CS 540 Introduction to Artificial Intelligence
Unsupervised Learning I

Yingyu Liang
University of Wisconsin-Madison
Oct 5, 2021

Based on slides by Fred Sala

Recap of Supervised/Unsupervised

Supervised learning:

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

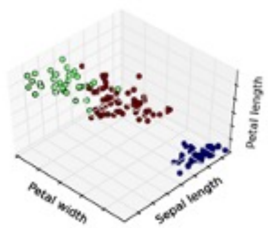
Features / Covariates / Input Labels / Outputs

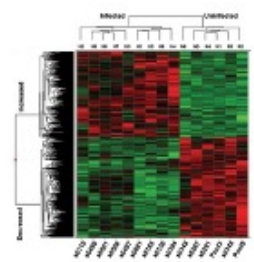
- Goal: find function $f : X \rightarrow Y$ to predict label on **new** data

Recap of Supervised/Unsupervised

Unsupervised learning:

- No labels; generally won't be making predictions
- Dataset: $x_1, x_2, \ldots, x_n$
- Goal: find patterns & structures that help better understand data.





Mulvey and Gingold

Recap of Supervised/Unsupervised

Note that there are **other kinds** of ML:

- Mixtures: semi-supervised learning, self-supervised
  - Idea: different types of "signal"
- Reinforcement learning
  - Learn how to act in order to maximize rewards
  - Later on in course…

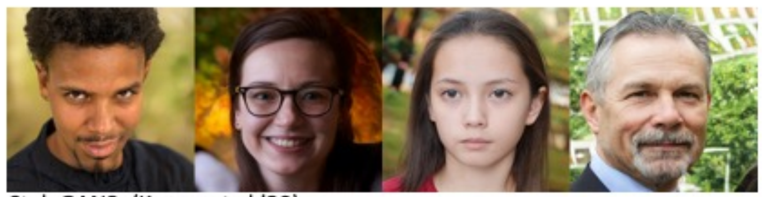
DeepMind

Outline

- Intro to Clustering
  - Clustering Types, Centroid-based, k-means review
- Hierarchical Clustering
  - Divisive, agglomerative, linkage strategies

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)
- Clustering is popular & useful!

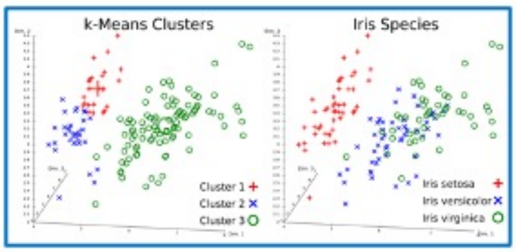


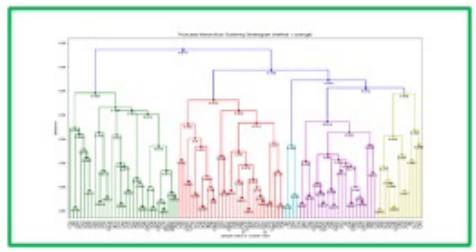
StyleGAN2 (Kerras et al '20)

Clustering Types

- Several types of clustering

Partitional	**Hierarchical**	**Bayesian**
- Centroid	- Agglomerative	- Decision-based
- Graph-theoretic	- Divisive	- Nonparametric
- Spectral		





Partitional: to get a partition (ie, a set of disjoint clusters whose union is the whole dataset)

1) centroid: use centers and assign data points to centers to form clusters
2) Graph-theoretical: the input is a graph (instead of a set of numeric vectors), and would like to partition the nodes into clusters
3) Spectral: an approach for doing graph clustering

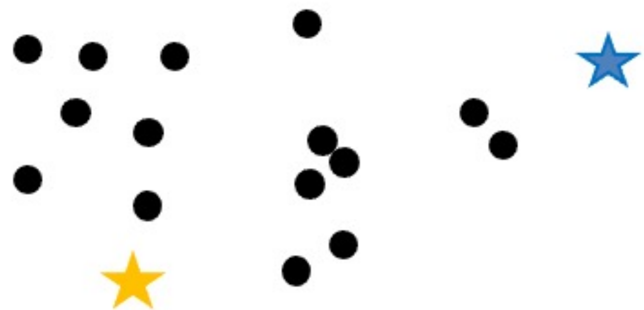
Hierarchical: to get a tree on the data points

1) Agglomerative: begin with each point as a singleton cluster, and keep merging them until all are merged into one cluster
2) Divisive: begin with all points in one cluster, and keep splitting the clusters to smaller ones until containing only one point (or satisfying some other stopping criteria)

Bayesian: a family of methods using Bayes' rule to do clustering. Can produce a partition or a tree. Not covered in this course.

