Applied Machine Learning

Unsupervised Learning

Isabeau Prémont-Schwarz

McGill
School of Computer Science

Supervised v.s. Unsupervised Learning

1. Supervised learning: learning from examples (**labeled** data)
   - regression, classification $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$
   - e.g. predict if patient has cancer based on different tests/measurements, learn from previous diagnosis cases (previous cancer/no cancer cases)
2. Unsupervised learning: inferring from patterns (**unlabeled** data)
   - clustering $\mathcal{D} = \{x^{(n)}\}_{n=1}^{N}$
   - e.g. find different types of patients based on different tests/measurements

In many practical cases we do not have labels, and unsupervised techniques could be useful as a first option to understand our data or might be sufficient on their own based on the task

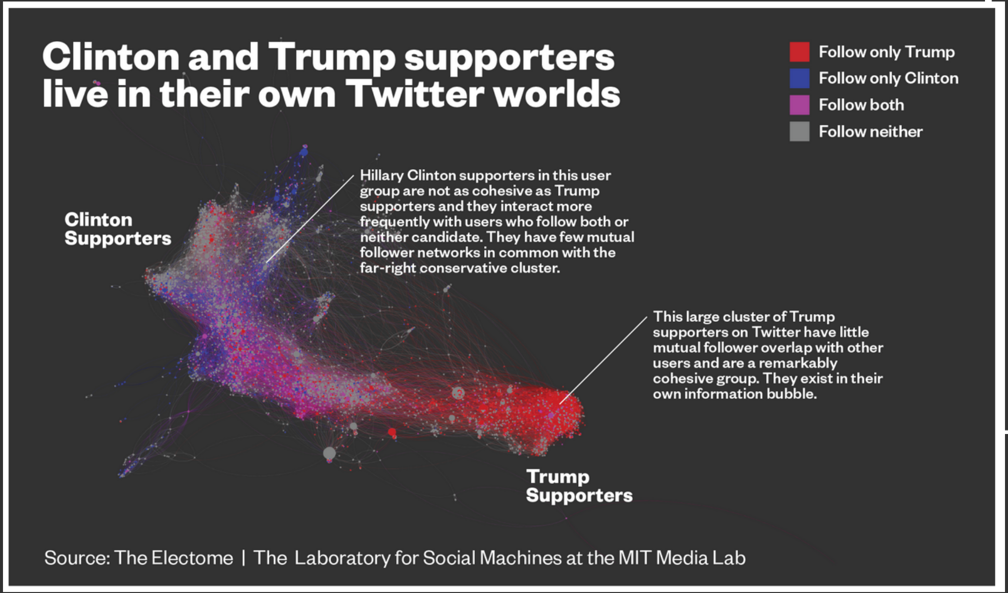
Learning objectives

- what is clustering and when is it useful?
- what are the different types of clustering?
- some clustering algorithms:
    - k-means, k-medoid, DB-SCAN, hierarchical clustering

Clustering Examples

for many applications we want to classify the data without having any labels

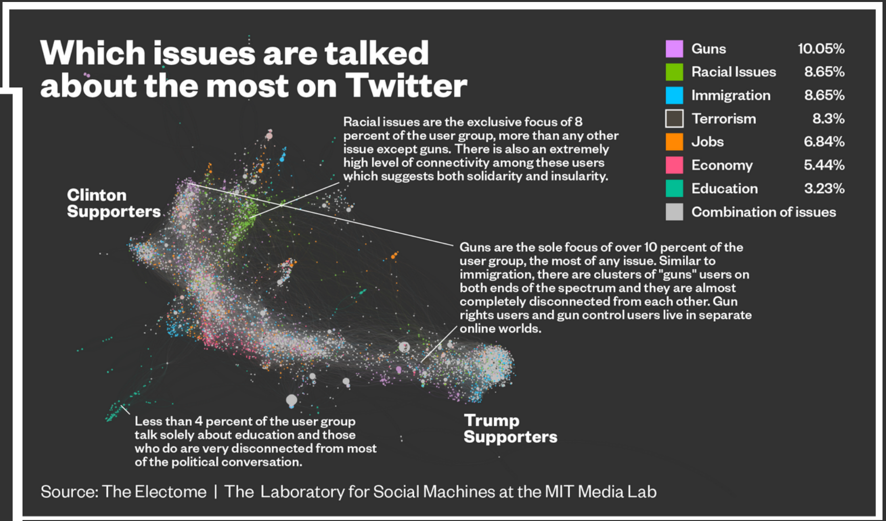
- categories of shoppers or items based on their shopping patterns
- communities in a social network



Hillary Clinton supporters in this user group are not as cohesive as Trump supporters and they interact more frequently with users who follow both or neither candidate. They have few mutual follower networks in common with the far-right conservative cluster.

This large cluster of Trump supporters on Twitter have little mutual follower overlap with other users and are a remarkably cohesive group. They exist in their own information bubble.

Source: The Electome | The Laboratory for Social Machines at the MIT Media Lab



🟪	Guns	10.05%
🟩	Racial Issues	8.65%
🟦	Immigration	8.65%
⬛	Terrorism	8.3%
🟧	Jobs	6.84%
🟥	Economy	5.44%
🟩	Education	3.23%
⬜	Combination of issues	

Racial issues are the exclusive focus of 8 percent of the user group, more than any other issue except guns. There is also an extremely high level of connectivity among these users which suggests both solidarity and insularity.

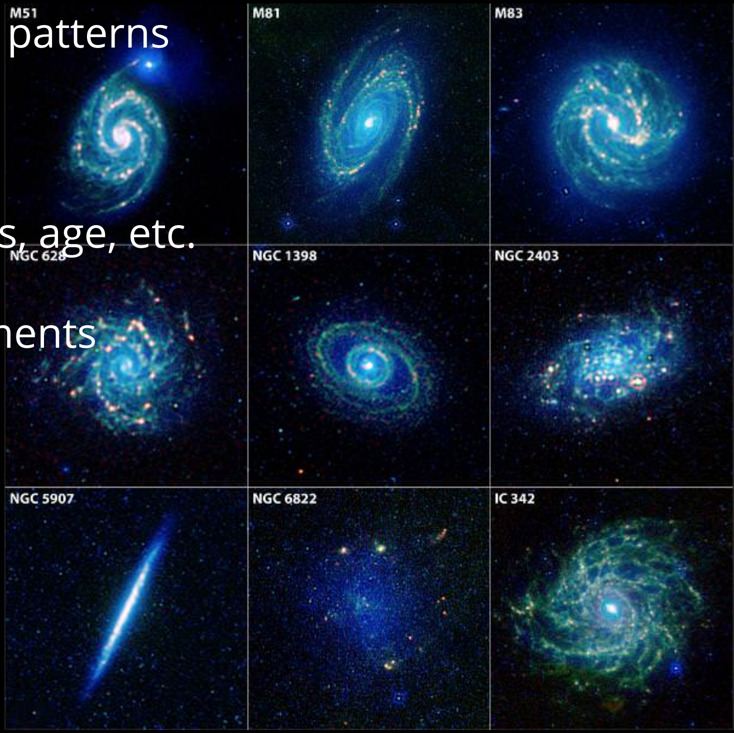
Guns are the sole focus of over 10 percent of the user group, the most of any issue. Similar to immigration, there are clusters of "guns" users on both ends of the spectrum and they are almost completely disconnected from each other. Gun rights users and gun control users live in separate online worlds.

