CSE/STAT 416

Hierarchical Clustering

Hunter Schafer
Paul G. Allen School of Computer Science & Engineering
University of Washington

May 12, 2021

Clustering

Define Clusters

In their simplest form, a **cluster** is defined by

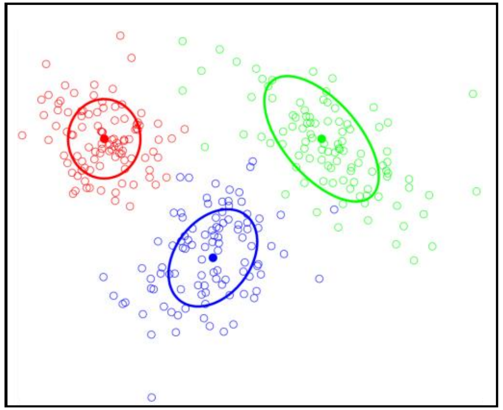
- The location of its center (**centroid**)
- Shape and size of its **spread**

Clustering is the process of finding these clusters and **assigning** each example to a particular cluster.

- x_i gets assigned $z_i \in [1, 2, \ldots, k]$
- Usually based on closest centroid

Will define some kind of score for a clustering that determines how good the assignments are

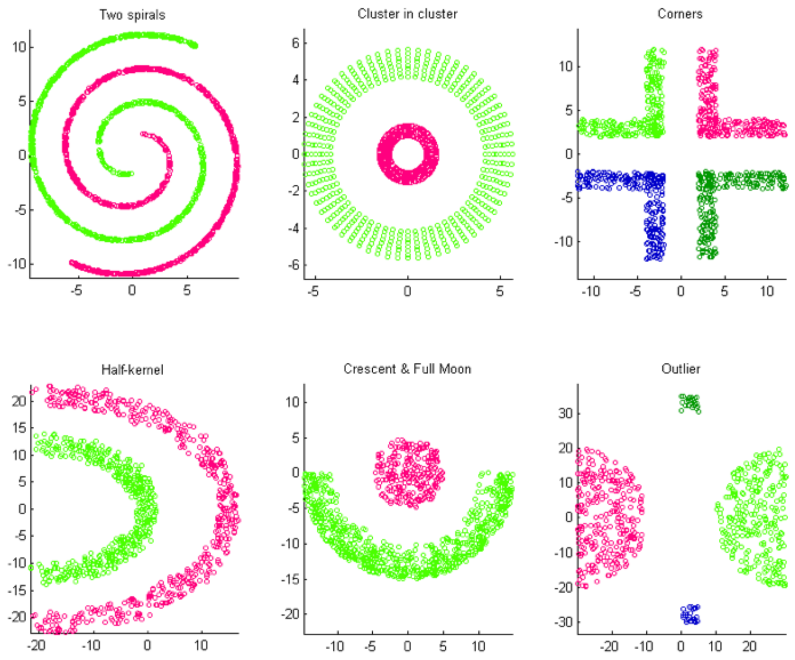
- Based on distance of assigned examples to each cluster



Not Always Easy

There are many clusters that are harder to learn with this setup

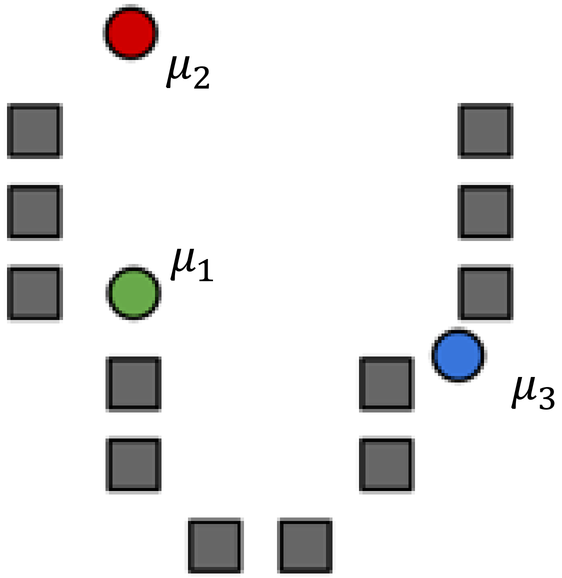
- Distance does not determine clusters



Step 0

Start by choosing the initial cluster centroids

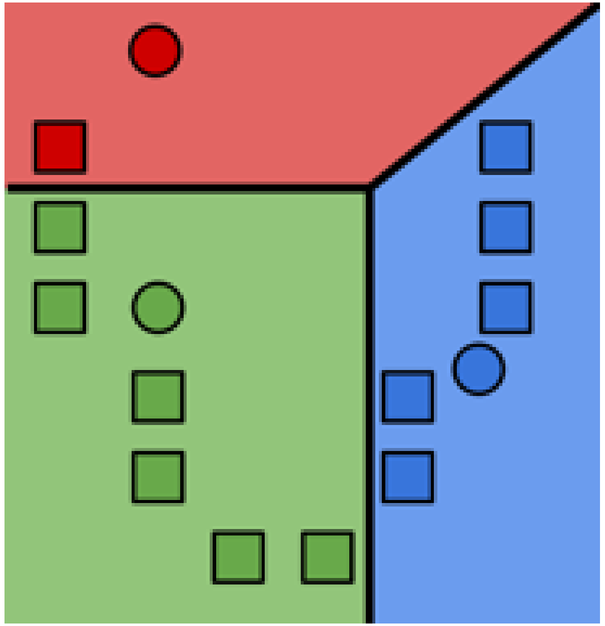
- A common default choice is to choose centroids at random
- Will see later that there are smarter ways of initializing



Step 1

Assign each example to its closest cluster centroid

$$z_i \leftarrow \underset{j \in [k]}{\operatorname{argmin}} \left|\left|\mu_j - x_i\right|\right|^2$$

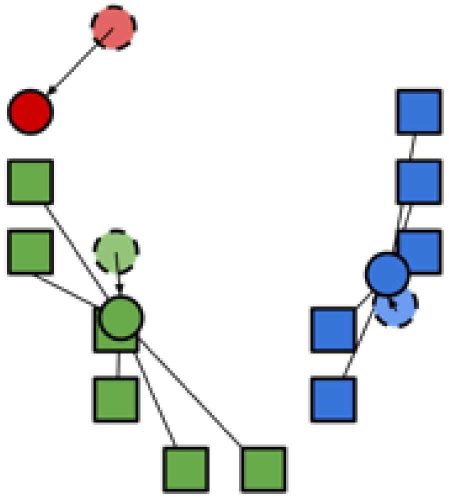


Step 2

Update the centroids to be the mean of all the points assigned to that cluster.

$$\mu_j \leftarrow \frac{1}{n_j} \sum_{i:z_i=j} x_i$$

Computes center of mass for cluster!