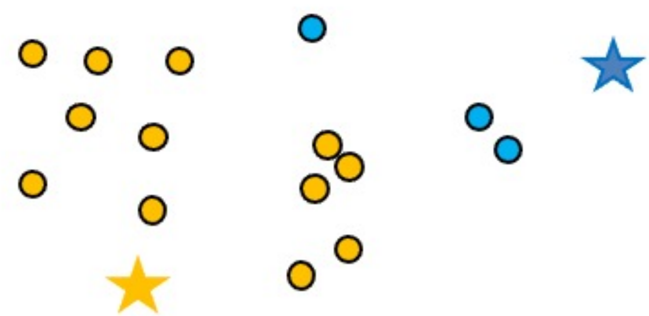
Center-based Clustering

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



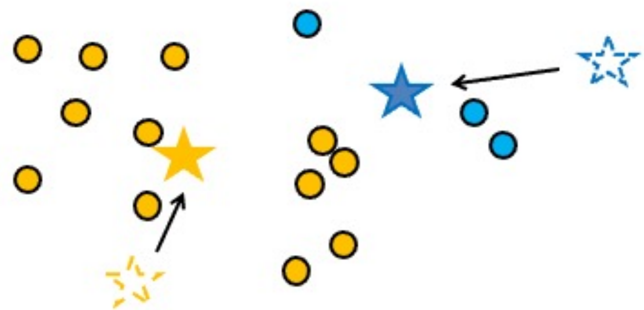
Center-based Clustering

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



Center-based Clustering

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



Center-based Clustering

- 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 1.1: 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)

The average of points in C1 is (4,4).
The average of points in C2 is (2,2).
The average of points in C3 is (7,7).

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 1.2: 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**

For the point (1,1): square-Euclidean-distance to C1 is 1, to C2 is 1, to C3 is 5
So it can be assigned to C1 or C2

For the point (-1,1): square-Euclidean-distance to C1 is 1, to C2 is 9, to C3 is 1
So it can be assigned to C1 or C3

Break & Quiz

Q 1.3: If we run K-means clustering twice with random starting 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

Break & Quiz

Q 1.3: If we run K-means clustering twice with random starting 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

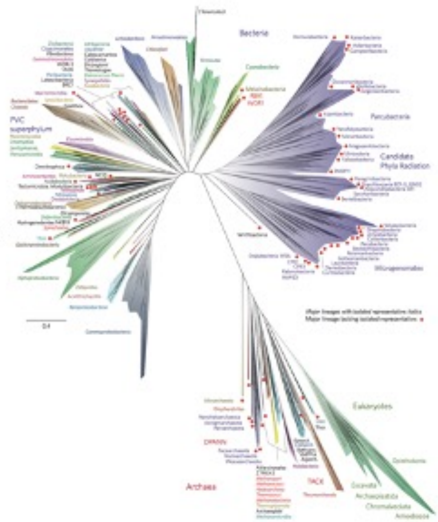
The clustering from k-means will depend on the initialization. Different initialization can lead to different outcomes.

K-means will always converge on a finite set of data points:
1. There are finite number of possible partitions of the points
2. The assignment and update steps of each iteration will only decrease the sum of the distances from points to their corresponding centers.
3. If it run forever without convergence, it will revisit the same partition, which is contradictory to item 2.

