CS 540 Introduction to Artificial Intelligence
Unsupervised Learning I

Yudong Chen
University of Wisconsin-Madison

Oct 5, 2021

Announcements

- **Homeworks:**
  - HW4 due next Tuesday
- Class roadmap:

Thursday, Sep 30	ML Intro
Tuesday, Oct 5	**ML Unsupervised I**
Thursday, Oct 7	ML Unsupervised II
Tuesday, Oct 12	ML Linear Regression
Thursday, Oct 14	ML: KNN, Naïve Bayes

Machine Learning

Recap of Supervised/Unsupervised

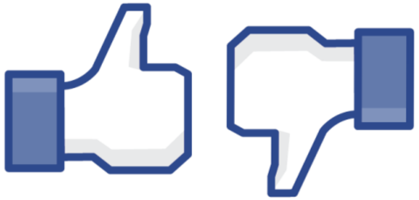
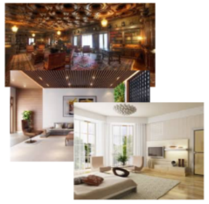
Supervised learning:

- Make predictions, classify data, perform regression
- Dataset: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

Features / Covariates / Input

Labels / Outputs

- Goal: find function $f : X \to Y$ to predict label on **new** data

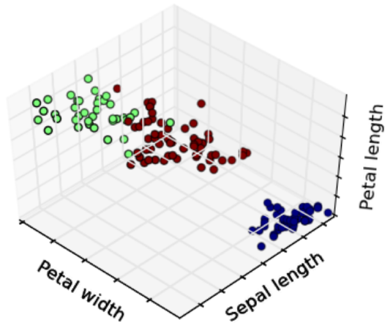


indoor

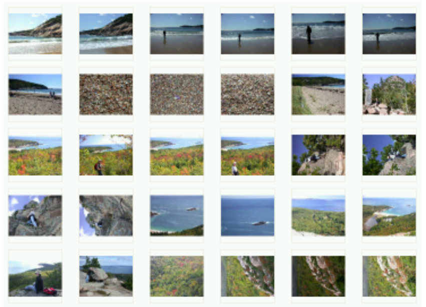
outdoor

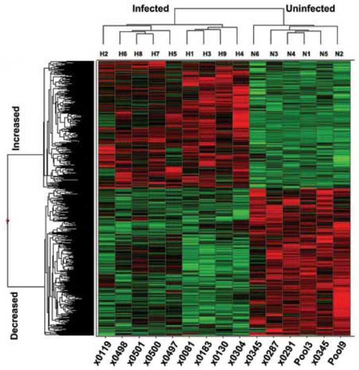
Recap of Supervised/Unsupervised

Unsupervised learning:

- No labels; generally won't be making predictions
- Dataset: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
- Goal: find patterns & structures that help better understand data.







Mulvey and Gingold

Recap of Reinforcement Learning

- Learn how to act in order to maximize rewards

DeepMind

- There are **other kinds** of ML:
  - Mixtures: semi-supervised learning, self-supervised

Outline

- Intro to Clustering
  - Clustering Types, Centroid-based, k-means review
- Hierarchical Clustering
  - Divisive, agglomerative, linkage strategies
- Other Clustering Types
  - Graph-based, cuts, spectral clustering

Unsupervised Learning & Clustering

- Note that clustering is just one type of unsupervised learning (**UL**)

- PCA is another unsupervised algorithm
- Estimating probability distributions also UL (GANs)

StyleGAN2 (Kerras et al ’20)

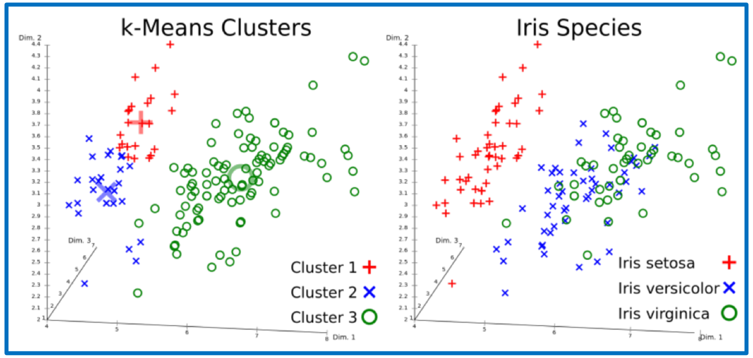
Clustering Types

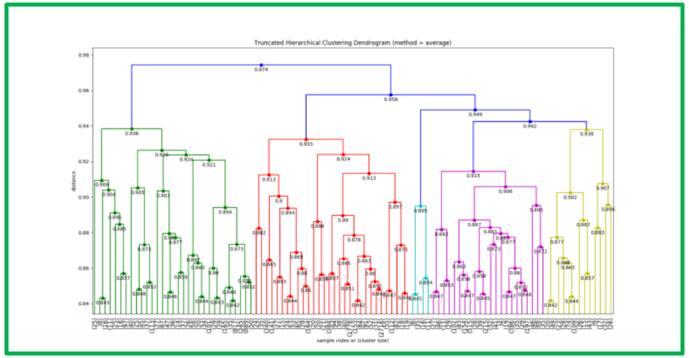
- Several types of clustering

Partitional
- Centroid
- Graph-theoretic
- Spectral

Hierarchical
- Agglomerative
- Divisive

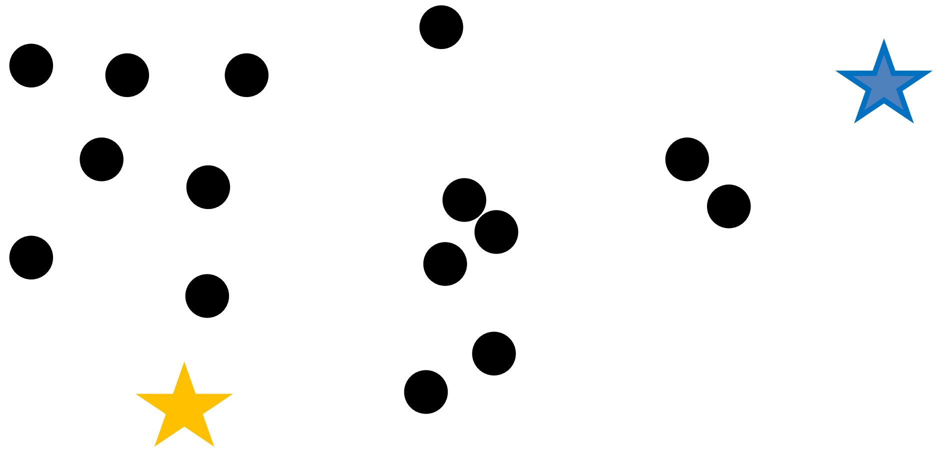
Bayesian
- Decision-based
- Nonparametric