Smart Initializing w/ k-means++

Making sure the initialized centroids are “good” is critical to finding quality local optima. Our purely random approach was wasteful since it’s very possible that initial centroids start close together.

Idea: Try to select a set of points farther away from each other.

k-means++ does a slightly smarter random initialization

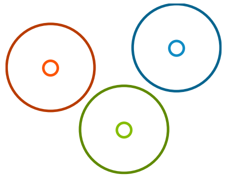
1. Choose first cluster μ_1 from the data uniformly at random
2. For the current set of centroids (starting with just μ_1), compute the distance between each datapoint and its closest centroid
3. Choose a new centroid from the remaining data points with probability of x_i being chosen proportional to $d(x_i)^2$
4. Repeat 2 and 3 until we have selected k centroids

Problems with k-means

In real life, cluster assignments are not always clear cut

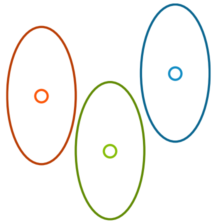
- E.g. The moon landing: Science? World News? Conspiracy?

Because we minimize Euclidean distance, k-means assumes all the clusters are spherical



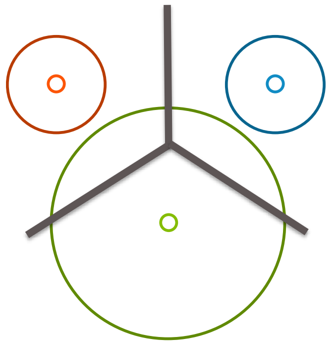
We can change this with weighted Euclidean distance

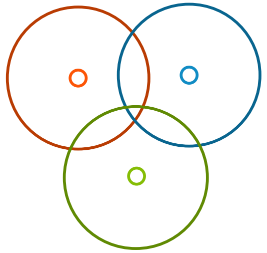
- Still assumes every cluster is the same shape/orientation

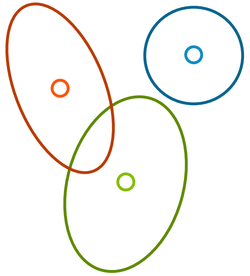


Failure Modes of k-means

If we don't meet the assumption of spherical clusters, we will get unexpected results


disparate cluster sizes


overlapping clusters


different shaped/oriented clusters

Mixture Models

A much more flexible approach is modeling with a **mixture model**

Model each cluster as a different probability distribution and learn their parameters

- E.g. Mixture of Gaussians
- Allows for different cluster shapes and sizes
- Typically learned using Expectation Maximization (EM) algorithm

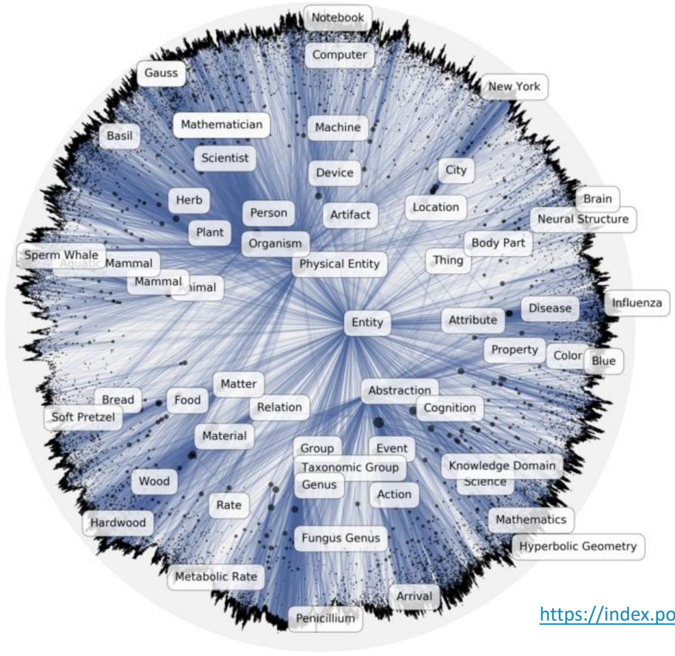
Allows **soft assignments** to clusters

- 54% chance document is about world news, 45% science, 1% conspiracy theory, 0% other