Hierarchical Clustering

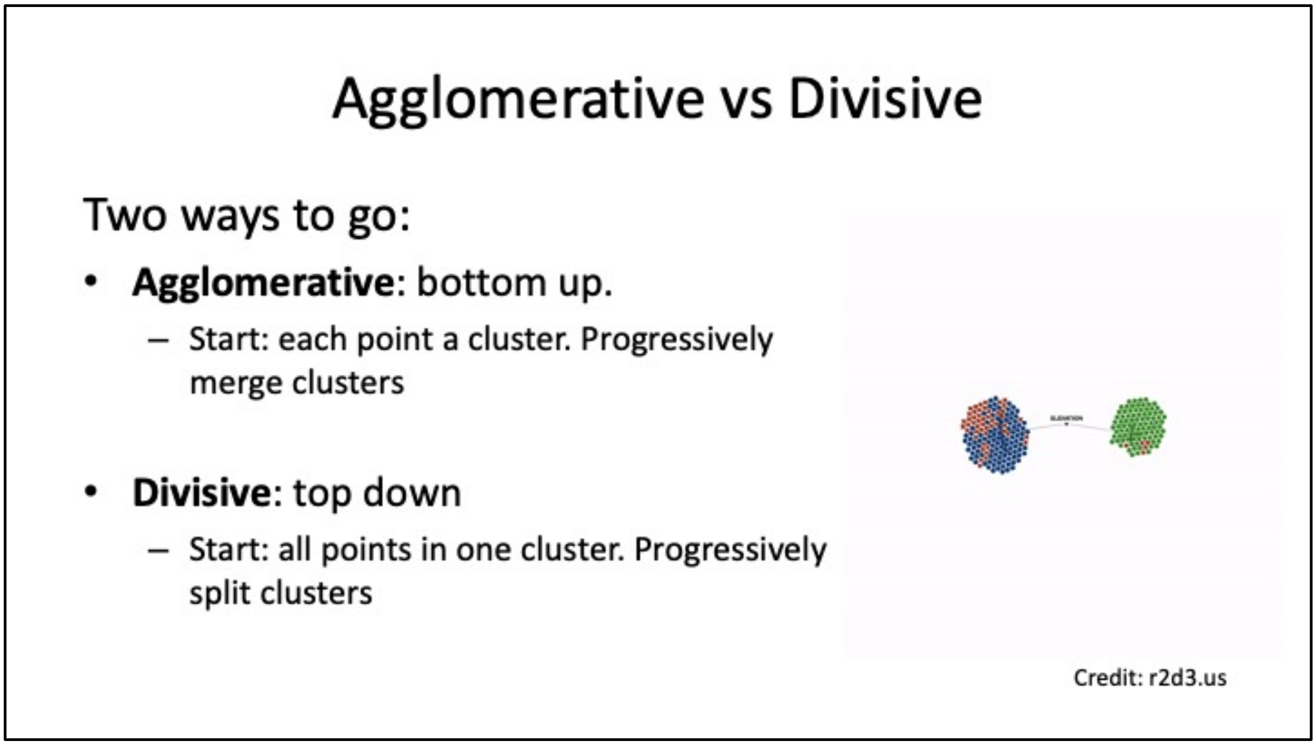
Basic idea: build a "hierarchy"

- Want: arrangements from specific to general
- One advantage: no need for k, number of clusters.
- **Input**: points. **Output**: a hierarchy
  - A binary tree



Credit: Wikipedia

Typically, the algorithms build a binary tree (each node only has 2 children). Sometimes can be a tree with branching factor more than 2.

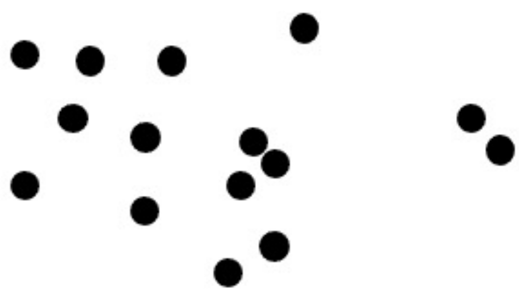


Hierarchical: to get a tree on the data points

1) Agglomerative: begin with each point as a singleton cluster, and keep merging them until all are merged into one cluster
2) Divisive: begin with all points in one cluster, and keep splitting the clusters to smaller ones until containing only one point (or satisfying some other stopping criteria)