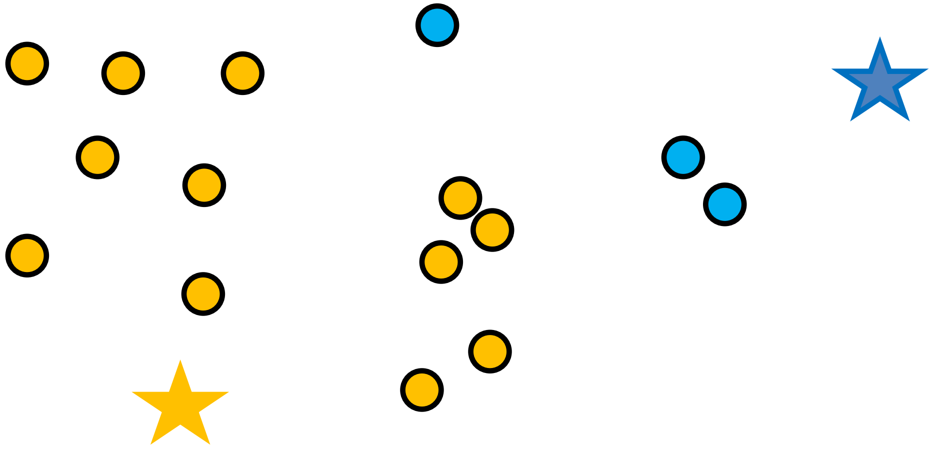
Clustering Types

- k-means is an example of partitional **centroid-based**
- Recall steps: **1.** Randomly pick k cluster centers



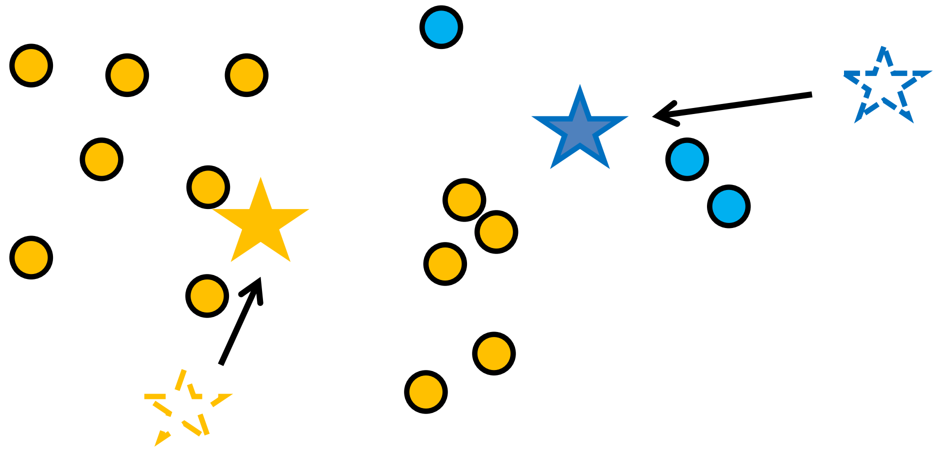
Clustering Types

- **2.** Find closest center for each point



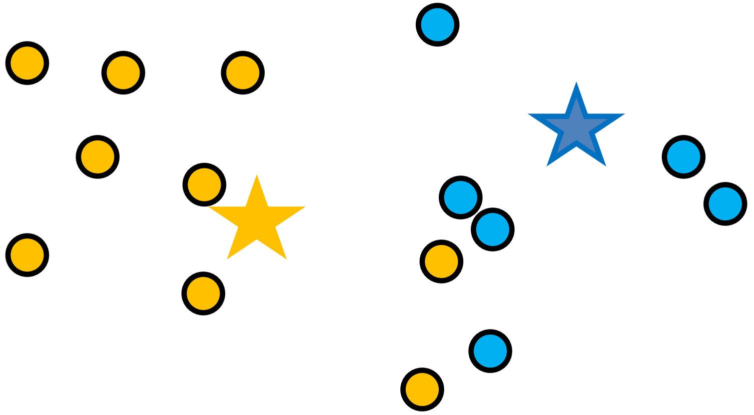
Clustering Types

- **3.** Update cluster centers by computing centroids



Clustering Types

- Repeat Steps 2 & 3 until convergence



Break & Quiz

Q 1.1: You have seven 2-dimensional points. You run 3-means on it, with initial clusters

$$C_1 = \{(2,2),(4,4),(6,6)\}, C_2 = \{(0,4),(4,0)\}, C_3 = \{(5,5),(9,9)\}$$

Cluster centroids at the next iteration are?

- A. C_1: (4,4), C_2: (2,2), C_3: (7,7)
- B. C_1: (6,6), C_2: (4,4), C_3: (9,9)
- C. C_1: (2,2), C_2: (0,0), C_3: (5,5)
- D. C_1: (2,6), C_2: (0,4), C_3: (5,9)

Break & Quiz

Q 1.2: We are running 3-means again. We have 3 centers, c_1=(0,1), c_2=(2,1), c_3=(-1,2). Which cluster assignment is possible for the points (1,1) and (-1,1), respectively? Ties are broken arbitrarily:

(i) c_1, c_1 (ii) c_2, c_3 (iii) c_1, c_3

- A. Only (i)
- B. Only (ii) and (iii)
- C. Only (i) and (iii)
- D. All of them

Break & Quiz

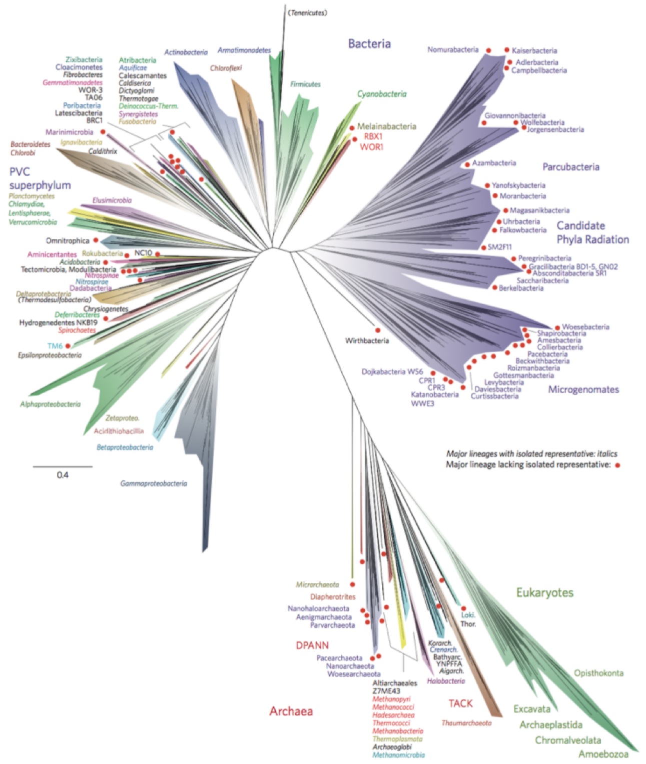
Q 1.3: If we run K-means clustering twice with random initial cluster centers, are we guaranteed to get same clustering results? Does K-means always converge?