Hierarchical Clustering

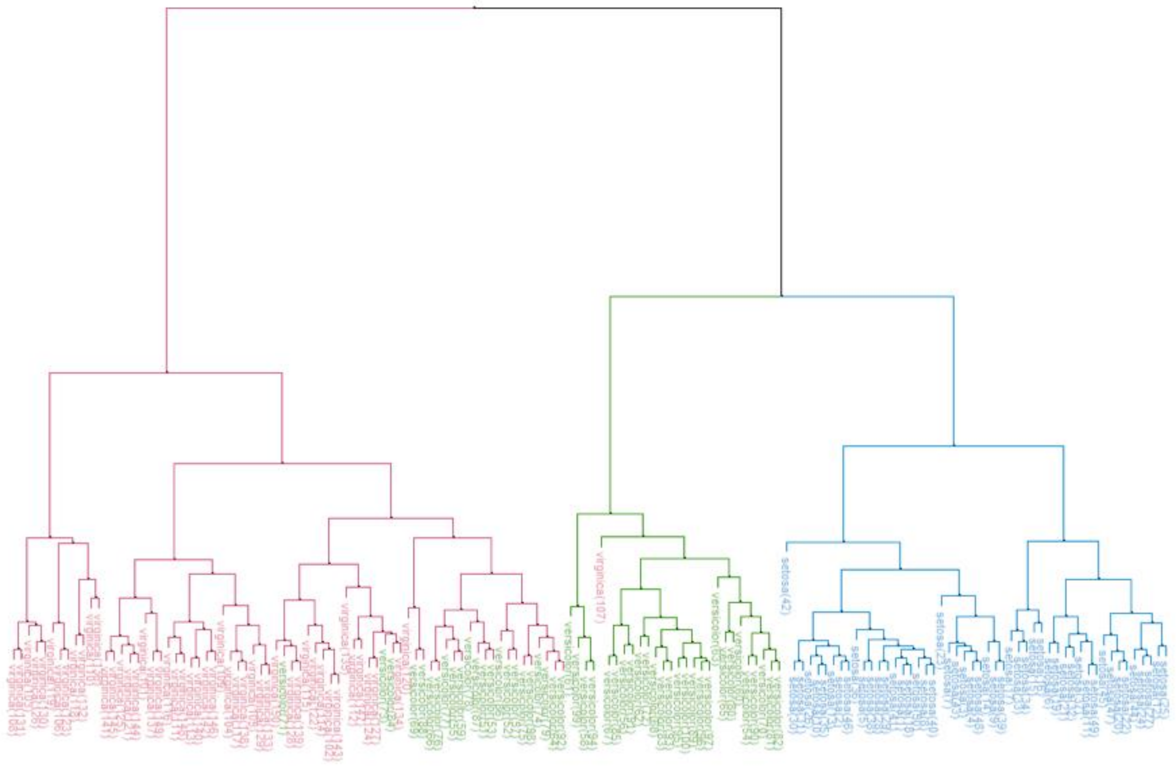
Nouns

Lots of data is hierarchical by nature



https://index.pocketcluster.io/facebookresearch-poincare-embeddings.html

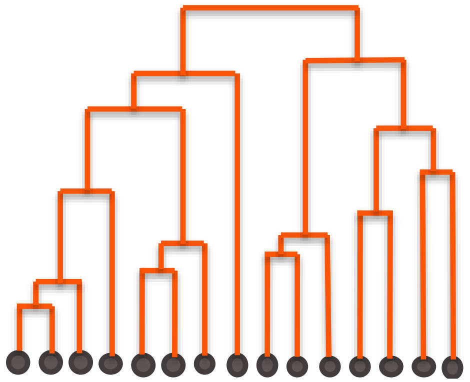
Species



Motivation

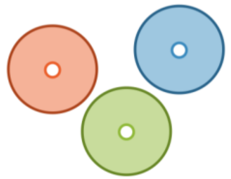
If we try to learn clusters in hierarchies, we can

- Avoid choosing the # of clusters beforehand
- Use **dendrograms** to help visualize different granularities of clusters
- Allow us to use any distance metric
  - K-means requires Euclidean distance
- Can often find more complex shapes than k-means

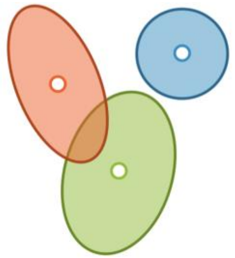


Finding Shapes

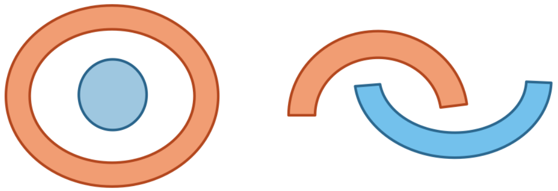
k-means



Mixture Models



Hierarchical Clustering



Types of Algorithms

Divisive, a.k.a. *top-down*

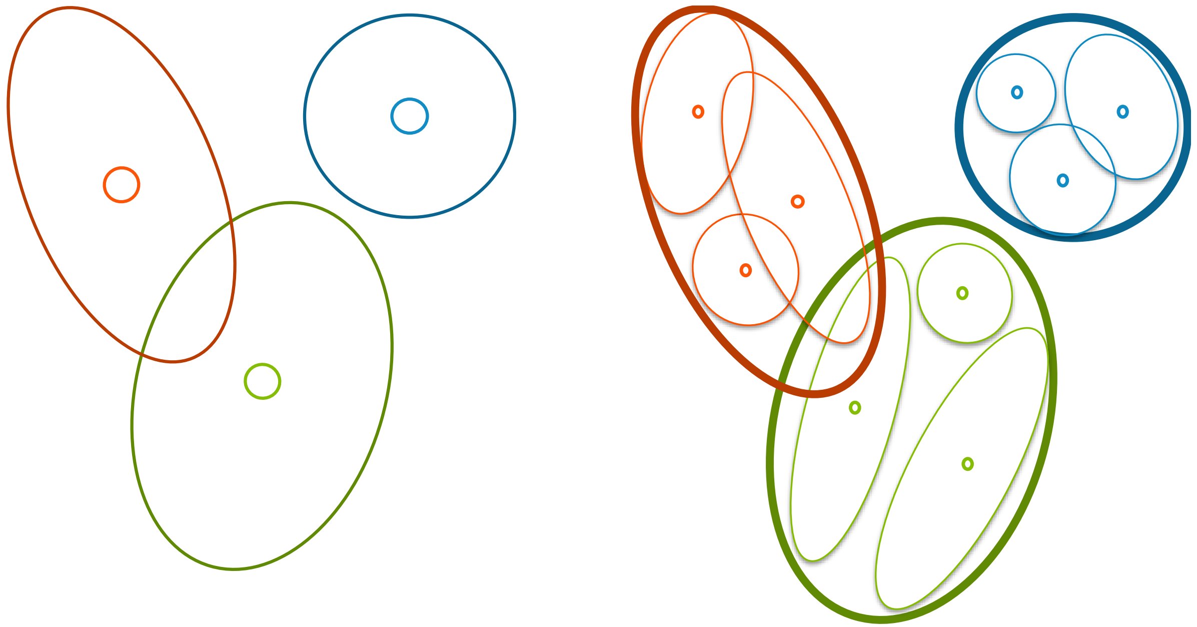
- Start with all the data in one big cluster and then recursively split the data into smaller clusters
  - Example: **recursive k-means**

Agglomerative, a.k.a. *bottom-up*:

- Start with each data point in its own cluster. Merge clusters until all points are in one big cluster.
  - Example: **single linkage**