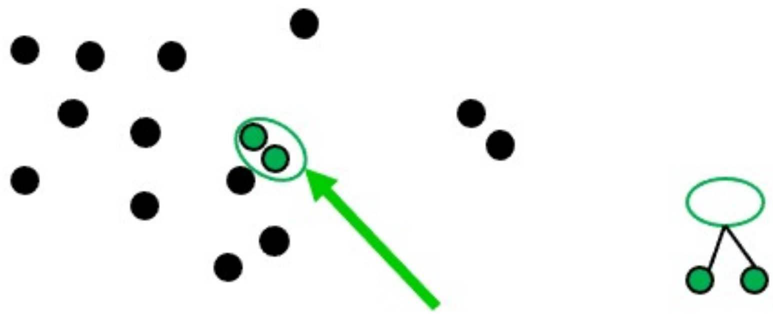
Agglomerative Clustering Example

Agglomerative. Start: every point is its own cluster



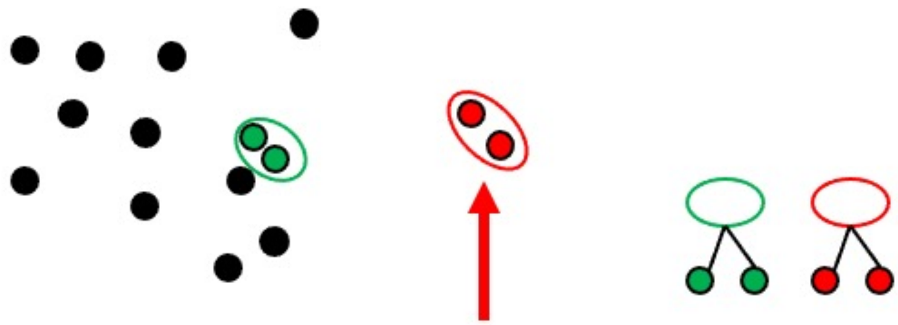
Agglomerative Clustering Example

Get pair of clusters that are closest and merge



Agglomerative Clustering Example

Repeat: Get pair of clusters that are closest and merge

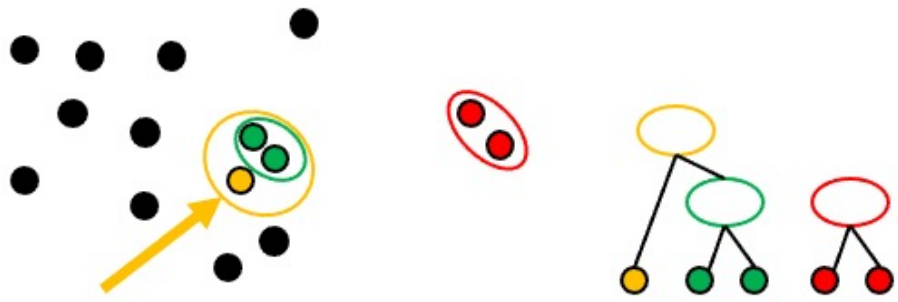


Keep merging the closest pair of clusters.

We only have a definition of distance between data points. Need a definition of distance between clusters!

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

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

We can have different definitions of distances between clusters, which lead to different algorithms.

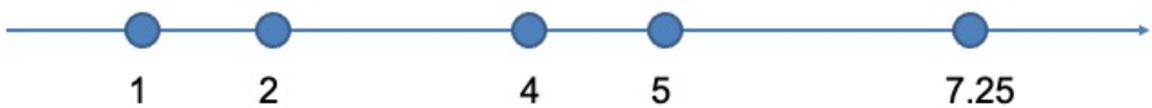
Once we have the definition, we can compute the distances and find the closest pair of clusters and merge them.

Note: in complete-linkage, we find the closest pair of clusters by
(A*, B*) = argmin_{clusters A B} d(A,B) = = argmin_{clusters A B} max_{x1 \in A, x2 \in B} d(x1, x2)
Do not confuse the max over data points with the min over clusters. That is, while we compute the distance between clusters, we take the maximum over the points; but we are still looking for the closest pair of clusters, not the farthest pair of clusters.