- A. Yes, Yes
- B. No, Yes
- C. Yes, No
- D. No, No

Hierarchical Clustering

Basic idea: build a "hierarchy"

- One advantage: no need for k, number of clusters.
- **Input**: points in $\mathbb{R}^d$
- **Output**: a hierarchy
  - A binary tree

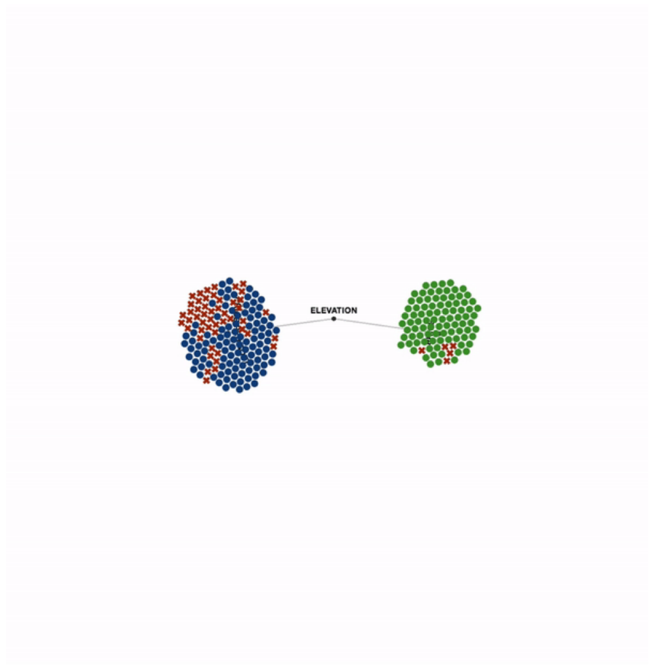


Credit: Wikipedia

Agglomerative vs Divisive

Two ways to go:

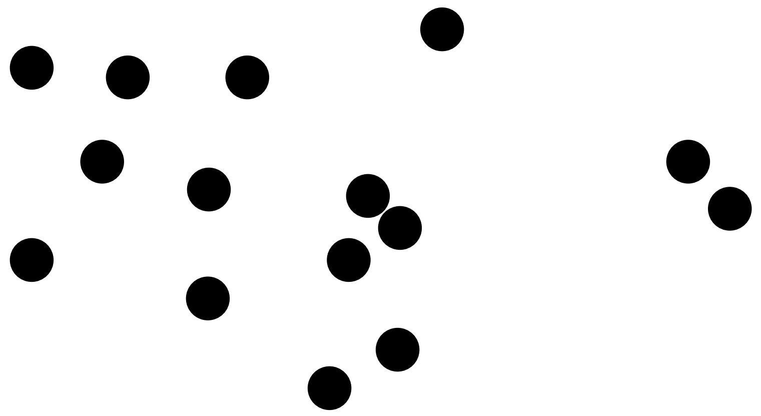
- **Agglomerative**: bottom up.
  - Start: each point a cluster. Progressively merge clusters
- **Divisive**: top down
  - Start: all points in one cluster. Progressively split clusters