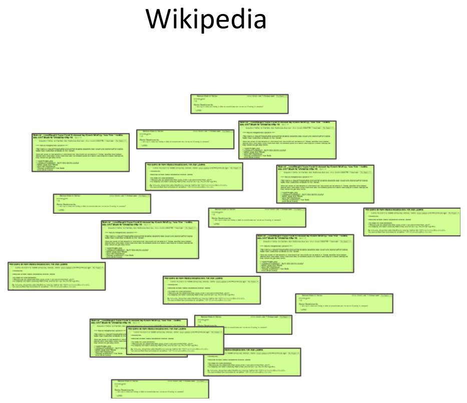
Divisive Clustering

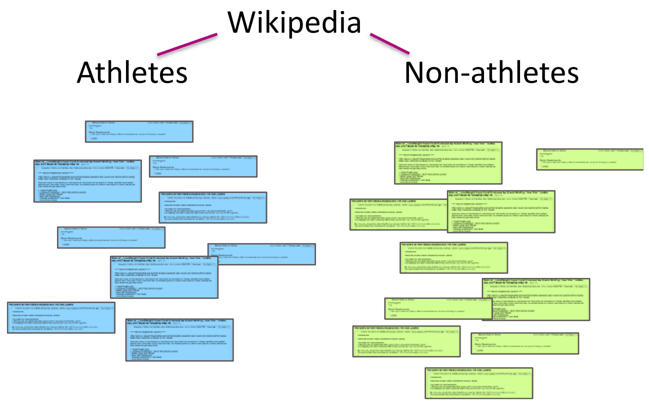
Start with all the data in one cluster, and then run k-means to divide the data into smaller clusters. Repeatedly run k-means on each cluster to make sub-clusters.

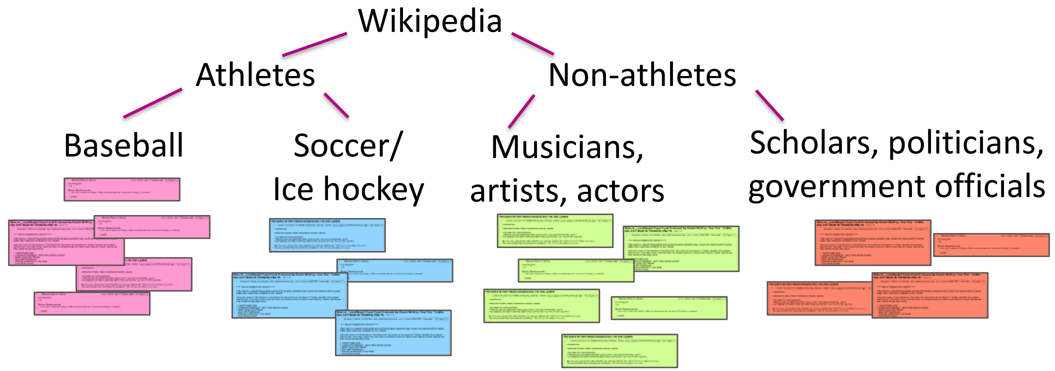


Example

Using Wikipedia






Wikipedia
Athletes
Non-athletes
Baseball
Soccer/ Ice hockey
Musicians, artists, actors
Scholars, politicians, government officials

Choices to Make

For decisive clustering, you need to make the following choices:

- Which algorithm to use
- How many clusters per split
- When to split vs when to stop
  - **Max cluster size**
    Number of points in cluster falls below threshold
  - **Max cluster radius**
    distance to furthest point falls below threshold
  - **Specified # of clusters**
    split until pre-specified # of clusters is reached

Agglomerative Clustering

Algorithm at a glance

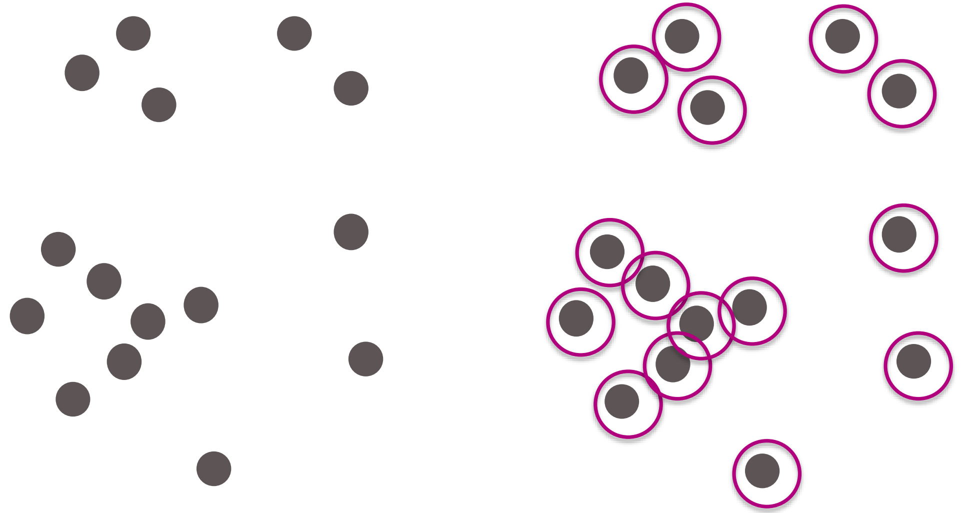
```
1.  Initialize each point in its own cluster
2.  Define a distance metric between clusters

While there is more than one cluster

    3.  Merge the two closest clusters
```

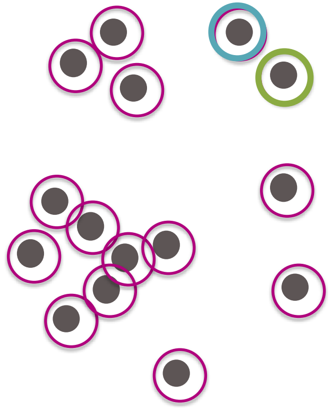
Step 1

1. Initialize each point to be its own cluster



Step 2

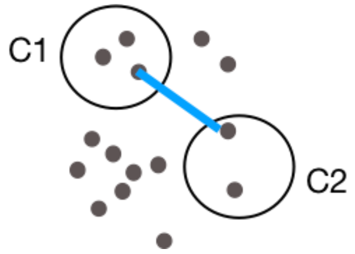
2. Define a distance metric between clusters



Single Linkage

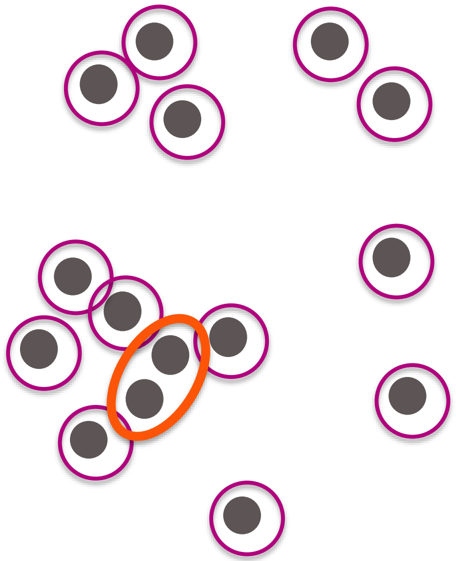
$$distance(C_1, C_2) = \min_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$$

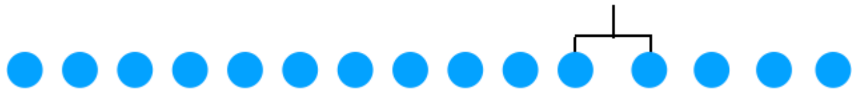
This formula means we are defining the distance between two clusters as the smallest distance between any pair of points between the clusters.



