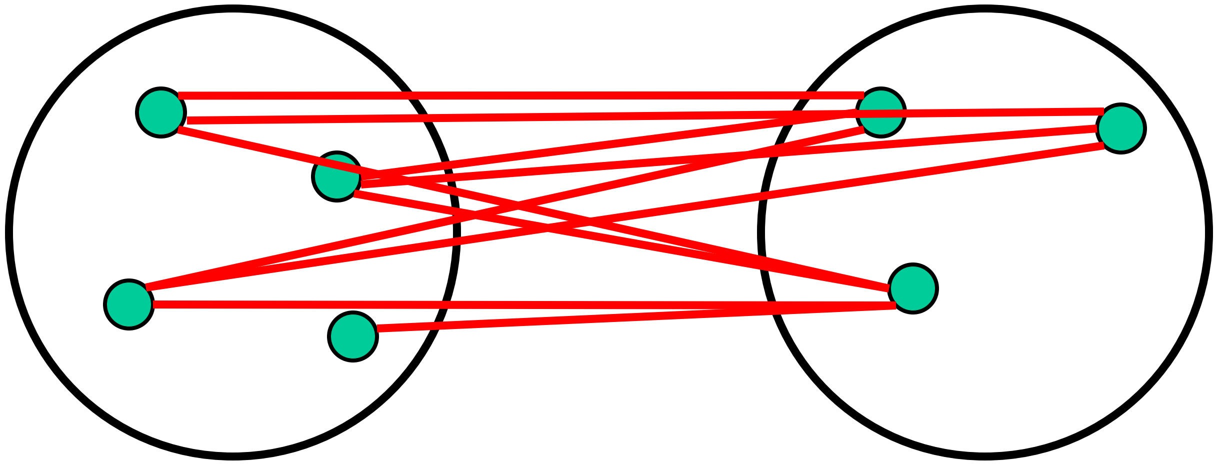


$$\left\|\mu(L)-\mu(R)\right\|^2$$

Weka: linkType=CENTROID

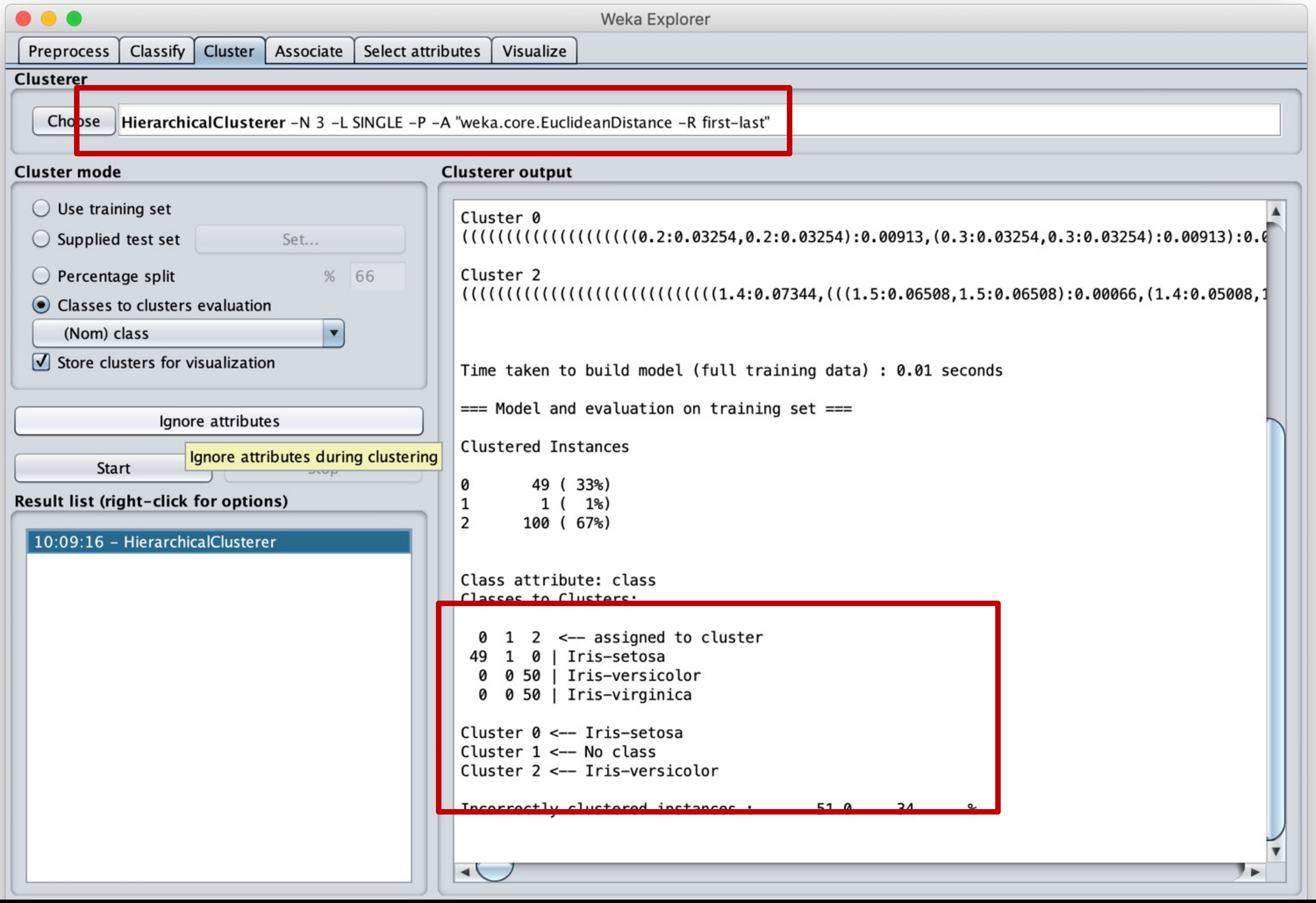
Distance between clusters

Average-link: Average similarity between all pairs of elements

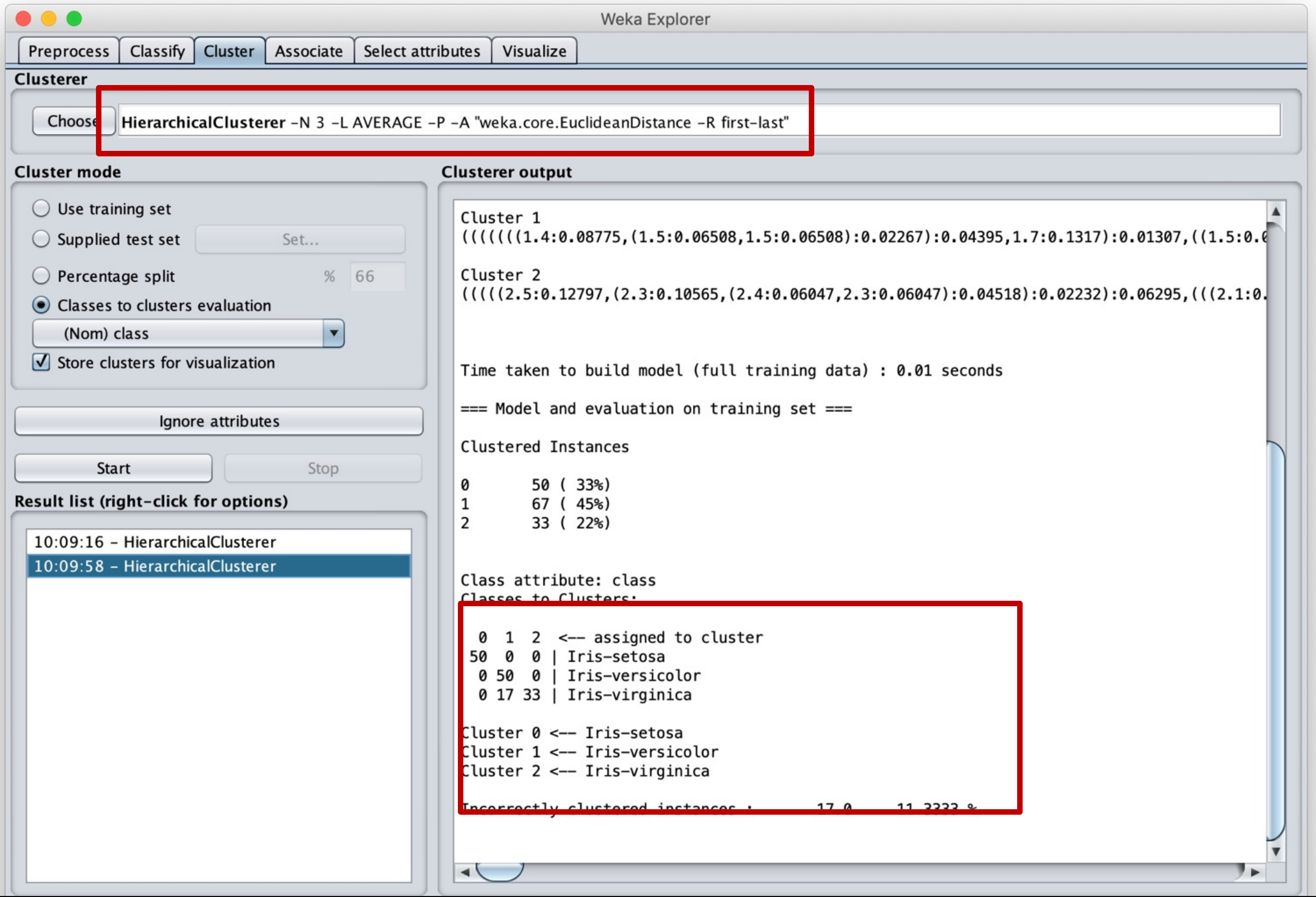


$$\frac{1}{|L|\cdot|R|}\sum_{x\in L,\, y\in R}\|x-y\|^2$$

Weka: linkType=AVERAGE