Less than 4 percent of the user group talk solely about education and those who do are very disconnected from most of the political conversation.

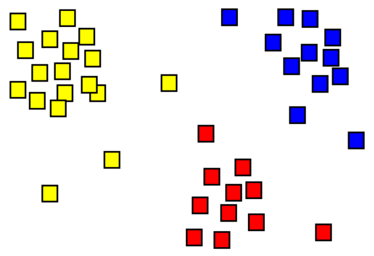
Source: The Electome | The Laboratory for Social Machines at the MIT Media Lab

Clustering Examples

- categories of shoppers or items based on their shopping patterns
- communities in a social network
- categories of stars or galaxies based on light profile, mass, age, etc.
- categories of minerals based on spectroscopic measurements
- categories of webpages in meta-search engines
- categories of living organisms based on their genome
- ...



What is a cluster?



a **subset** of entities that are similar to each other and different from other entities

we can try and organize clustering methods based on

- form of input data
- types of cluster / task
- general methodology

Common Types of input

1. features $X \in \mathbb{R}^{N \times D}$

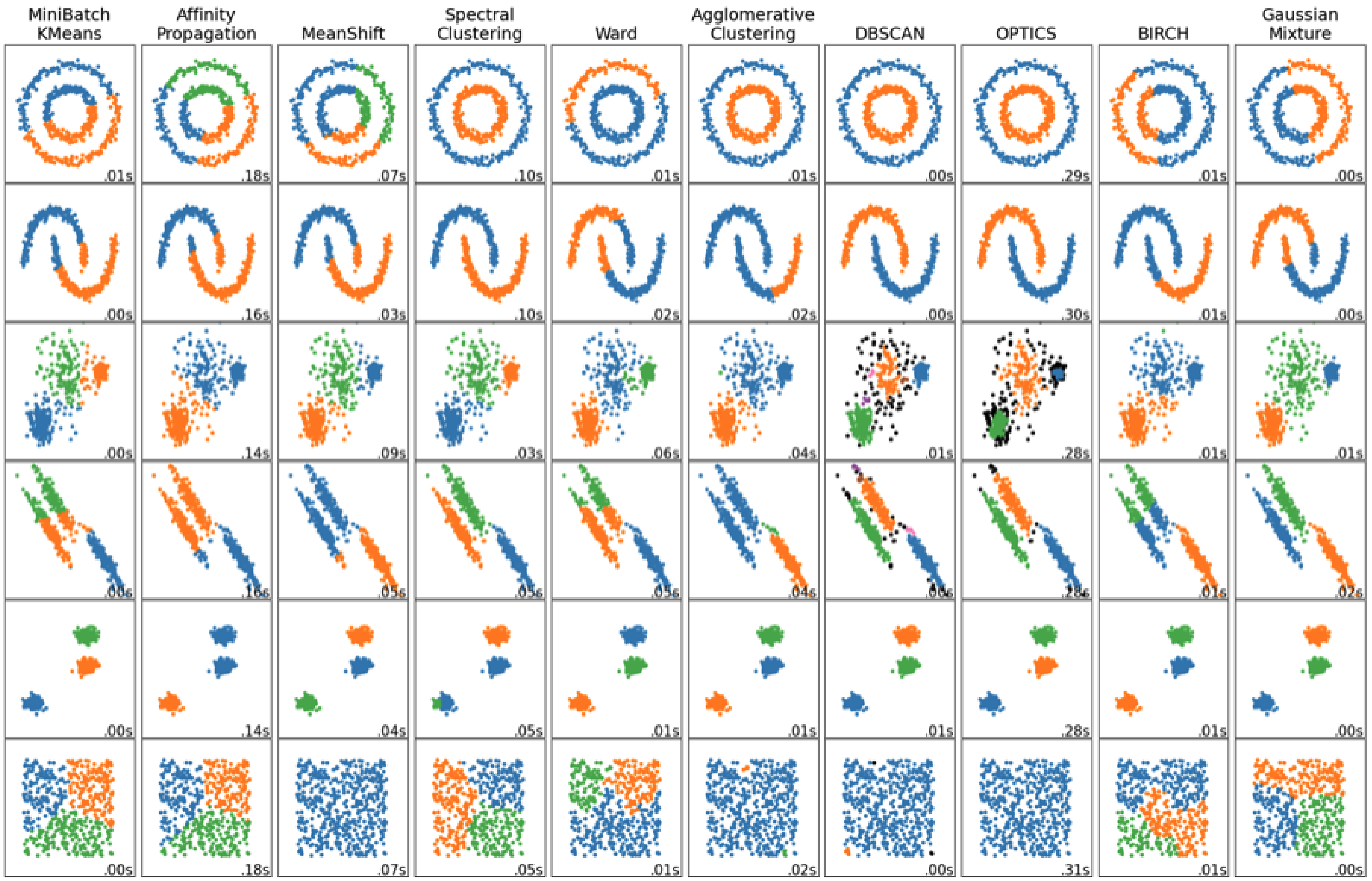
2. pairwise distances or similarities $D \in \mathbb{R}^{N \times N}$

- we can often produce similarities from features
- infeasible for very large D

3. graphs with attributes on nodes $X \in \mathbb{R}^{N \times D}, A \in \mathbb{R}^{N \times N}$

- node attribute is similar to feature in the first family
- edge attribute can represent similarity or distance