Credit: r2d3.us

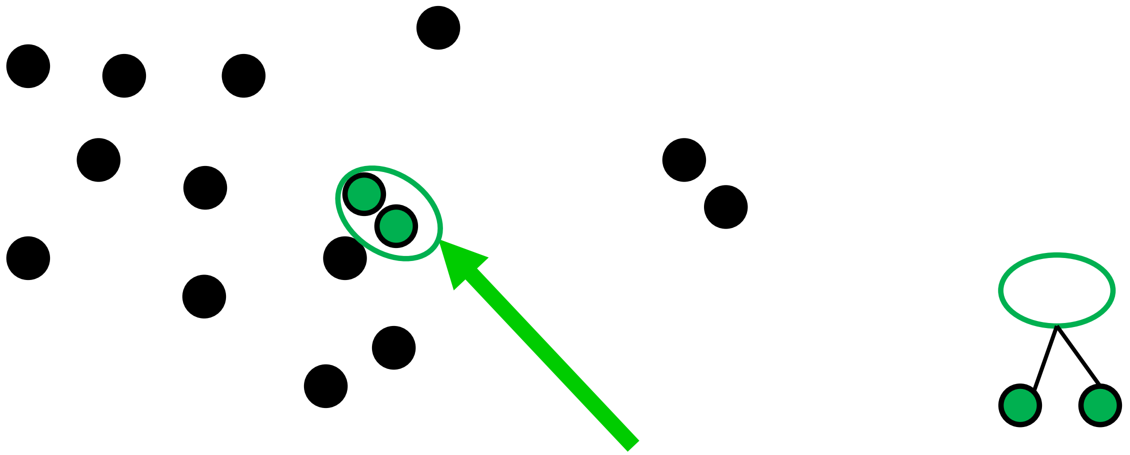
Agglomerative Clustering Example

Agglomerative. Start: every point is its own cluster



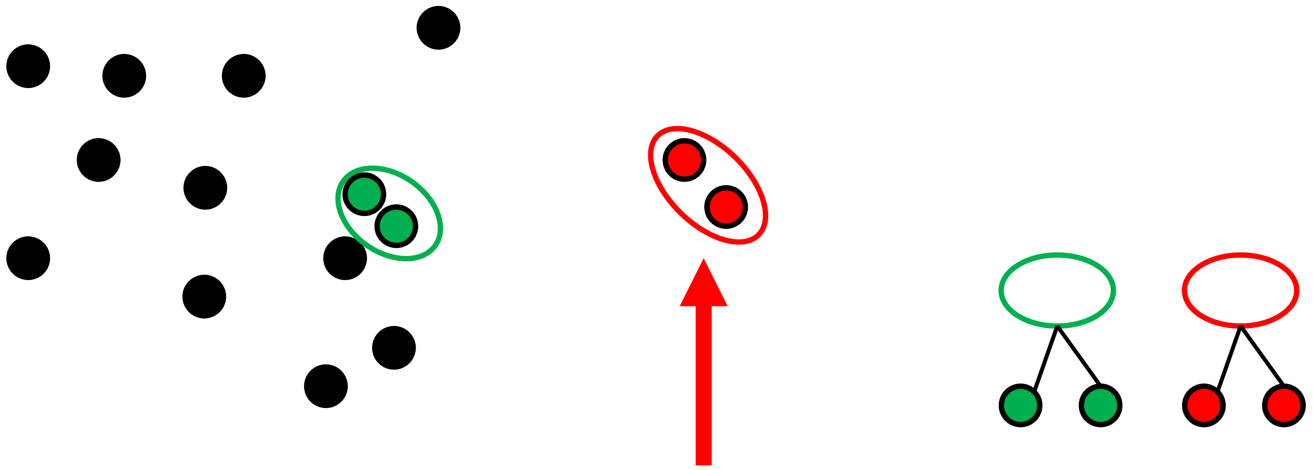
Agglomerative Clustering Example

Get pair of clusters that are closest and merge



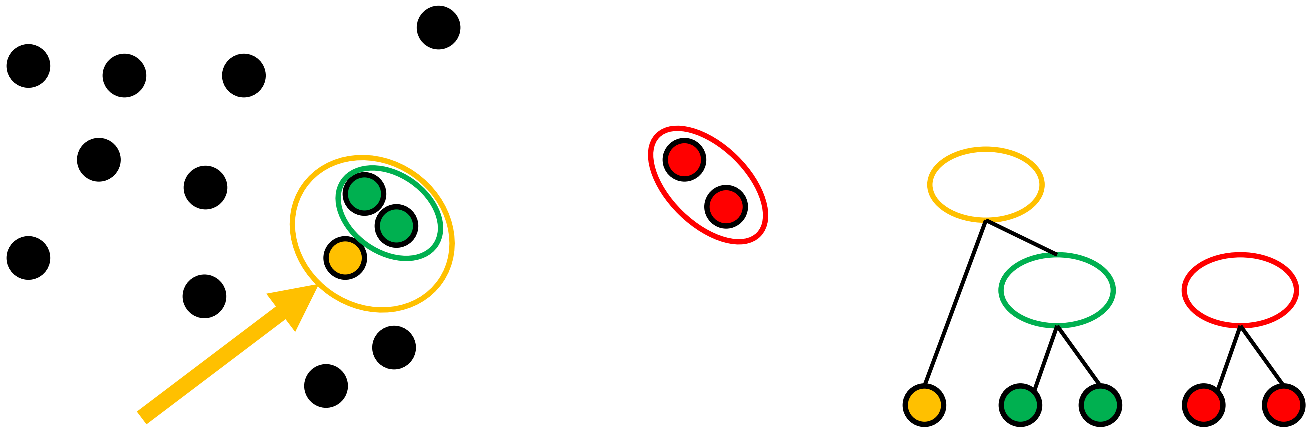
Agglomerative Clustering Example

Repeat: Get pair of clusters that are closest and merge



Agglomerative Clustering Example

Repeat: Get pair of clusters that are closest and merge



Merging Criteria

Merge: use closest clusters. Define closest?