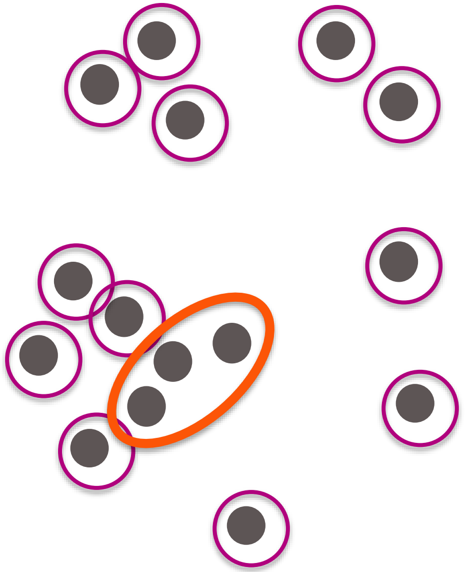
Step 3

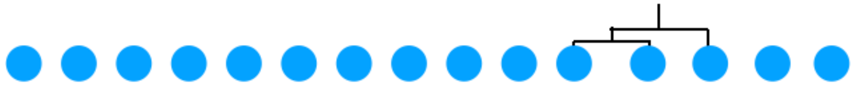
Merge closest pair of clusters





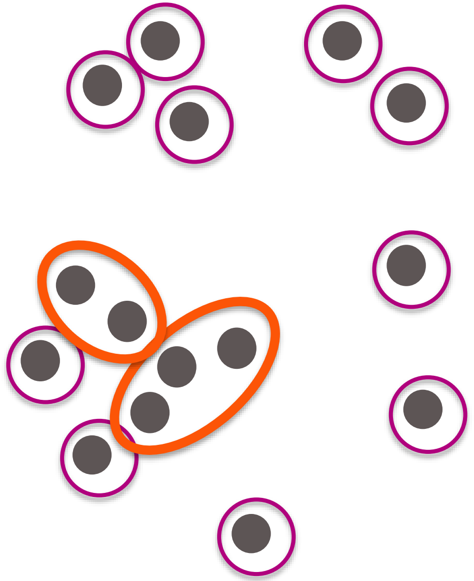
Repeat

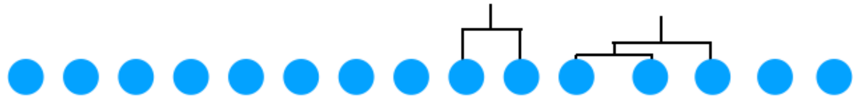




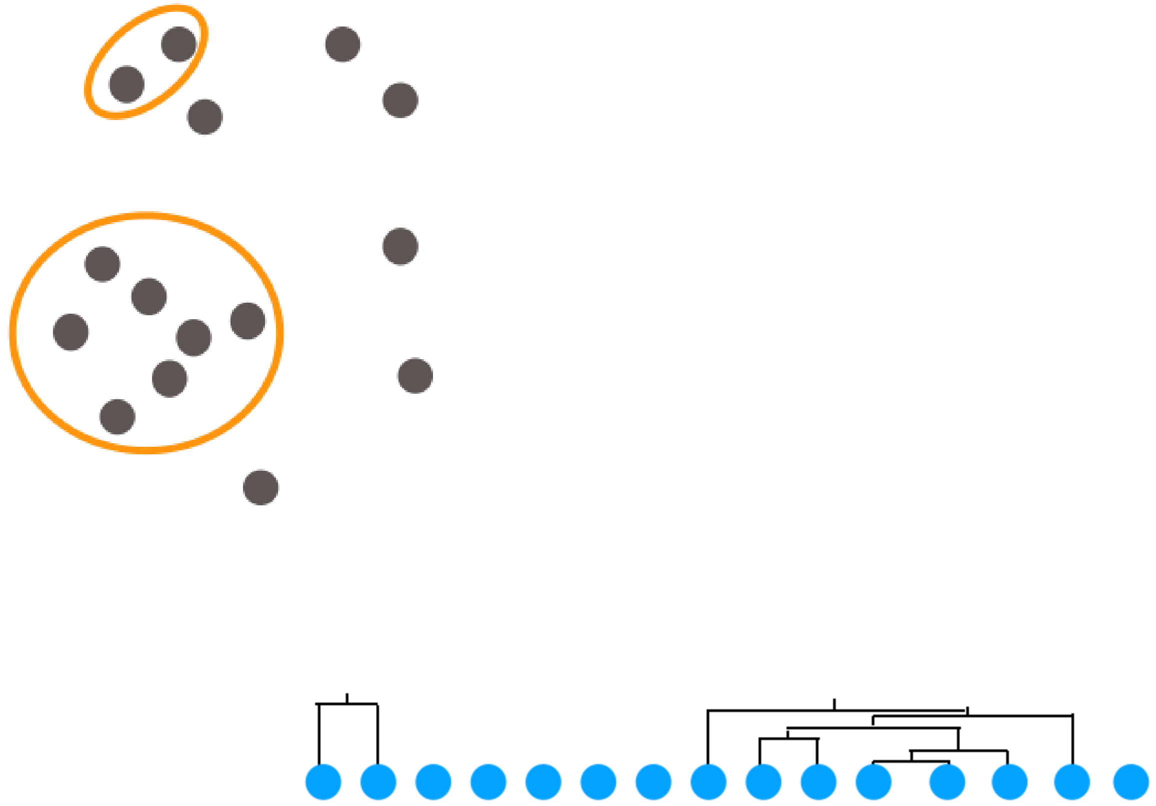
Repeat

Notice that the height of the dendrogram is growing as we group points farther from each other