Common Clustering Algorithms

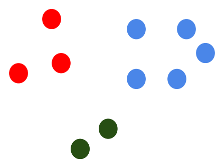


A comparison of the clustering algorithms in scikit-learn

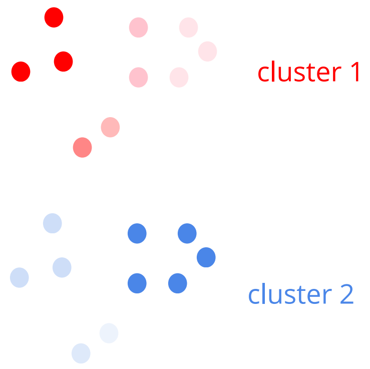
read more here

Types of cluster / task

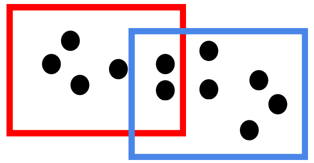
partitioning or hard clusters



soft fuzzy membership



overlapping

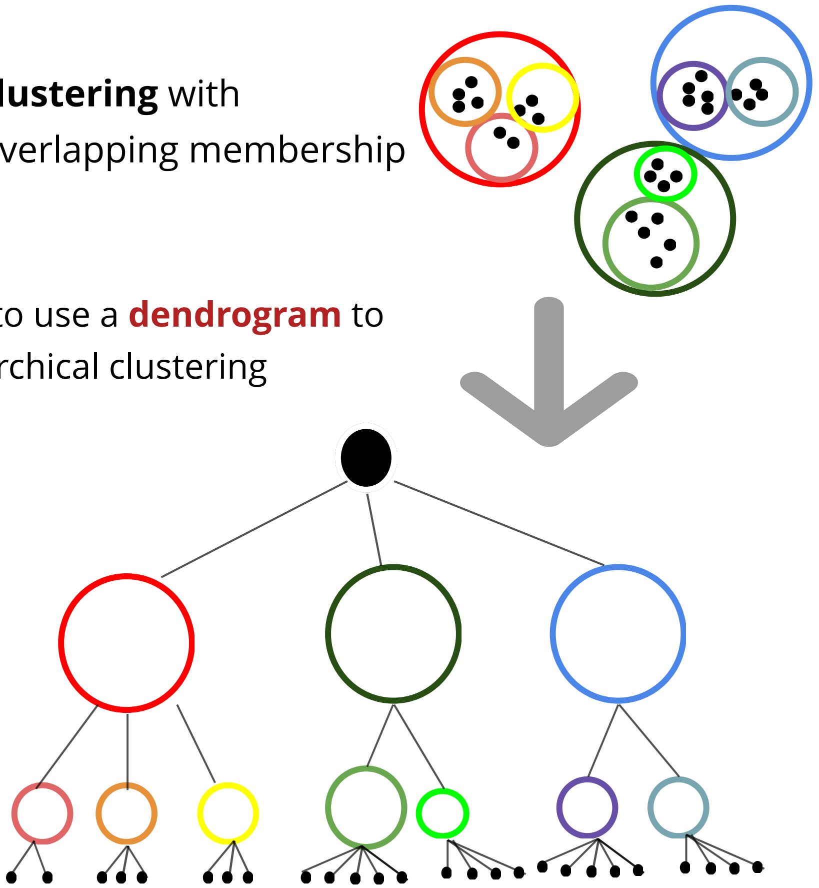


Types of cluster / task

other categories!

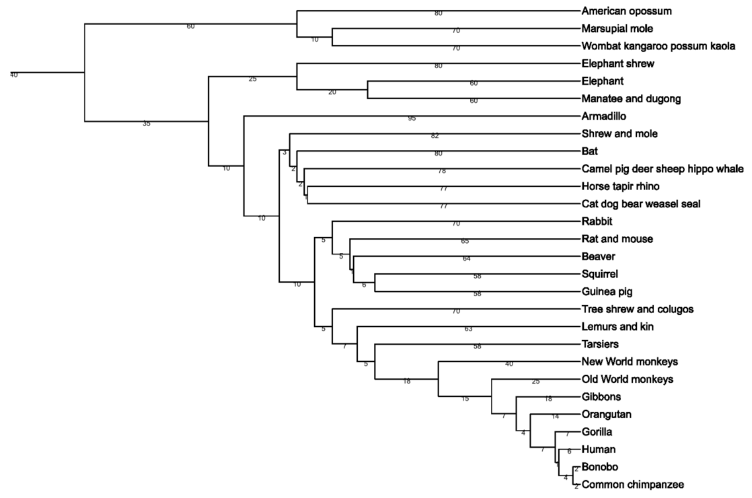
hierarchical clustering with
hard, soft, or overlapping membership

it is customary to use a **dendrogram** to
represent hierarchical clustering



example

tree of life (clustering of genotypes)



Centroid methods

identify the centers, prototypes or exemplars of clusters

each cluster is
represented by its center

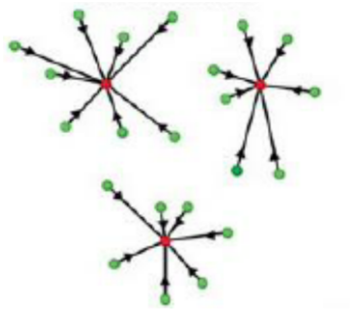


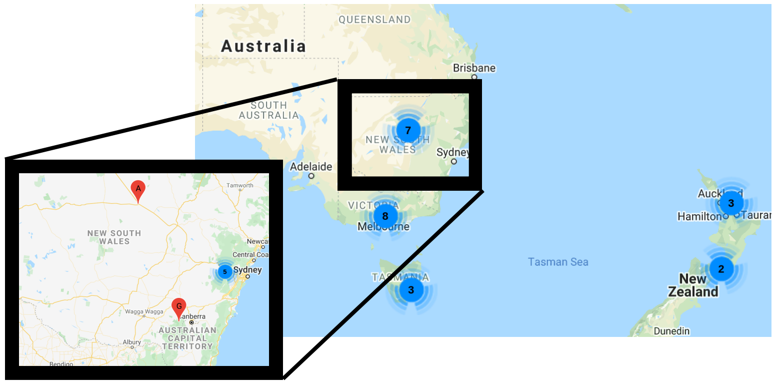
image: Frey & Dudek'00

early use of clustering in psychology

example

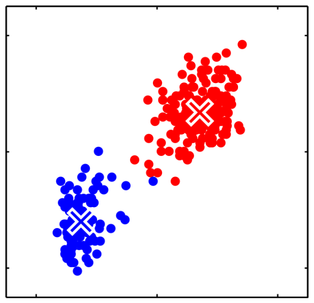
cluster centers are shown on the map

a hierarchical clustering with level of hierarchy depending on the zoom level



K-means is an example of a centroid method

K-means clustering: objective



idea partition the data into K-clusters to minimize the sum of distance to the cluster mean/center

equivalent to minimizing the within cluster distances

K is a hyperparameter

cluster center (mean) $\mu_k = \frac{\sum_n x^{(n)} r_{n,k}}{\sum_n r_{n,k}}$

cost function

number of points

number of clusters

$$J(\{r_{n,k}\}, \{\mu_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x^{(n)} - \mu_k||_2^2$$

cluster membership $r_{n,k} = \begin{cases} 1 & \text{point n belongs to cluster k} \\ 0 & \text{otherwise} \end{cases}$

we need to find **cluster memberships** and **cluster centers**

how to minimize the cost?