- Single-linkage

$$d(A,B) = \min_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

- Complete-linkage

$$d(A,B) = \max_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

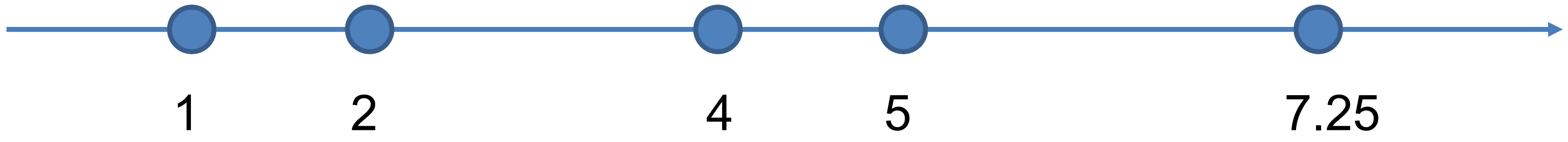
- Average-linkage

$$d(A,B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

Single-linkage Example

We’ll merge using single-linkage

- 1-dimensional vectors.
- Initial: all points are clusters

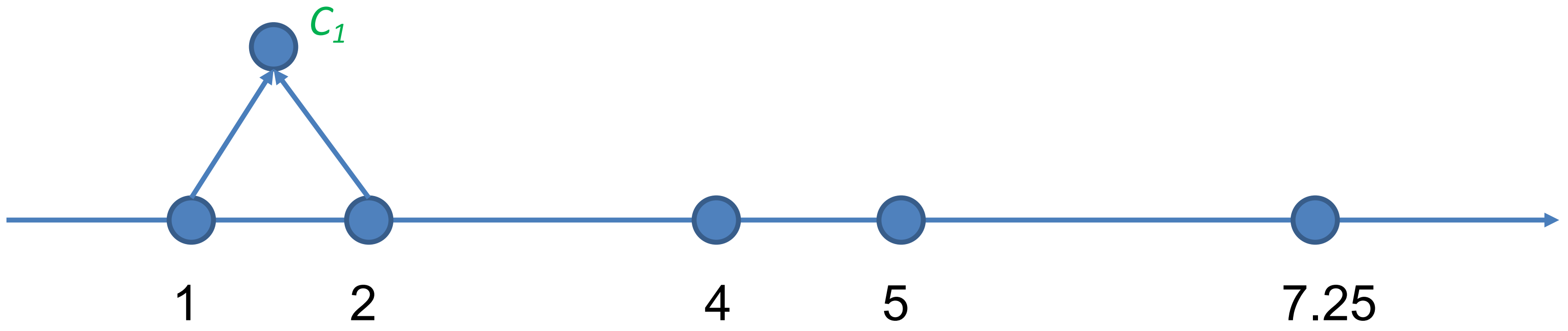


Single-linkage Example

We'll merge using single-linkage

$$d(C_1, \{4\}) = d(2, 4) = 2$$

$$d(\{4\}, \{5\}) = d(4, 5) = 1$$

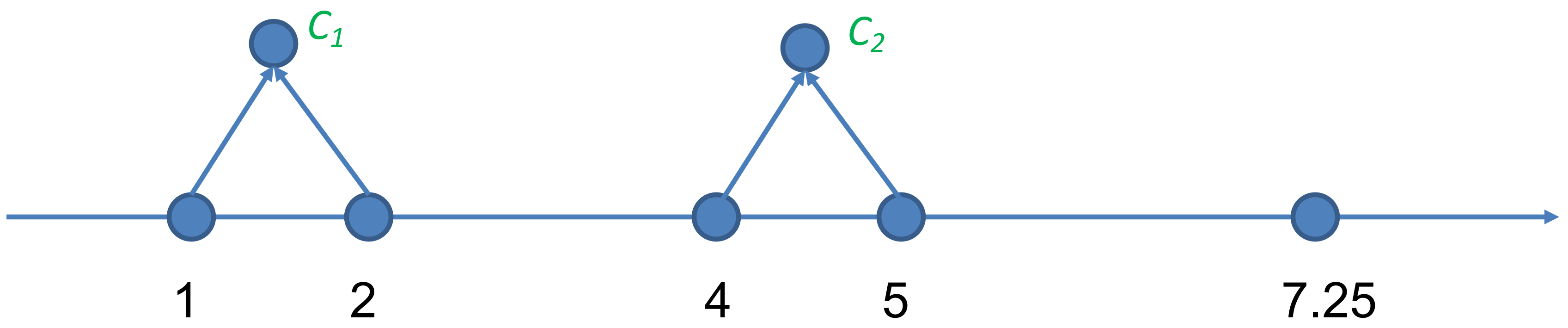


Single-linkage Example

Continue…

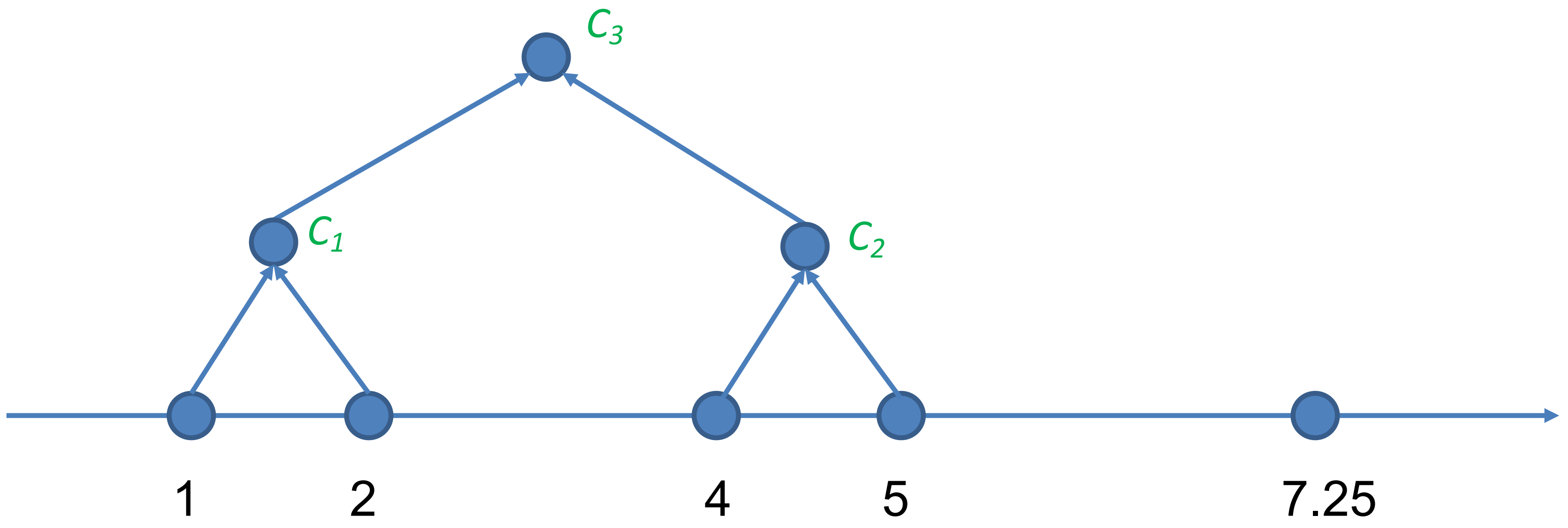
$$d(C_1, C_2) = d(2, 4) = 2$$

$$d(C_2, \{7.25\}) = d(5, 7.25) = 2.25$$