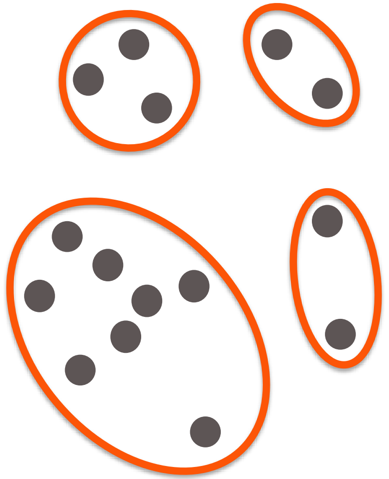


Repeat

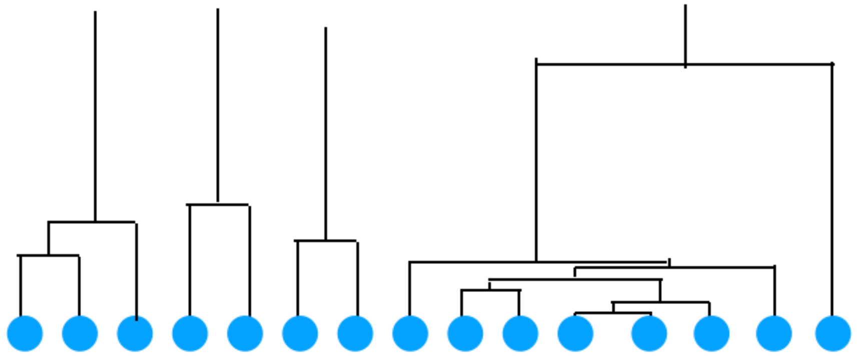


Repeat

Looking at the dendrogram, we can see there is a bit of an outlier!

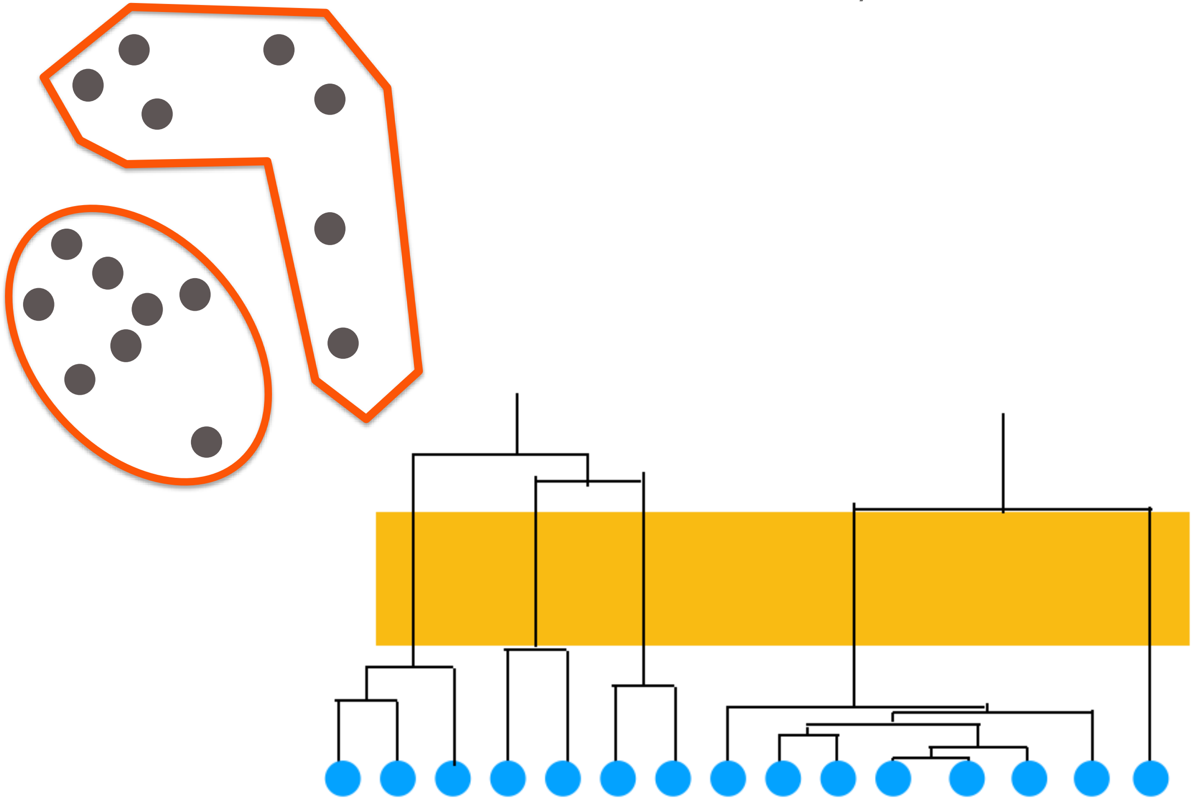


Can tell by seeing a point join a cluster with a really large distance.



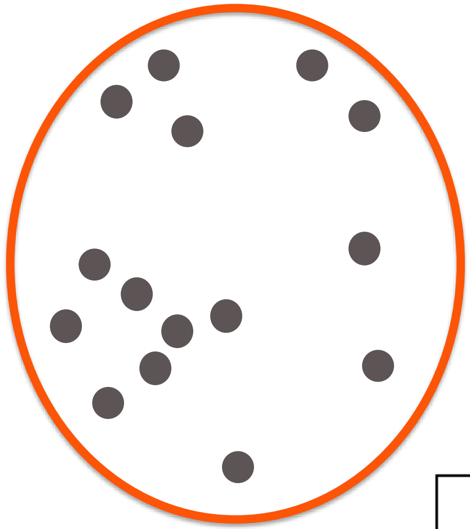
Repeat

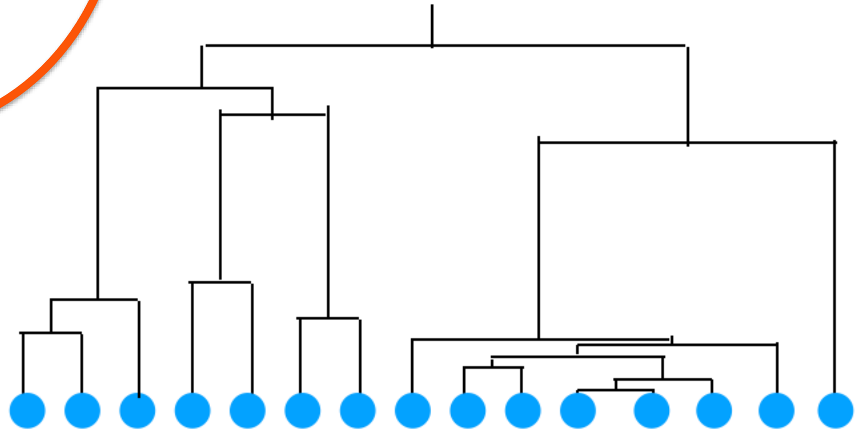
The tall links in the dendrogram show us we are merging clusters that are far away from each other