K-means clustering: algorithm

idea: iteratively update cluster memberships and cluster centers

```
start with some cluster centers  $\{\mu_k\}$

repeat until convergence:

    assign each point to the closest center:  $r_{n,k} \leftarrow \begin{cases} 1 & k = \arg\min_c ||x^{(n)} - \mu_c||_2^2 \\ 0 & \text{otherwise} \end{cases}$

    re-calculate the center of each cluster:  $\mu_k \leftarrow \frac{\sum_n x^{(n)} r_{n,k}}{\sum_n r_{n,k}}$
```

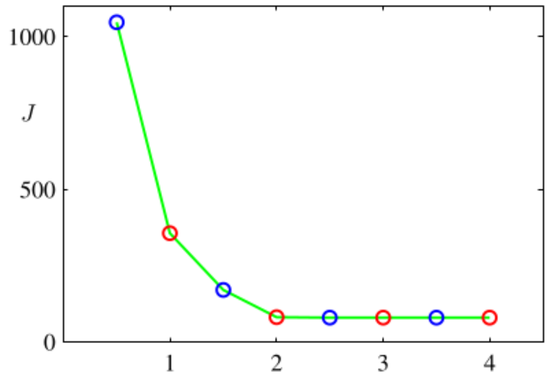
since each iteration can only reduce the cost, the algorithm has to stop

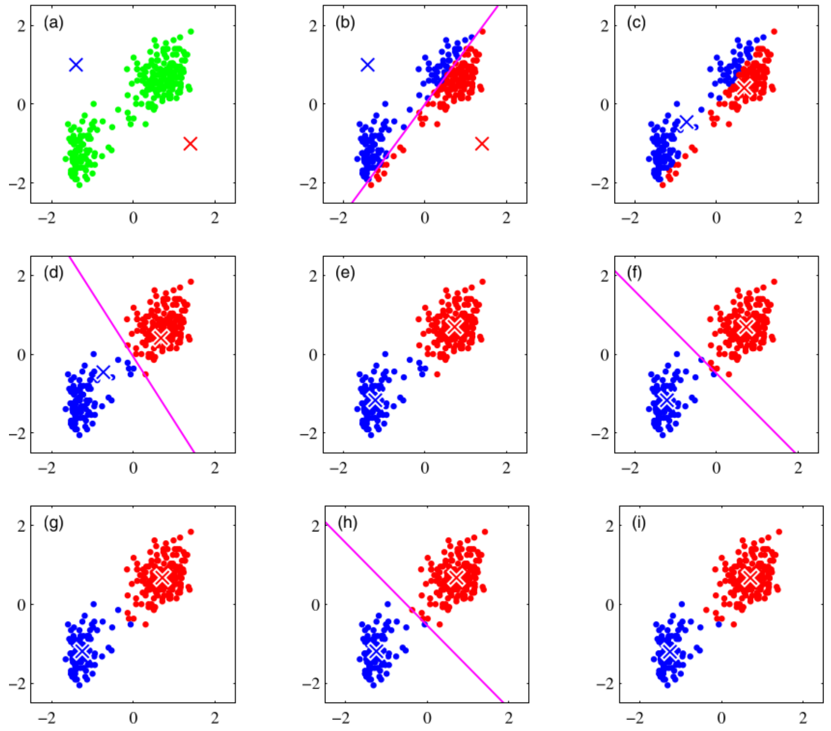
K-means clustering: algorithm

example iterations of k-means (K=2) for 2D data.

Two steps in each iteration are shown.

the cost decreases at each step





K-means clustering: derivation

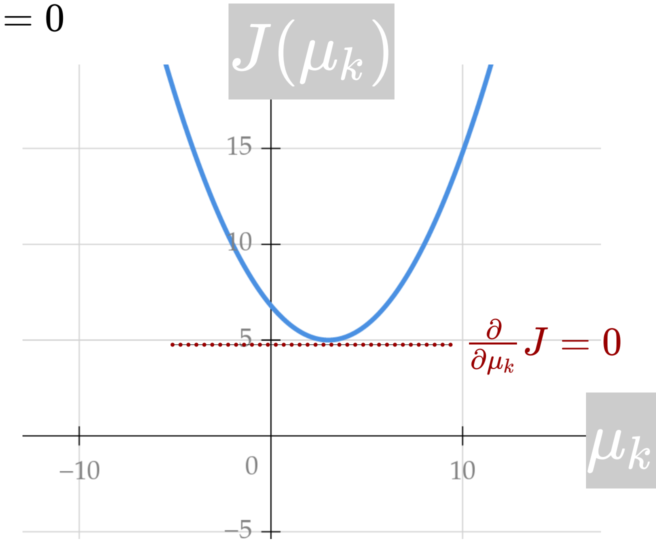
why this procedure minimizes the cost? $J(\{r_{n,k}\}, \{\mu_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x^{(n)} - \mu_k||_2^2$

1. fix memberships $\{r_{n,k}\}$ and optimize centers $\{\mu_k\}$

   set the derivative wrt μ_k to zero: $\frac{\partial}{\partial \mu_k} J = \frac{\partial}{\partial \mu_k} \sum_n r_{n,k} ||x^{(n)} - \mu_k||_2^2 = 0$

$$2 \sum_n r_{n,k}(x^{(n)} - \mu_k) = 0$$

$$\mu_k = \frac{\sum_n x^{(n)} r_{n,k}}{\sum_n r_{n,k}}$$ ✓



2. fix centers $\{\mu_k\}$ and optimize memberships $\{r_{n,k}\}$

   finding the "closest" center minimizes the cost

$$r_{n,k} \leftarrow \begin{cases} 1 & k = \arg\min_c ||x^{(n)} - \mu_c||_2^2 \\ 0 & \text{otherwise} \end{cases}$$ ✓

3. repeat 1 & 2 until convergence

K-means clustering: complexity

```
start with some cluster centers $\{\mu_k\}$

repeat until convergence:

    assign each point to the closest center: $r_{n,k} \leftarrow \begin{cases} 1 & k = \arg\min_c ||x^{(n)} - \mu_c||_2^2 \\ 0 & \text{otherwise} \end{cases}$

    re-calculate the center of each cluster: $\mu_k \leftarrow \frac{\sum_n x^{(n)} r_{n,k}}{\sum_n r_{n,k}}$
```