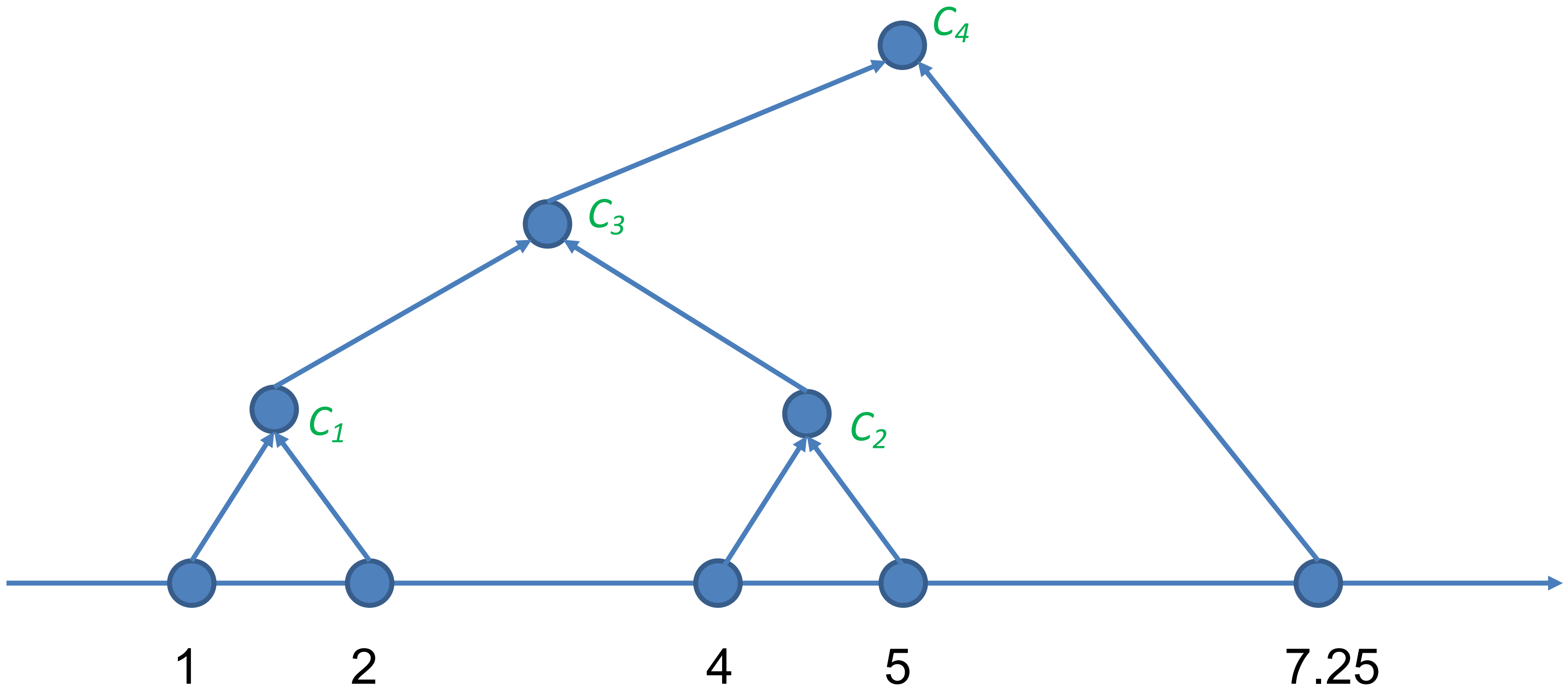


Single-linkage Example

Continue…



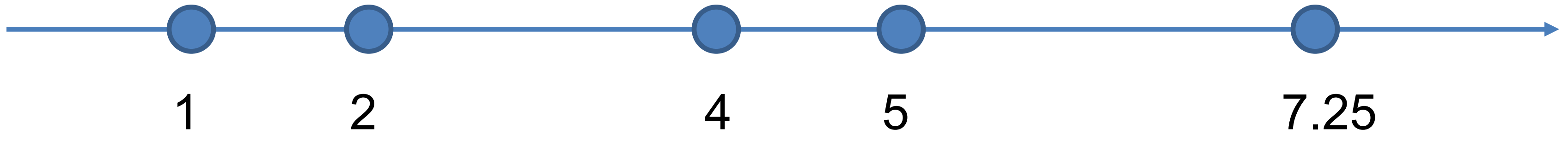
Single-linkage Example



Complete-linkage Example

We'll merge using complete-linkage

- 1-dimensional vectors.
- Initial: all points are clusters

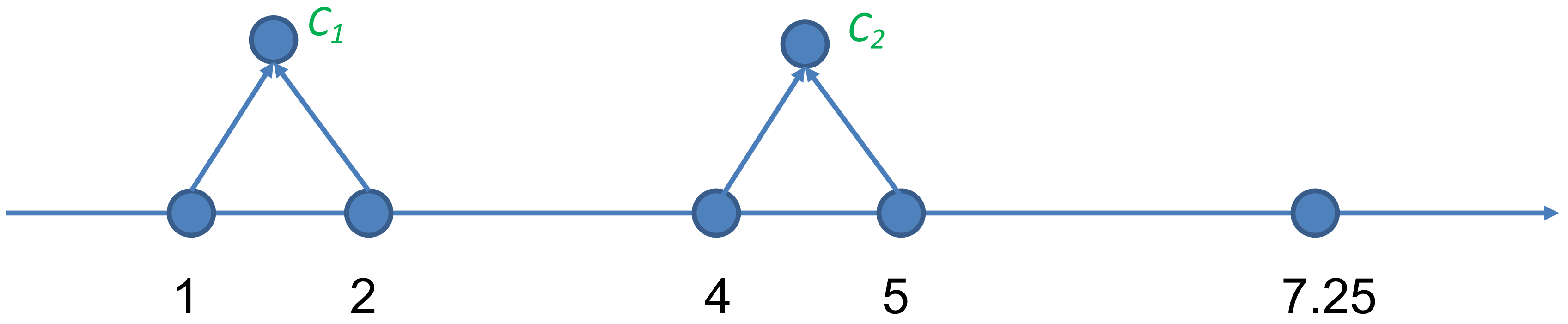


Complete-linkage Example

Beginning is the same…

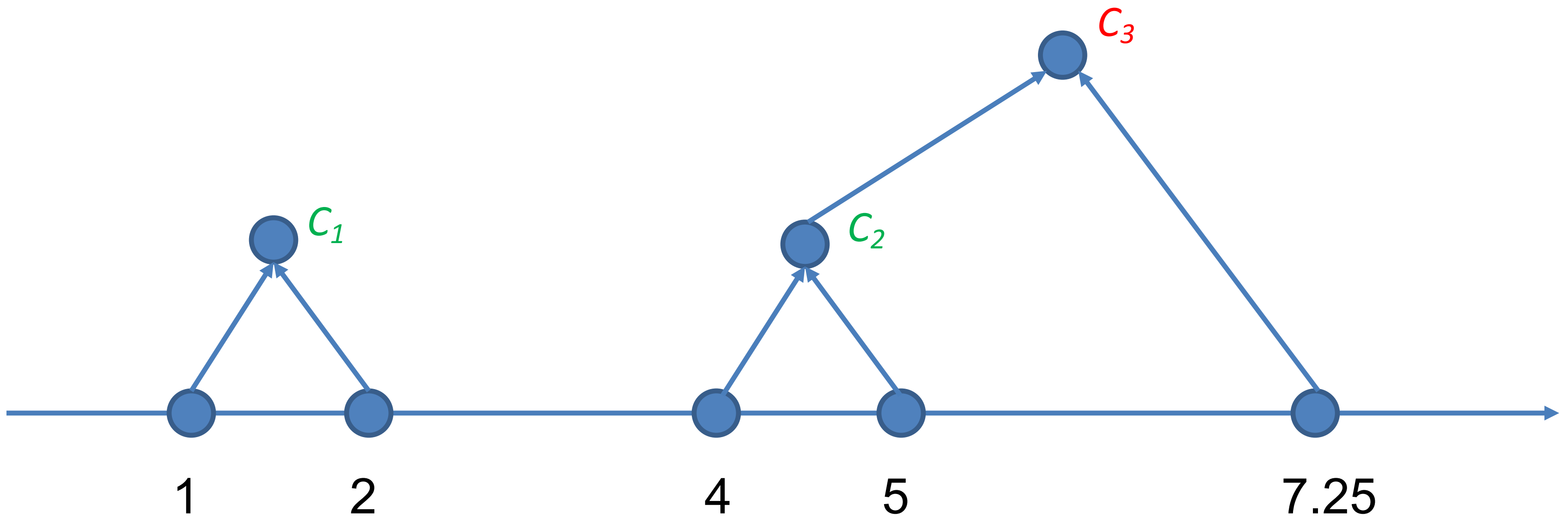
$$d(C_1, C_2) = d(1, 5) = 4$$

$$d(C_2, \{7.25\}) = d(4, 7.25) = 3.25$$

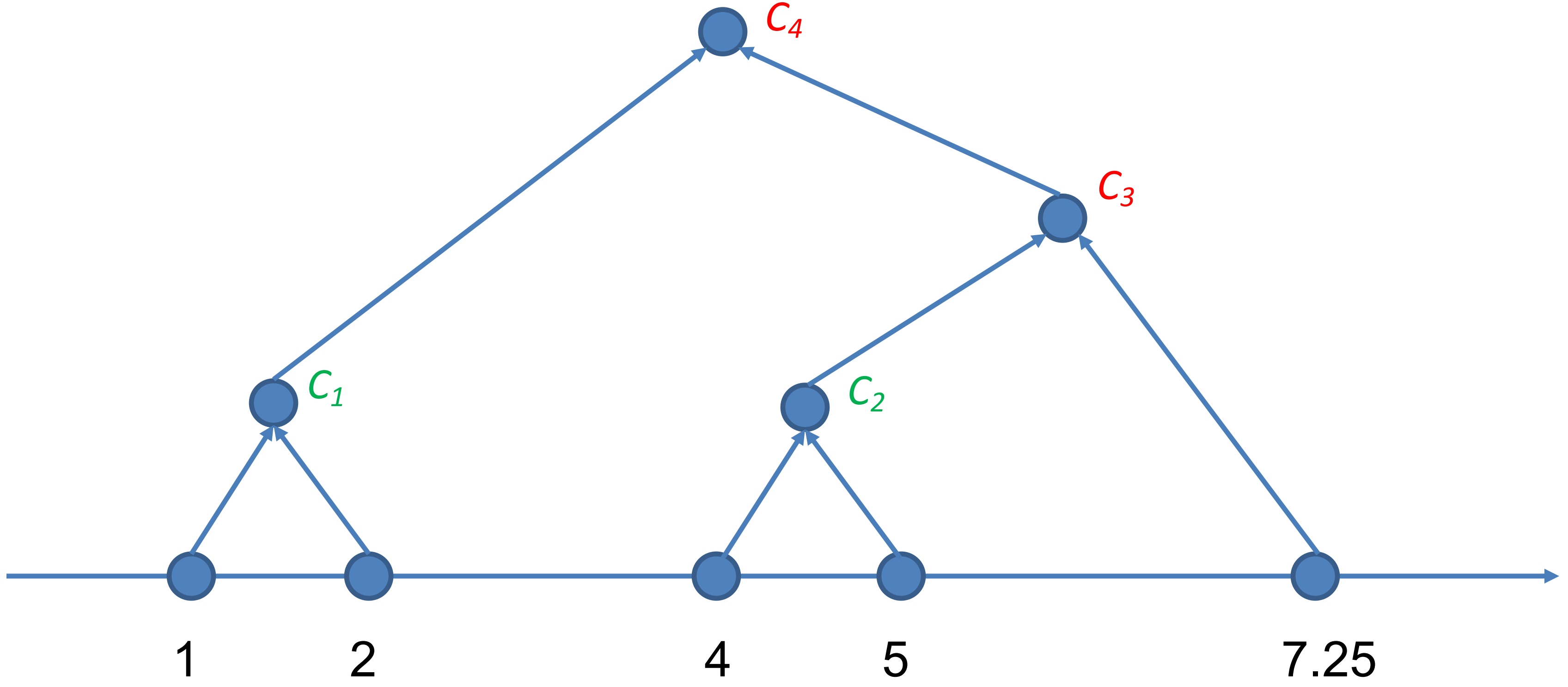


Complete-linkage Example

Now we diverge:



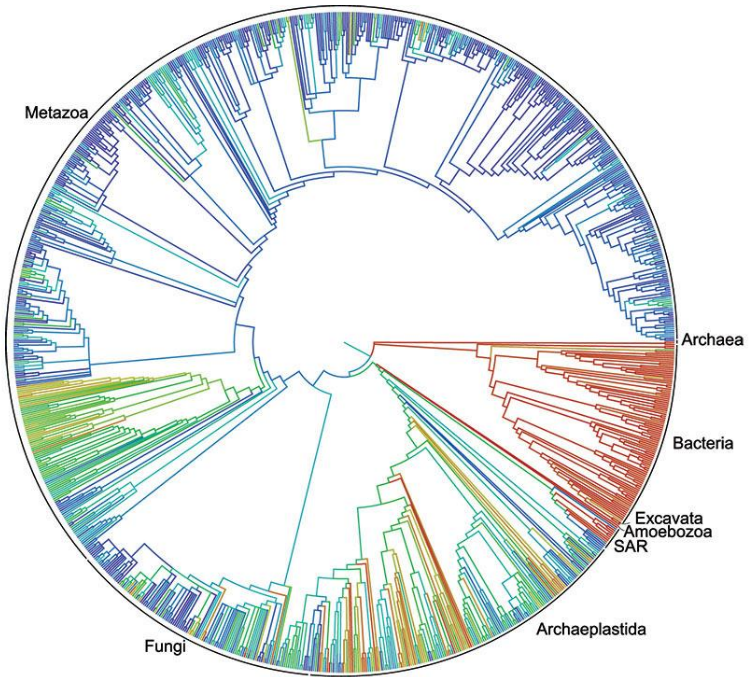
Complete-linkage Example



When to Stop?

No simple answer:

- Use the binary tree (a **dendogram**)
- Cut at different levels (get different heights/depths)