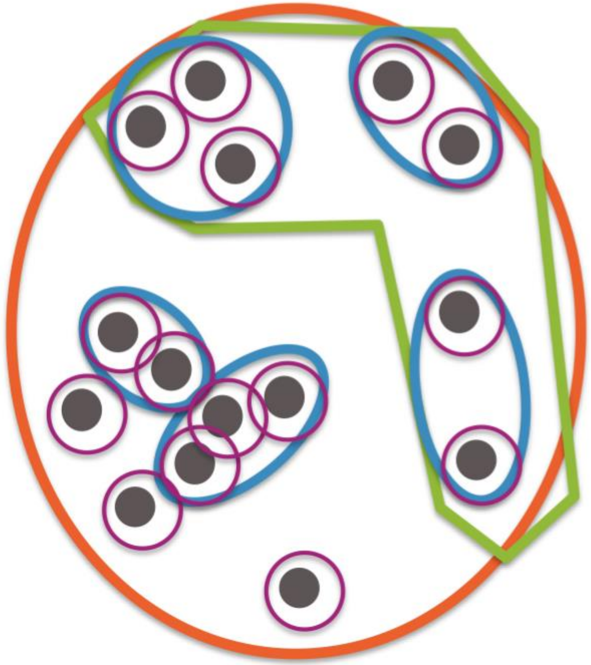
Repeat

Final result after merging all clusters





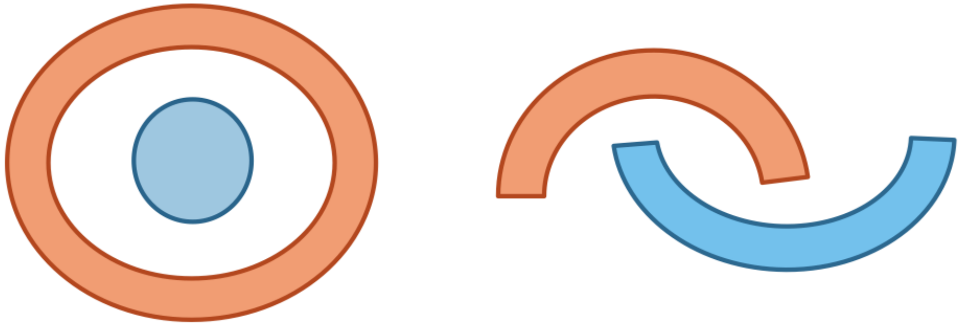
Final Result



Brain Break

Agglomerative Clustering

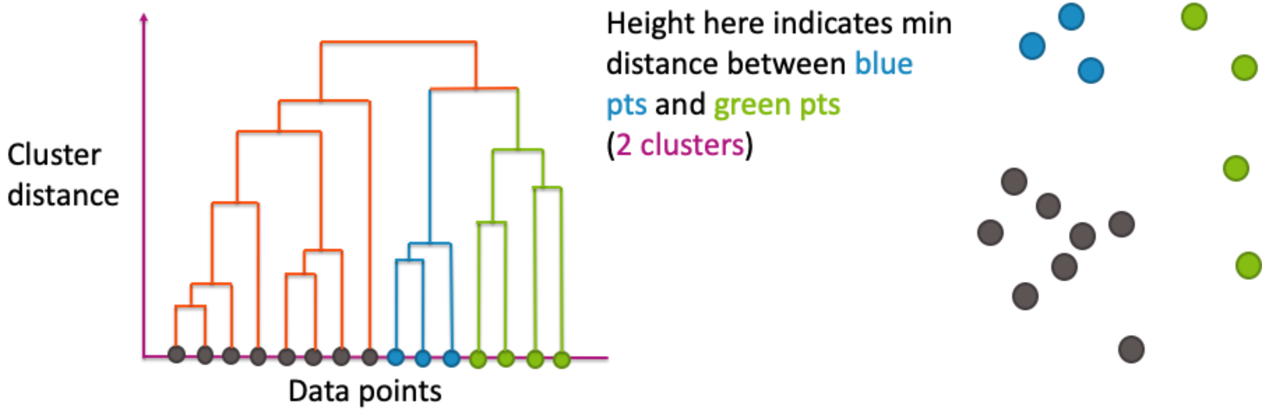
With agglomerative clustering, we are now very able to learn weirder clusterings like



Dendrogram

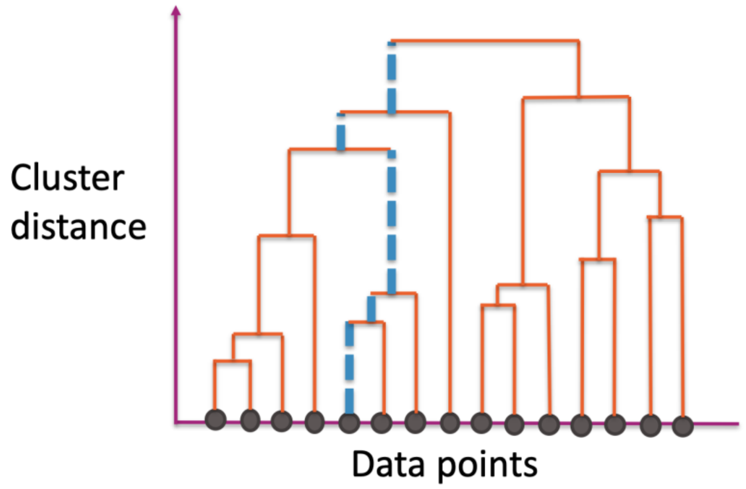
x-axis shows the datapoints (arranged in a very particular order)

y-axis shows distance between pairs of clusters


Cluster distance
Data points
Height here indicates min distance between blue pts and green pts (2 clusters)