calculating the mean of all cluster $\mathcal{O}(ND)$

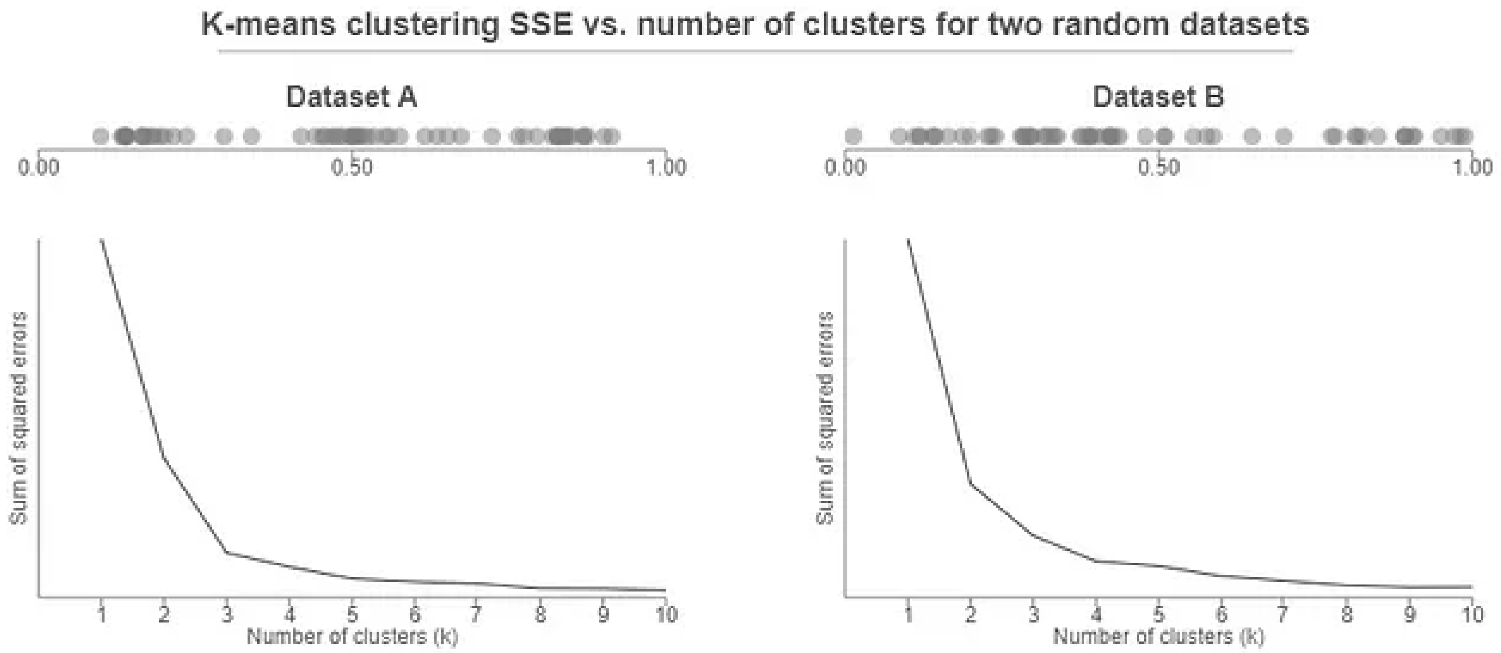
calculating distance of a node to center $\mathcal{O}(D)$, number of features

we do this for each point (n) and each center (k)

total cost is $\mathcal{O}(NKD)$

K-means clustering: **choice of K?**

Elbow/Hockey-stick🏒 Method

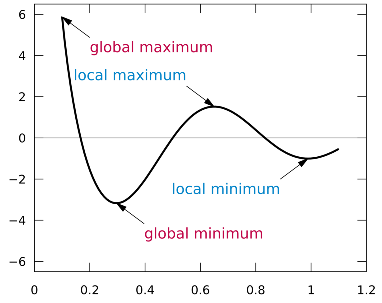


Source: bl.ocks.org/

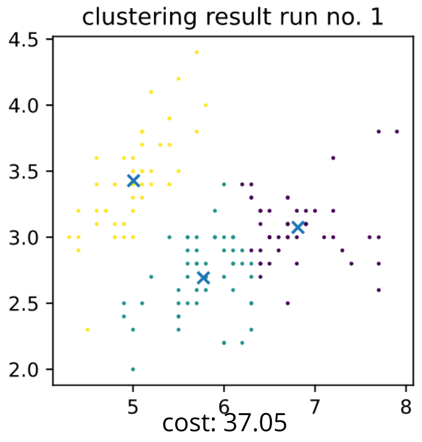
K-means clustering: performance

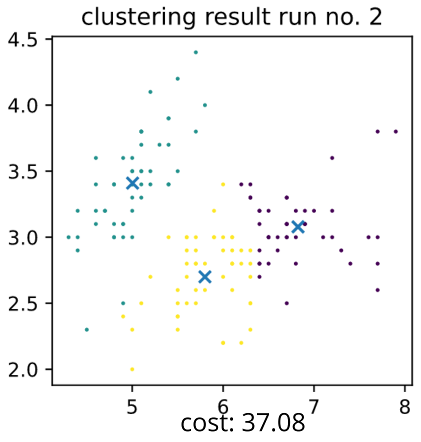
K-means' alternating minimization finds a **local** minimum

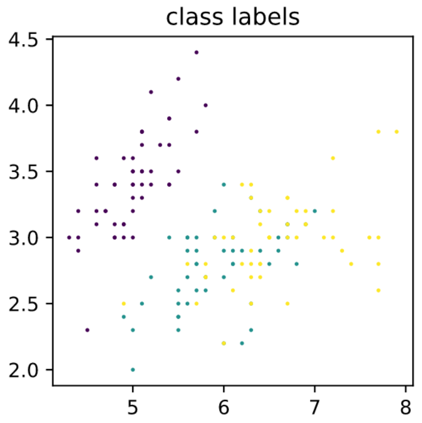
different **initialization** of cluster centers gives different clustering



example: Iris flowers dataset (also interesting to compare to true class labels)







even if the clustering is the same cluster indices (colors) could swap optional

K-means clustering: initialization

K-means' alternative minimization finds a **local** minimum

different **initialization** gives different clustering:

- run many times and pick the clustering with the lowest cost
- use good heuristics for initialization:

K-means++ initialization

- pick a random data-point to be the first center
- calculate the distance of each point to the nearest center d_n
- pick a new point as a new center with prob $p(n) = \frac{d_n^2}{\sum_i d_i^2}$

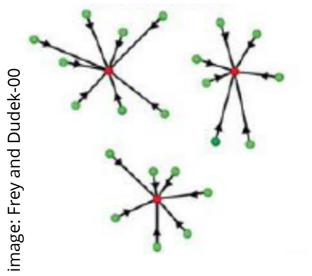
often faster convergence to better solutions

the clustering is [in expectation] within $\mathcal{O}(\log(K))$ x optimal solution

Application: vector quantization

given a dataset $\mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\}$ of vectors $x^{(n)} \in \mathbb{R}^D$

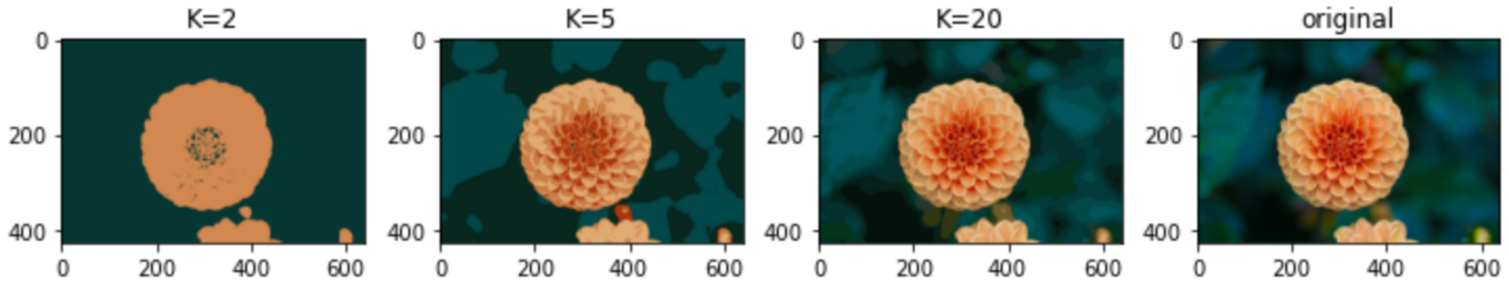
storage $\mathcal{O}(NDC)$ *C is the number of bits for a scalar (e.g., 32bits)*



compress the data using k-means:

- replacing each data-point with its cluster center
- store only the cluster centers and memberships $\mathcal{O}(KDC + N\log(K))$

apply this to compress images (denote each pixel by $x^{(n)} \in \mathbb{R}^3$)



cluster the colors

K-medoids

K-means objective minimizes squared Euclidean distance

the minimizer for a set of points is given by their mean

$D(x, x') = \sum_d |x_i - x'_i|$

if we use Manhattan distance the minimizer is the median (K-medians)

for general distance functions the minimizer doesn't have a *closed form* *(computationally expensive)*

solution pick the cluster center from the points themselves (**medoids**)

K-medoid **objective**

$$J(\{r_{n,k}\}, \{\mu_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} dist(x^{(n)}, \mu_k) \quad \text{and} \quad \mu_k \in \{x^{(1)}, \ldots, x^{(N)}\}$$

algorithm

- assign each point to the "closest" center
- set the point with the min. overall distance to other points as the center of the cluster

K-medoids

solution pick the cluster center from the points themselves (**medeoids**)

K-medoid objective

$$J(\{r_{n,k}\}, \{\mu_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \mathit{dist}(x^{(n)}, \mu_k) \quad \text{and} \quad \mu_k \in \{x^{(1)}, \ldots, x^{(N)}\}$$

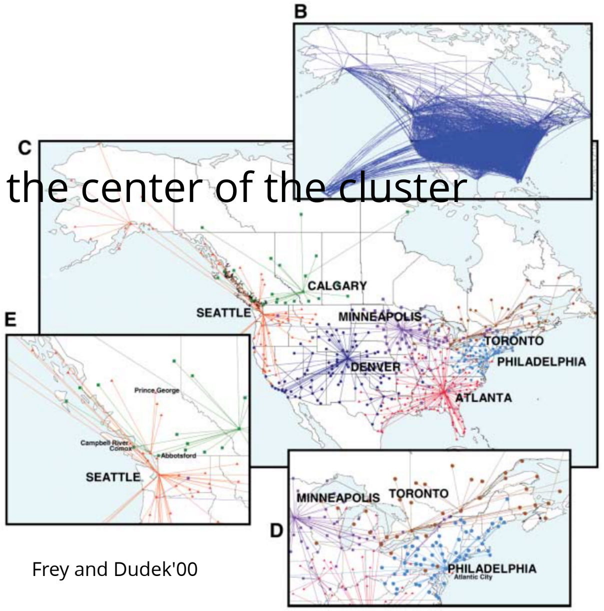
algorithm

- assign each point to the "closest" center
- set the point with the min. overall distance to other points as the center of the cluster

example finding key air-travel hubs (as medeoids)

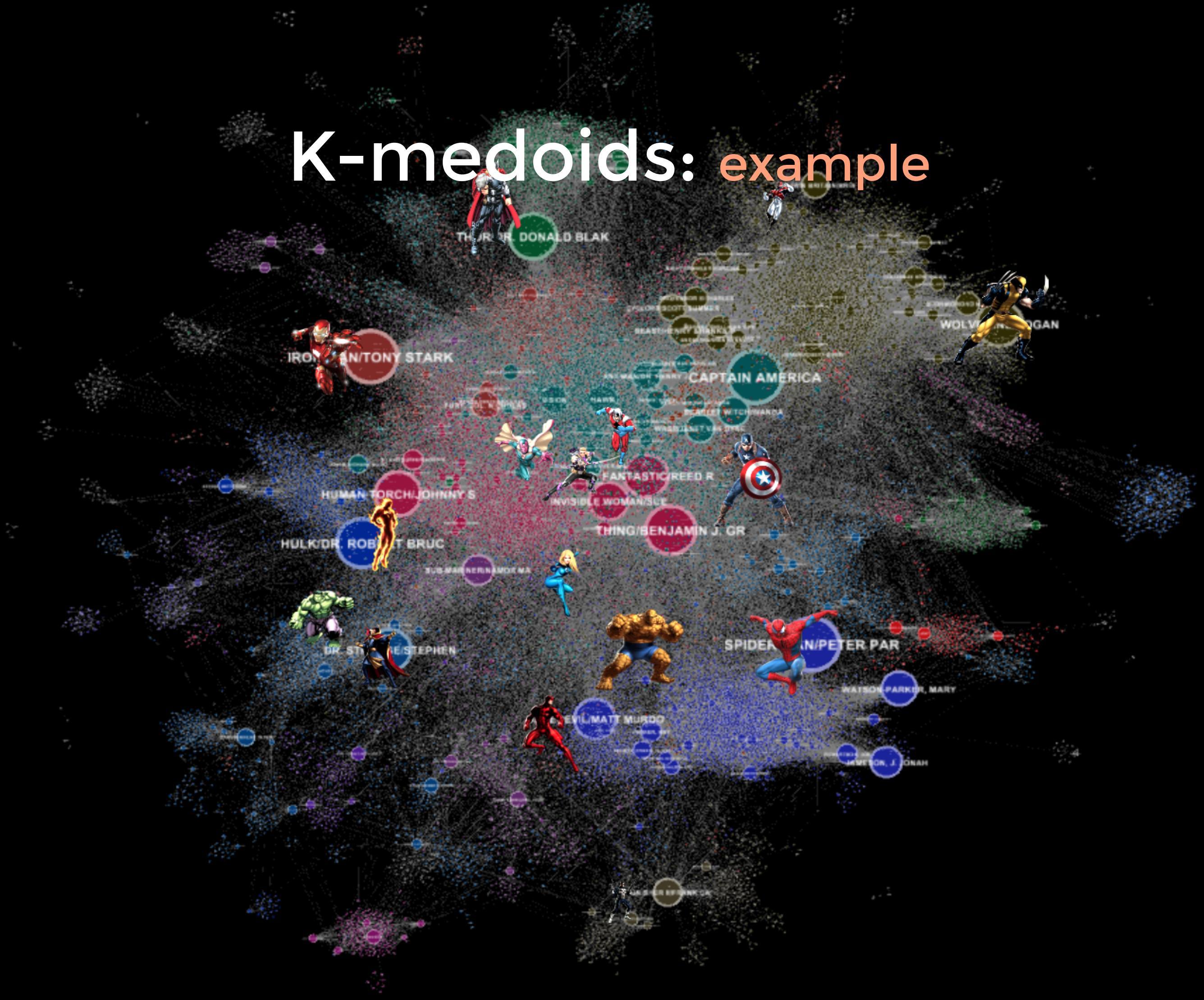
K-medoid also makes sense when the input is **graph**

(nodes become centers)