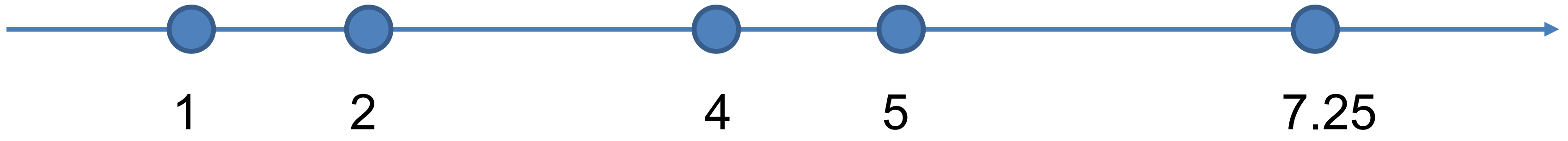


http://opentreeoflife.org/

Break & Quiz

Q 2.1: Let’s do hierarchical clustering for **two** clusters with average linkage on the dataset below. What are the clusters?

- A. {1}, {2,4,5,7.25}
- B. {1,2}, {4, 5, 7.25}
- C. {1,2,4}, {5, 7.25}
- D. {1,2,4,5}, {7.25}



Break & Quiz

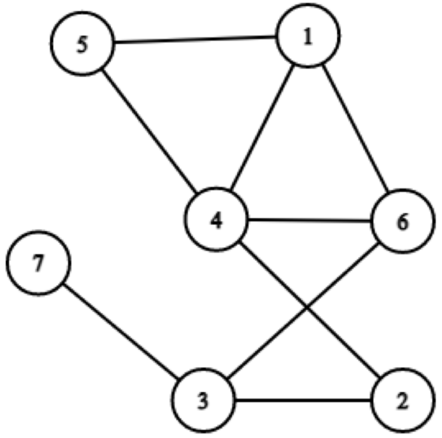
Q 2.2: If we do hierarchical clustering on *n* points, the maximum depth of the resulting tree is

- A. 2
- B. log *n*
- C. *n*/2
- D. *n*-1

Other Types of Clustering

Graph-based/proximity-based

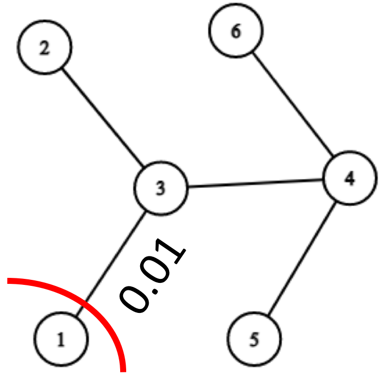
- Recall: Graph G = (V,E) has vertex set V, edge set E.
  - Edges can be weighted or unweighted
  - Encode **similarity**
- Don’t need vectors here
  - Just edges (and maybe weights)