Single-linkage Example

We'll merge using single-linkage

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

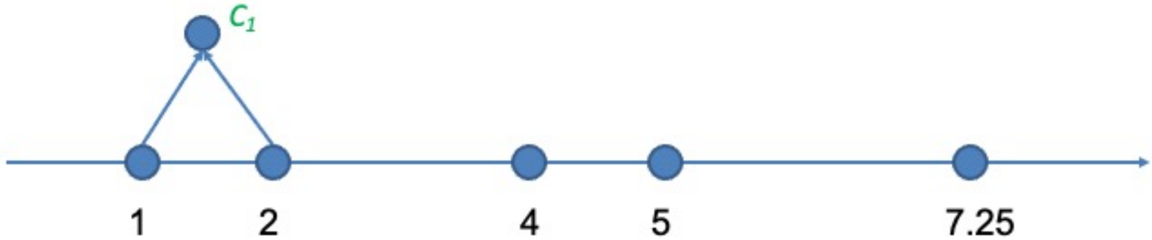


Single-linkage Example

We'll merge using single-linkage

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

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

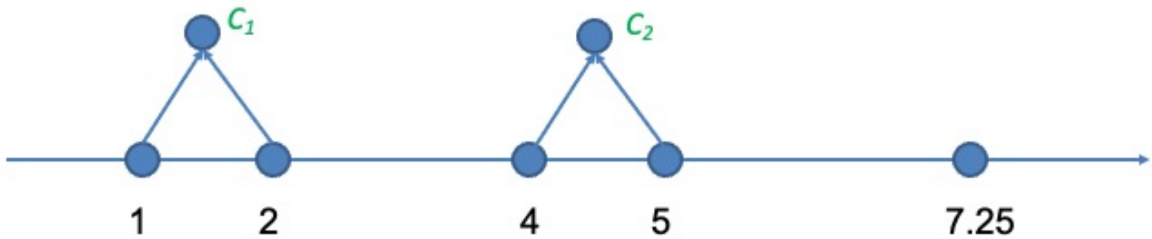


Single-linkage Example

Continue…