Frey and Dudek'00

K-medoids: example

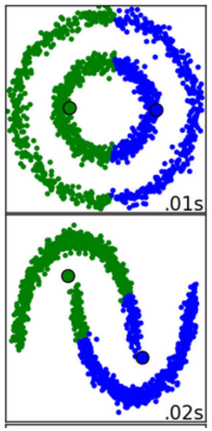


Density based methods

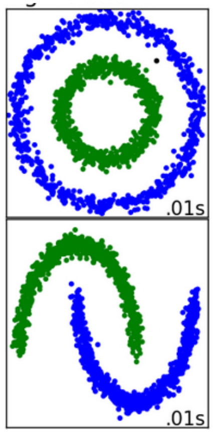
dense regions define clusters

a notable method is **d**ensity-**b**ased **s**patial **c**lustering of **a**pplications with **n**oise (**DB-SCAN**)

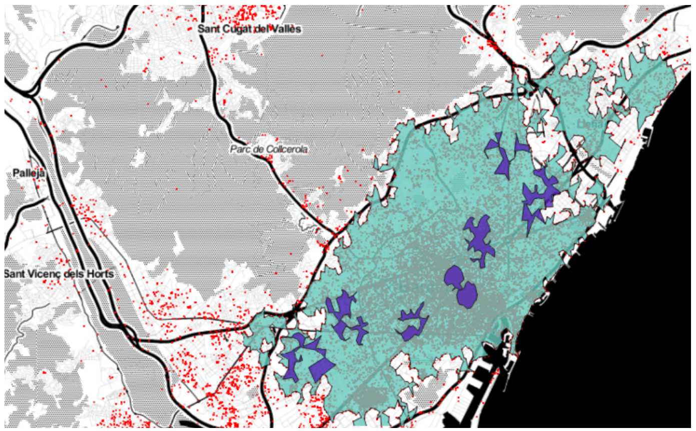
K-means



DB-SCAN



geospatial clustering



astronomical data

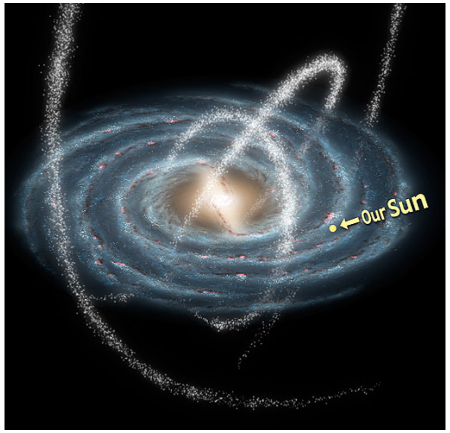


image credit: wiki, https://doublebyteblog.wordpress.com

DB-SCAN

points that have more than C neighbors in ϵ-neighborhood are called **core** points

if we connect nearby core points we get a graph

connected components of the graph give us **clusters**

all the other points are either:

- ϵ-close to a core, so belong to that cluster
- labeled as **noise**

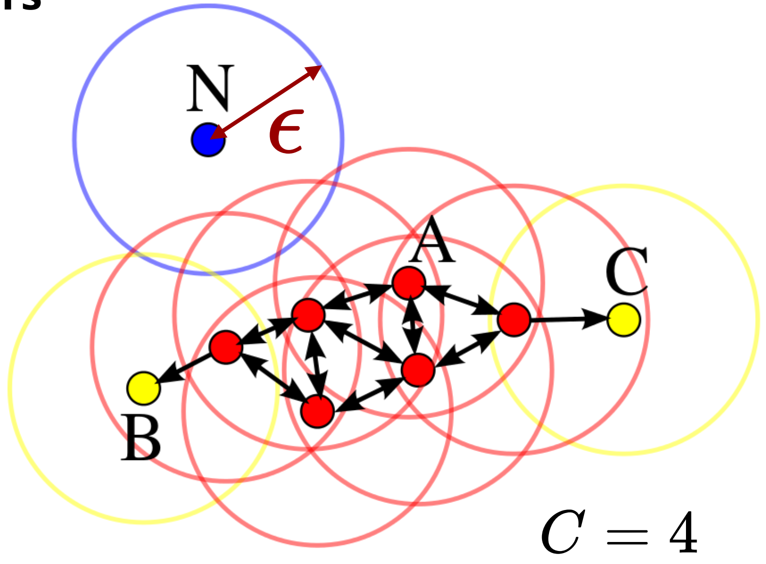


image credit: wiki

Hierarchical clustering heuristics

bottom-up hierarchical clustering (**agglomerative clustering)**

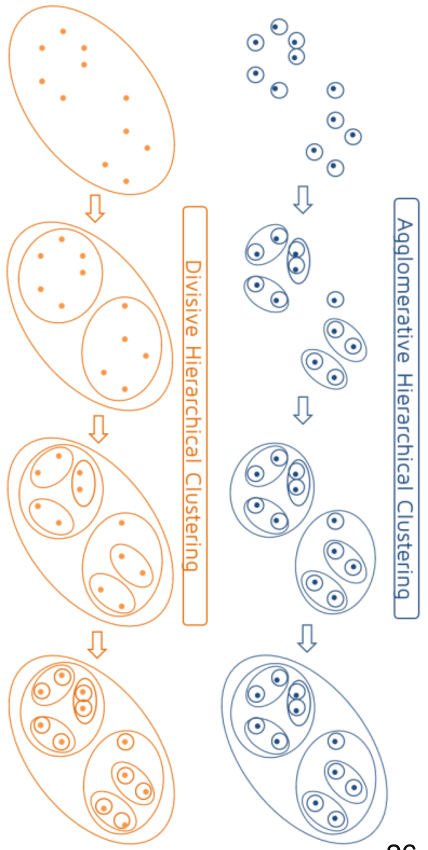
- start from each item as its own cluster
- merge most similar clusters

top-down hierarchical clustering (**divisive clustering**)

- start from having one big cluster
- at each iteration pick the "widest" cluster and split it

  (e.g. using k-means)

these methods often do not optimize a specific objective function (hence heuristics)
they are often too expensive for very large datasets

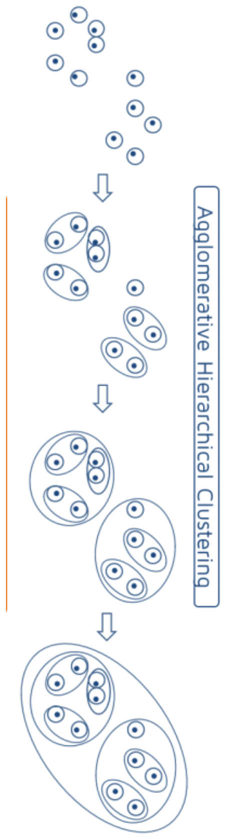

Divisive Hierarchical Clustering
Agglomerative Hierarchical Clustering

Agglomerative clustering

- start from each item as its own cluster
- merge most similar clusters

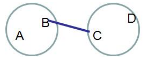
```
initialize clusters   $\mathcal{C}_n \leftarrow \{n\}, \quad n \in \{1, \ldots, N\}$
initialize set of clusters available for merging   $\mathcal{A} \leftarrow \{1, \ldots, N\}$
for  $t = 1, \ldots$
    pick two clusters that a most similar   $i, j \leftarrow \arg\min_{c, c' \in \mathcal{A}} \mathrm{distance}(c, c')$
    merge them to get a new cluster   $\mathcal{C}_{t+N} \leftarrow \mathcal{C}_i \cup \mathcal{C}_j$
    if  $\mathcal{C}_{t+N}$ contains all nodes, we are done!
    update clusters available for merging   $\mathcal{A} \leftarrow \mathcal{A} \cup \{t + N\} \backslash \{i, j\}$
    calculate dissimilarities for the new cluster   $\mathrm{distance}(t + N, n) \quad \forall n \in \mathcal{A}$
```

- how to define dissimilarity or distance of two clusters?