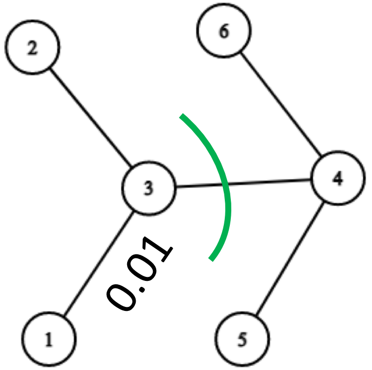


Graph-Based Clustering

Want: partition V into V_1 and V_2

- Implies a graph "cut"
- One idea: minimize the **weight** of the cut
  - Downside: might just cut of one node
  - Need: "**balanced**" cut





Partition-Based Clustering

Want: partition V into V_1 and V_2

- Just minimizing weight isn't good… want **balance!**
- **Approaches:**

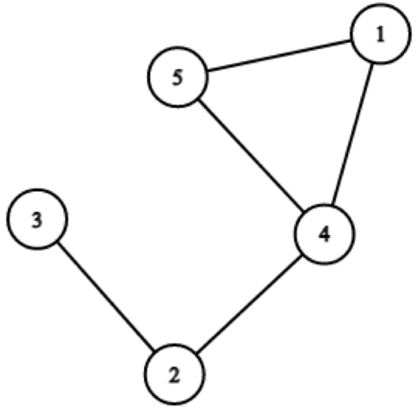
$$\mathrm{CCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{|V_1|} + \frac{\mathrm{Cut}(V_1, V_2)}{|V_2|}$$

$$\mathrm{NCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{\sum_{\imath \in V_1} d_i} + \frac{\mathrm{Cut}(V_1, V_2)}{\sum_{\imath \in V_2} d_i}$$

Partition-Based Clustering

How do we compute these?

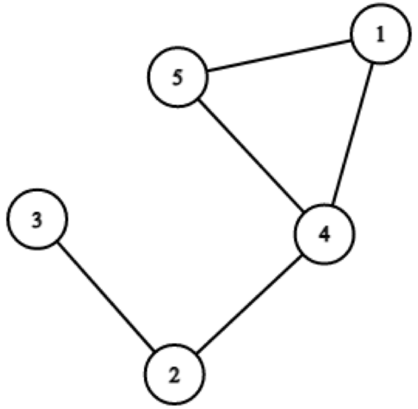
- Hard problem → heuristics
  - Greedy algorithm
  - "Spectral" approaches
- Spectral clustering approach:
  - **Adjacency** matrix



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Partition-Based Clustering

- Spectral clustering approach:
  - **Adjacency** matrix
  - **Degree** matrix

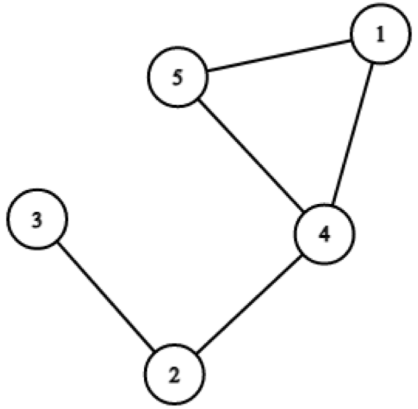


$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Spectral Clustering

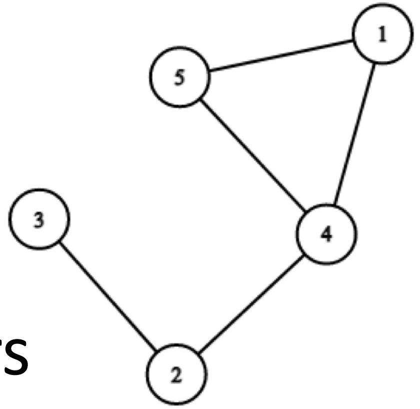
- Spectral clustering approach:
  - 1. Compute **Laplacian** L = D – A

(Important tool in graph theory)



$$L = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}}_{\text{Degree Matrix}} - \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Adjacency Matrix}} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}}_{\text{Laplacian}}$$

Spectral Clustering



- Spectral clustering approach:
  - 1. Compute **Laplacian L** = **D** – **A**
  - 2. Compute *k* **smallest** eigenvectors
  - 3. Set *U* to be the *n* x *k* matrix with $u_1, \ldots, u_k$ as columns. Treat *n* rows as *n* points in $\mathbb{R}^k$
  - 4. Run k-means on the representations

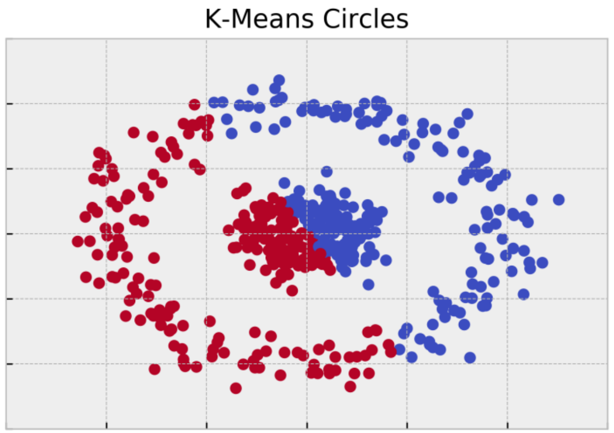
Spectral Clustering

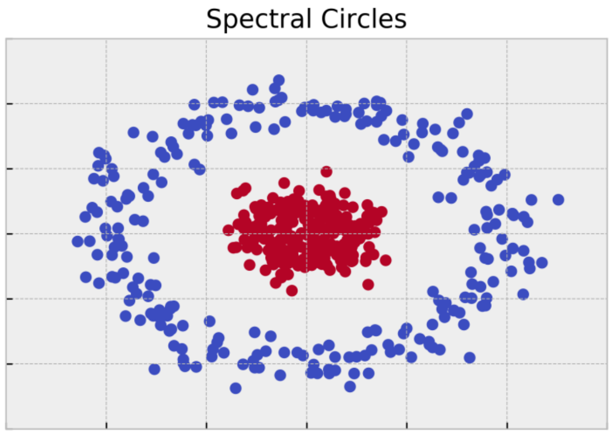
- Compare/contrast to **PCA**:
  - Use an **eigendecomposition /** dimensionality reduction
    - But, run on Laplacian (not covariance); use smallest eigenvectors, not largest
- Intuition: Laplacian encodes structure information
  - “Lower” eigenvectors give partitioning information

Spectral Clustering

Q: Why do this?

- 1. No need for points or distances as input
- 2. Can handle intuitive separation (k-means can't!)





Credit: William Fleshman