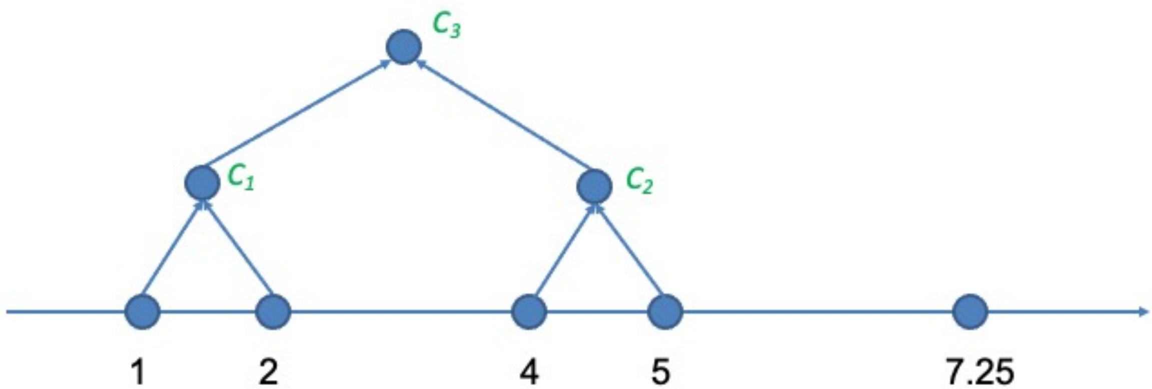
$$d(C_1, C_2) = d(2, 4) = 2$$

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

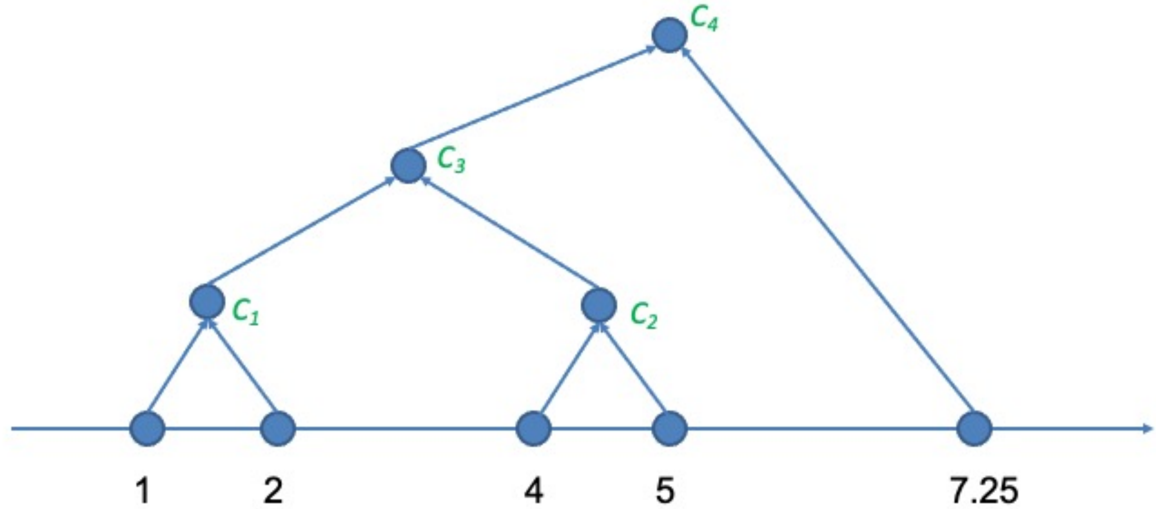


Single-linkage Example

Continue…



Single-linkage Example



Complete-linkage Example

We'll merge using complete-linkage

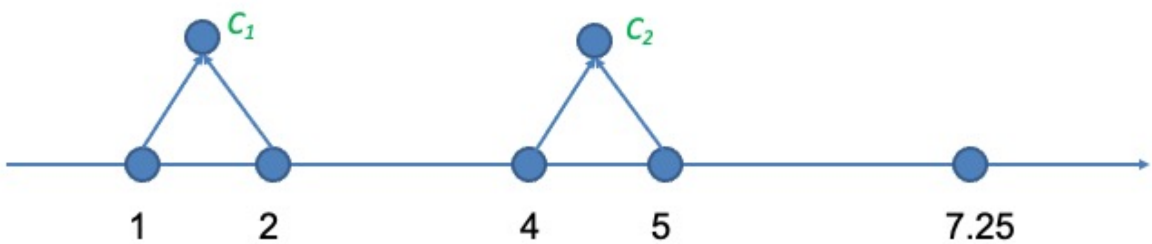
- 1-dimensional vectors.
- Initial: all points are clusters