Agglomorative clustering

how to define dissimilarity of two clusters?

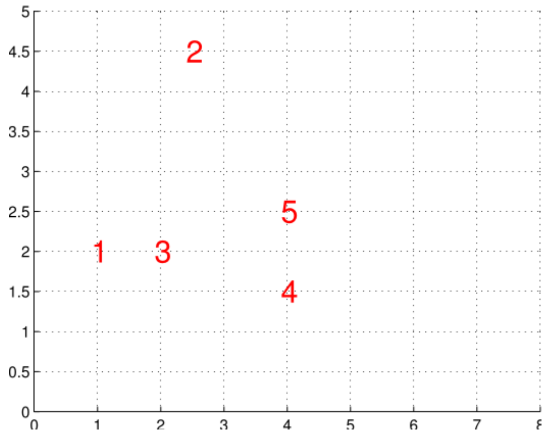


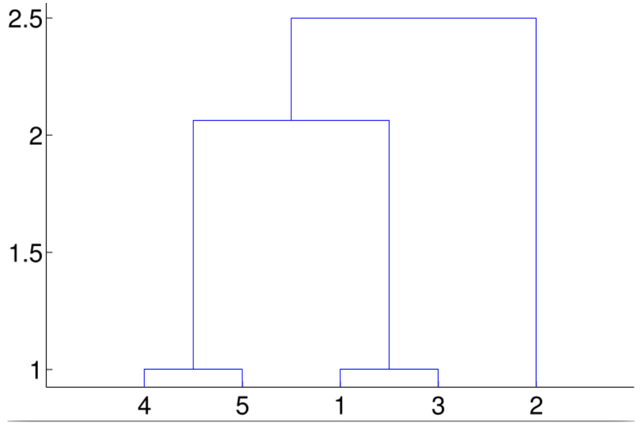
single linkage: distance between closest members

$$\text{distance}(c, c') = \min_{i \in \mathcal{C}_c, j \in \mathcal{C}_{c'}} \text{distance}(i, j)$$

example

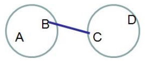
note that we can use the distance between clusters for the height of tree nodes in the dendrogram





Agglomorative clustering

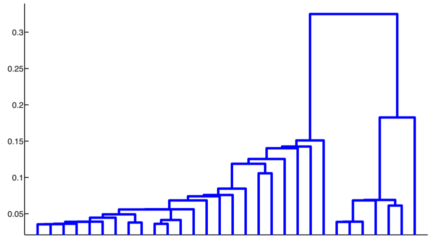
how to define dissimilarity of two clusters? some common choices:

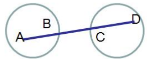


single linkage: distance between closest members

clusters can have members that are very far apart

$$\text{distance}(c, c') = \min_{i \in \mathcal{C}_c, j \in \mathcal{C}_{c'}} \text{distance}(i, j)$$

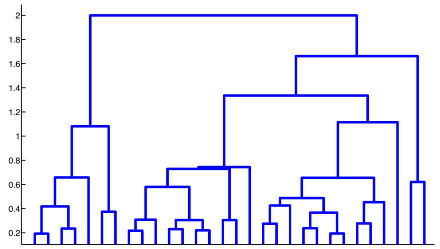


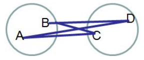


complete linkage: distance between furthest members

clusters that are more compact (all members should be close together)

$$\text{distance}(c, c') = \max_{i \in \mathcal{C}_c, j \in \mathcal{C}_{c'}} \text{distance}(i, j)$$

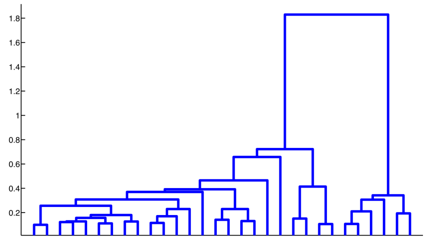




average linkage: average pairwise distance

a compromise between the above

$$\text{distance}(c, c') = \frac{1}{|\mathcal{C}_c||\mathcal{C}_{c'}|} \sum_{i \in \mathcal{C}_c, j \in \mathcal{C}_{c'}} \text{distance}(i, j)$$



How to Evaluate?

- Validate the algorithm on a set of benchmarks with known ground-truth
  - using a clustering agreement measure, e.g. **ARI**, NMI
  - useful in algorithm design

can be averaged over different datasets

$$A(C_1, G) > A(C_2, G)$$

```
>>> from sklearn.metrics.cluster import adjusted_rand_score
>>> adjusted_rand_score([0, 0, 1, 1], [0, 0, 1, 1])
1.0
>>> adjusted_rand_score([0, 0, 1, 1], [1, 1, 0, 0])
1.0
>>> adjusted_rand_score([0, 0, 1, 1], [0, 0, 1, 2])
0.57
>>> adjusted_rand_score([0, 0, 0, 0], [0, 1, 2, 3])
0.0
>>> adjusted_rand_score([0, 0, 1, 1], [0, 1, 0, 1])
-0.5
```

see more here

How to Evaluate?

- Validate the algorithm on a set of benchmarks with known ground-truth
  - using a clustering agreement measure, e.g. **ARI**, NMI
  - useful in algorithm design

can be averaged over different datasets

$$A(C_1, G) > A(C_2, G)$$

- Measure a relative criterion to compare goodness of different algorithms or hyperparameters (e.g. k in k-means) on your dataset

meaningful only on the same dataset

  - using a clustering quality index, e.g. **silhouette** score
  - useful in practice
  - measures (in different ways) within cluster to between cluster ratio

$$I(C_1) > I(C_2)$$

see more here

Summary

clustering can help us explore and understand our data
input to clustering methods can be features, similarities or graphs
clusters can be flat, hierarchical, overlapping, fuzzy...

we saw several clustering methods:

- K-means and K-medoid define clusters using **centers** and distance to these centers
  - optimization objective
  - algorithm to perform the optimization
- DB-SCAN as an example of **density-based** methods
- some heuristic **hierarchical** clustering methods

some notable methods we did not discuss

- popular community-mining methods such as modularity optimizing methods
- spectral clustering
- probabilistic generative models of clusters