Default **SINGLE** cluster distance gives poor results on IRIS



AVERAGE cluster distance measure improves results for IRIS

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 general metrics for cluster quality
  - E.g., Silhouette coefficient and Dunn Index
  - Use one of these to decide where to stop
- There are also domain specific heuristics for cluster quality

(3) DBSCAN Algorithm

- Density-Based Spatial Clustering of Applications with Noise
- It 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 points in low-density regions as outliers
- Needn't specify number of clusters expected
- Fast

DBSCAN

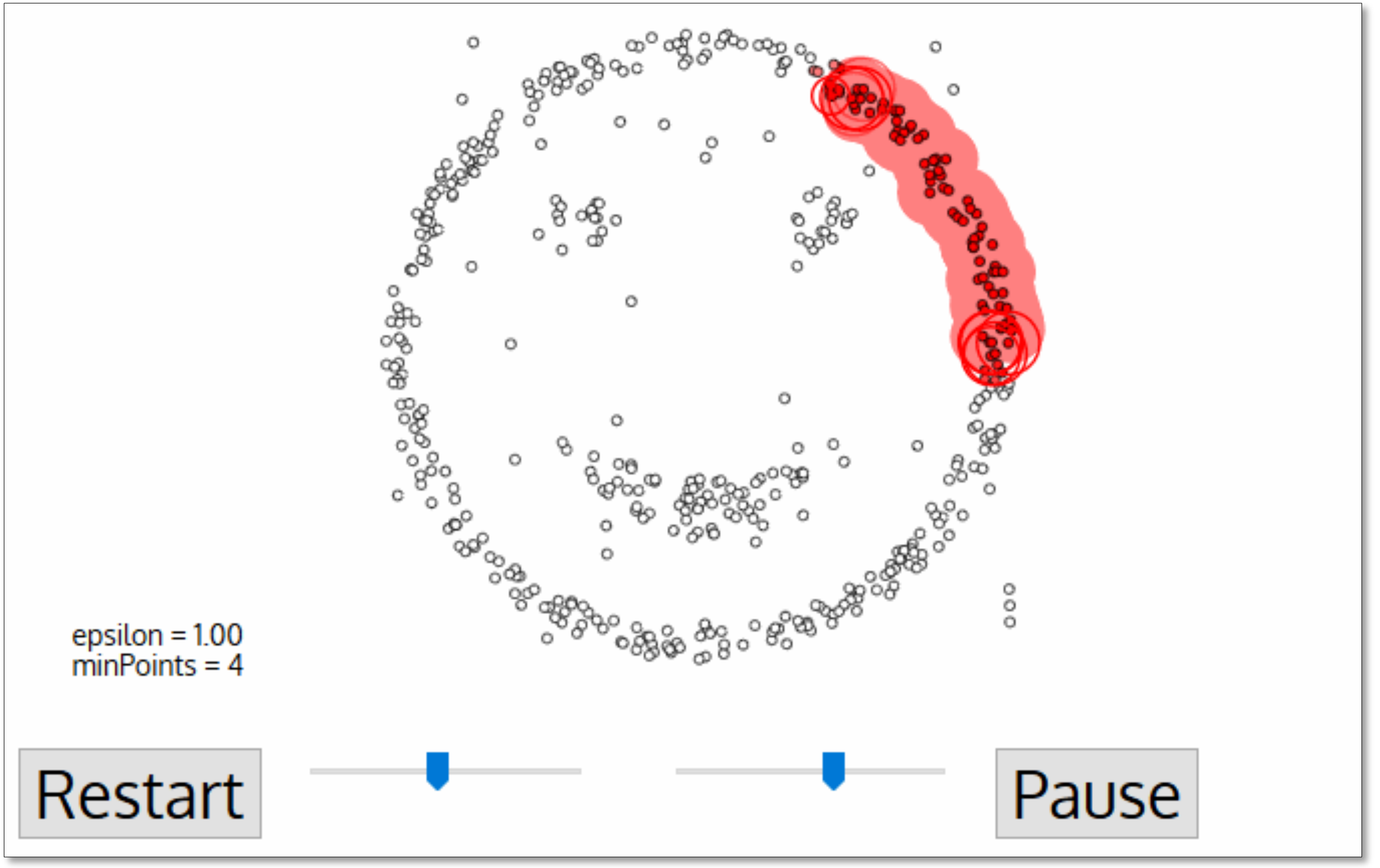
DBSCAN looks for densely packed observations and makes no assumptions about the number or shape of clusters.