Complete-linkage Example

Beginning is the same…

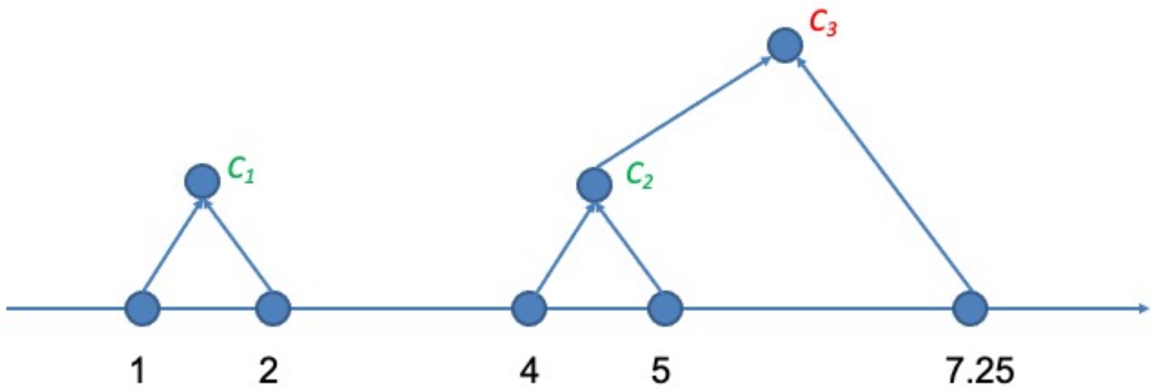
$$d(C_1, C_2) = d(1, 5) = 4$$

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

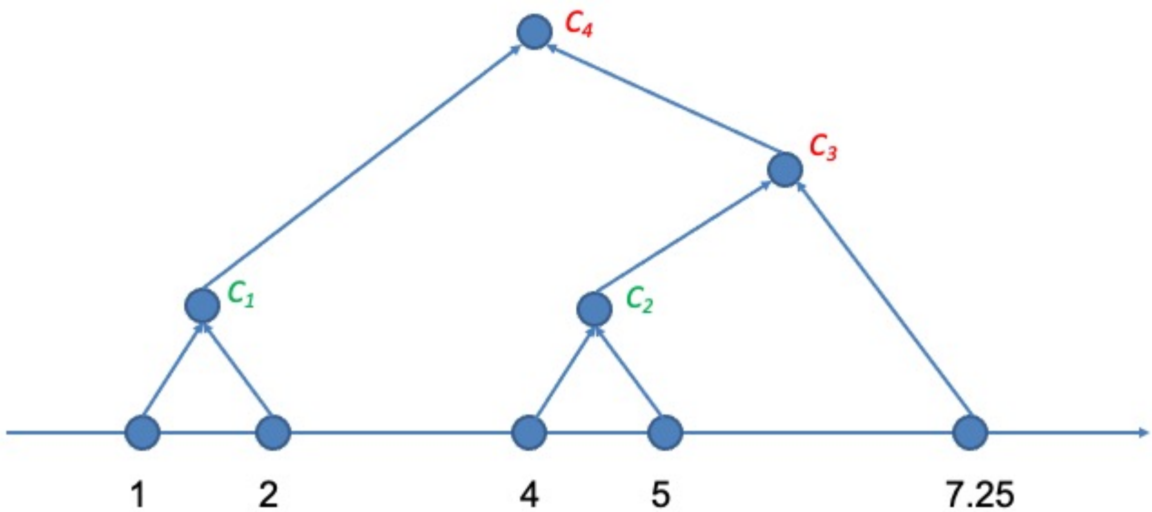


Complete-linkage Example

Now we diverge:



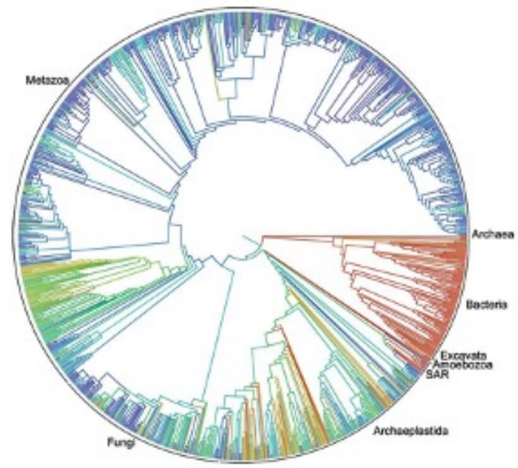
Complete-linkage Example



When to Stop?

No simple answer:

- Use the binary tree (a **dendogram**)
- Cut at different levels (g different heights/depth:



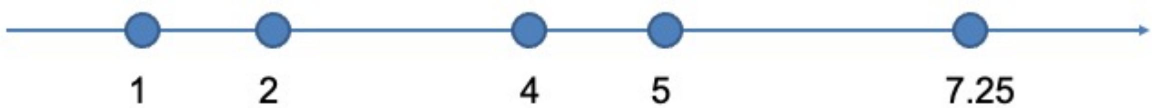
http://opentreeoflife.org/

Typical in practice: merge until only one cluster (the root). Then cut at different levels to get different partitions; number of clusters or the cut level is application-dependent.

Break & Quiz

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

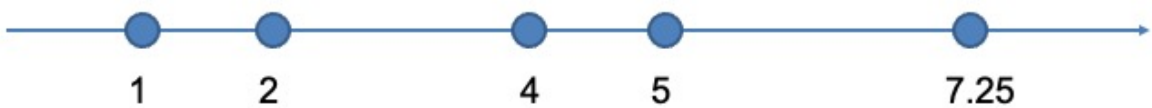
- A. {1}, {2,4,5,7.25}
- B. {1,2}, {4, 5, 7.25}
- C. {1,2,4}, {5, 7.25}
- D. {1,2,4,5}, {7.25}



Break & Quiz

Q 2.1: 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}



Iteration 1: merge 1 and 2
Iteration 2: merge 4 and 5
Iteration 3: Now we have clusters {1,2}, {4,5}, {7.25}.
distance({1,2}, {4,5})= 3
distance({4,5}, {7.25}) = 2.75
distance({1,2}, {7.25}) is clearly larger than the above two.
So average linkage will merge {4,5} and {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

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**

Denote the points as x_1, x_2, …, x_n

Suppose:
in iteration 1, we merge points x_1 and x_2
in iteration 2, we merge {x_1, x_2} with x_3
…
in iteration t, we merge {x_1, x_2, …, x_t} with x_{t+1}
…
in iteration n-1, we merge {x_1, x_{n-1}} with x_n

Then we will get a tree with depth n-1.