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

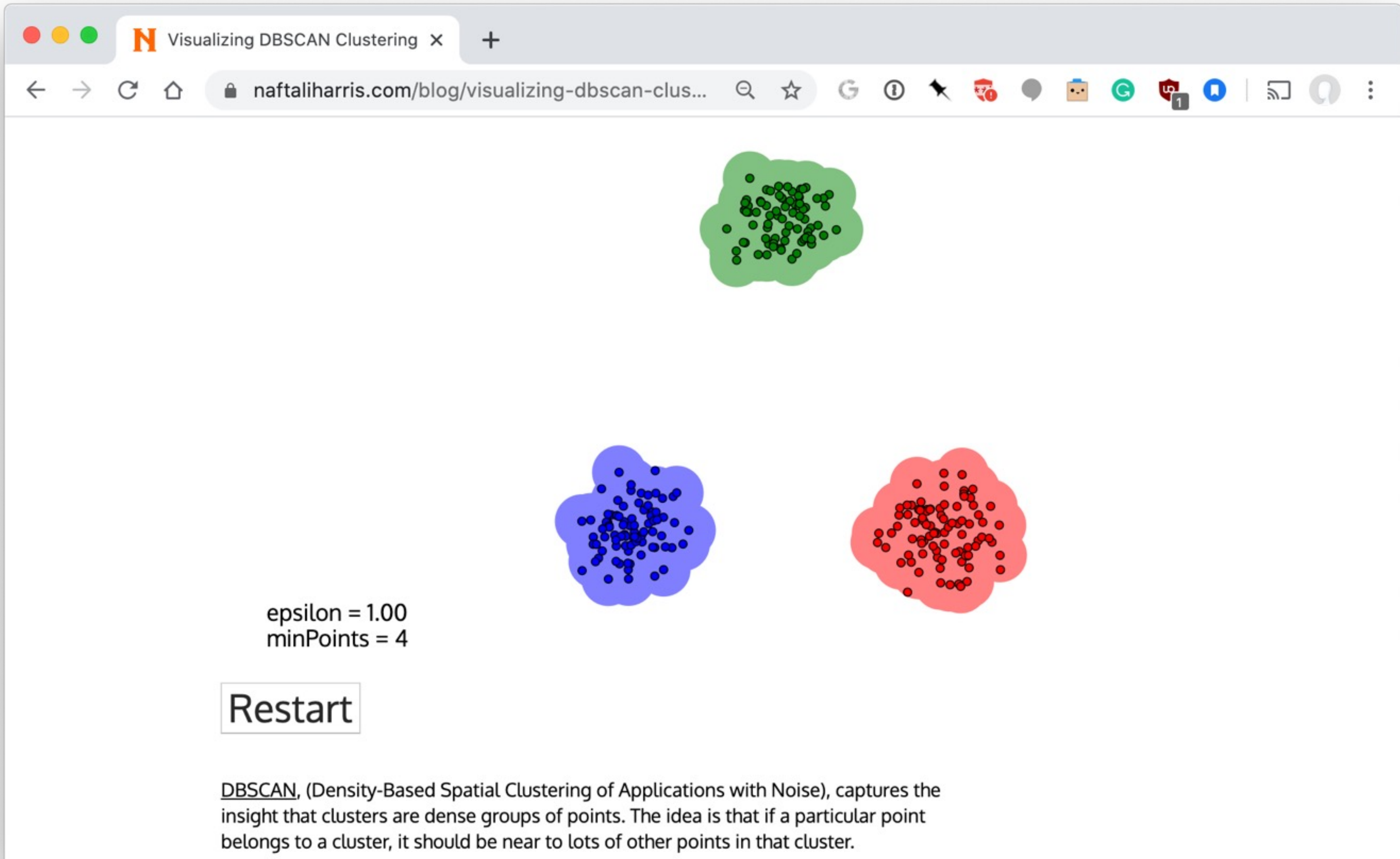
DBSCAN Example



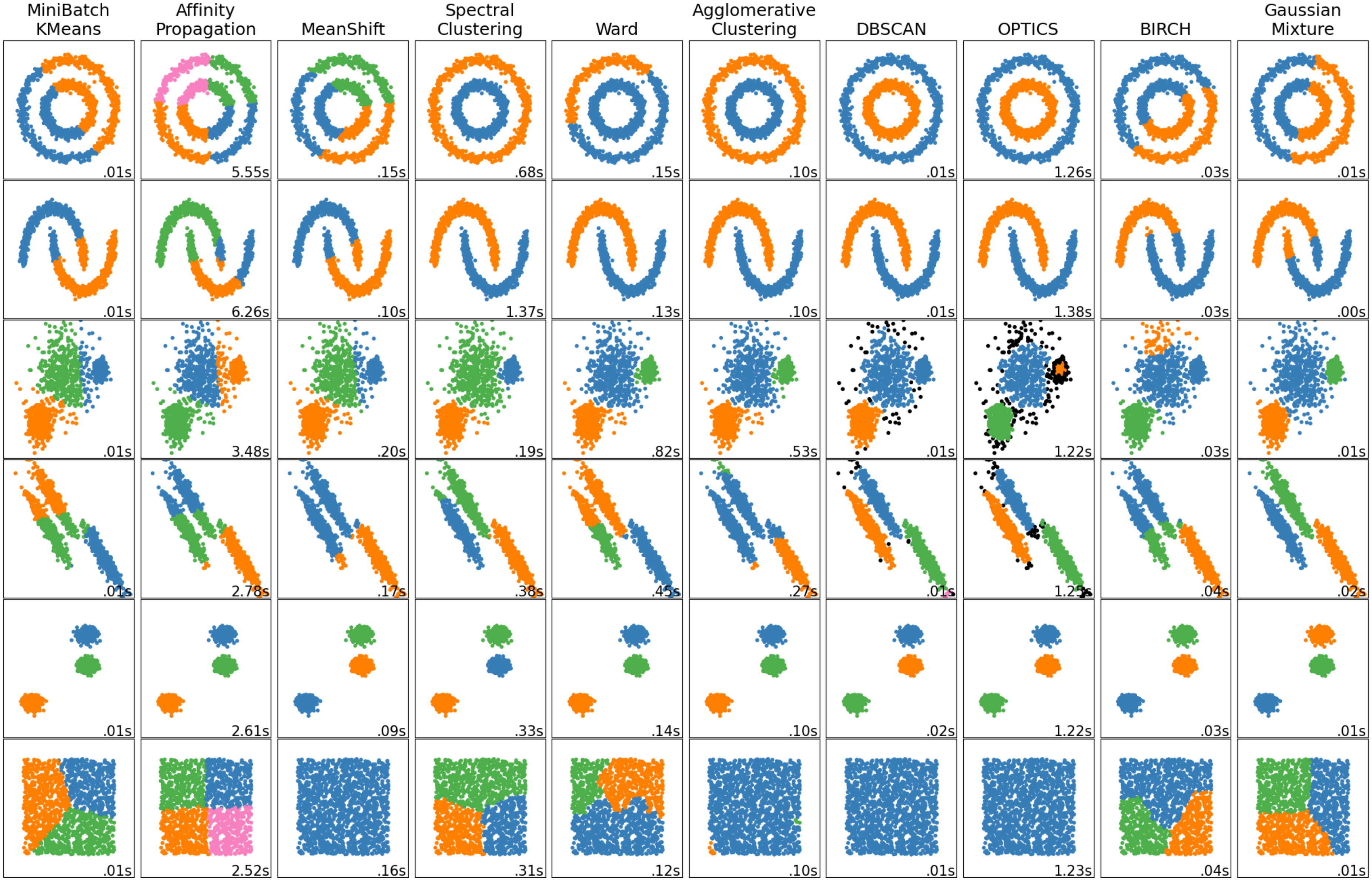
This gif (in ppt) shows how DBSCAN grows four clusters and identifies the remaining points as outliers

Visualizing DBSCAN

https://bit.ly/471dbscan



10 clustering algorithms on 6 datasets with scikit-learn



Clustering Summary

- Clustering useful & effective for many tasks
- K-means clustering one of simplest & fastest techniques, but
  - Requires knowing how many clusters is right
  - Doesn’t handle outliers well
- Hierarchical clustering slower & more general, but needs metric to know when to stop
- There are many other clustering options
  - DBSCAN is just one of them
  - Experiment to see what’s best for your application