Dendrogram

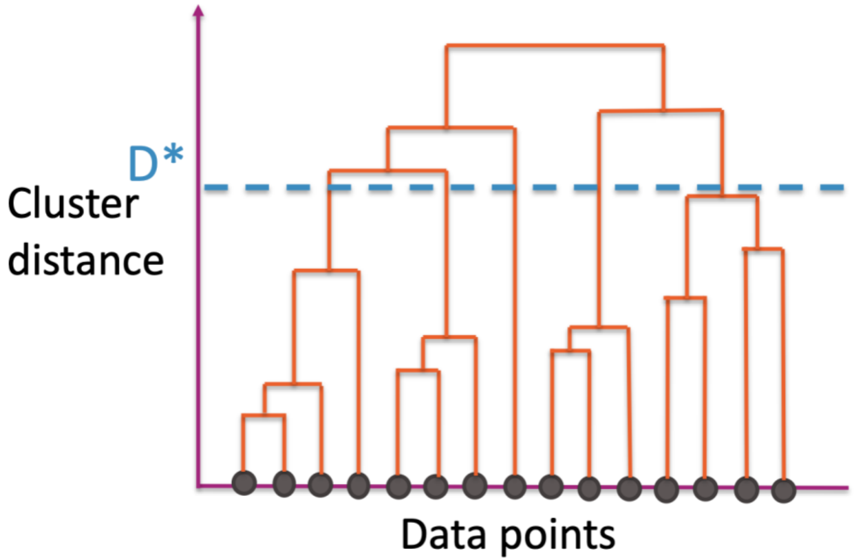
The path shows you all clusters that a single point belongs and the order in which its clusters merged



Cut Dendrogram

Choose a distance D^* to “cut” the dendrogram

- Use the largest clusters with distance $< D^*$
- Usually ignore the idea of the nested clusters after cutting



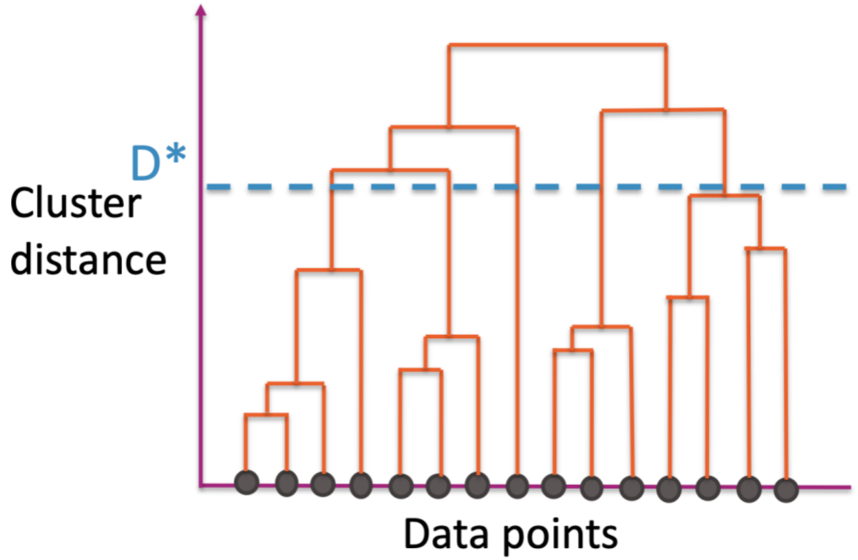
Poll Everywhere

Think

1 min

pollev.com/cs416

How many clusters would be have if we use this threshold?



1:00

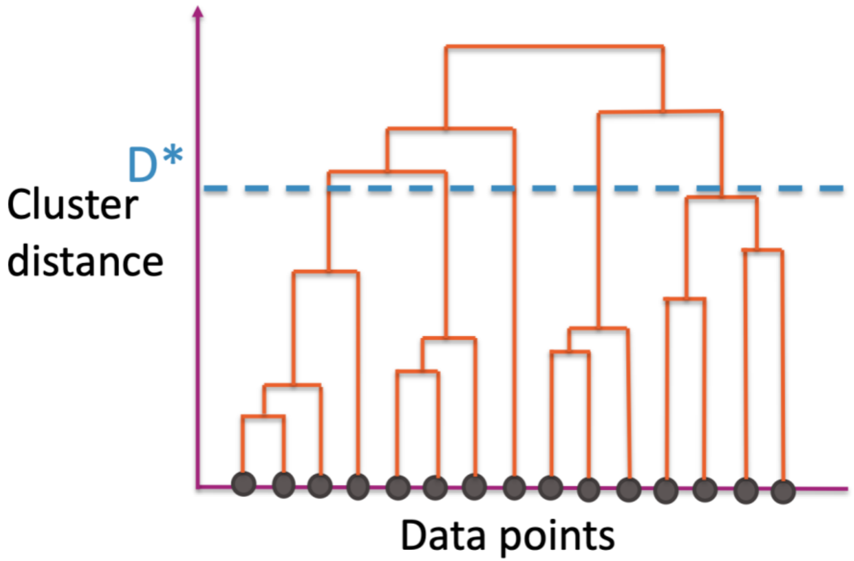
Poll Everywhere

Think

2 min

pollev.com/cs416

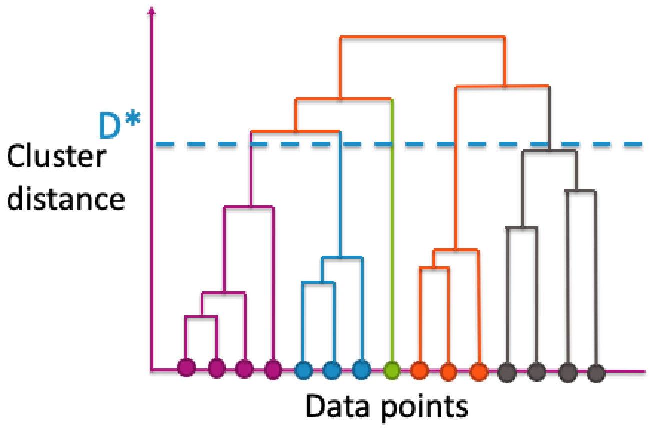
How many clusters would we have if we use this threshold?

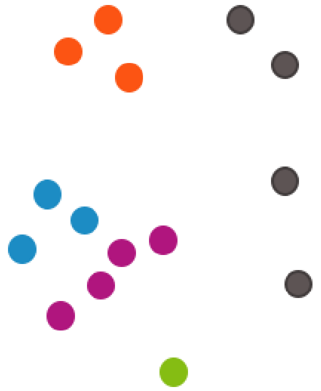


2:00

Cut Dendrogram

Every branch that crosses D^* becomes its own cluster





Choices to Make

For agglomerative clustering, you need to make the following choices:

- Distance metric $d(x_i, x_j)$
- Linkage function
  - Single Linkage:

$$\min_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$$

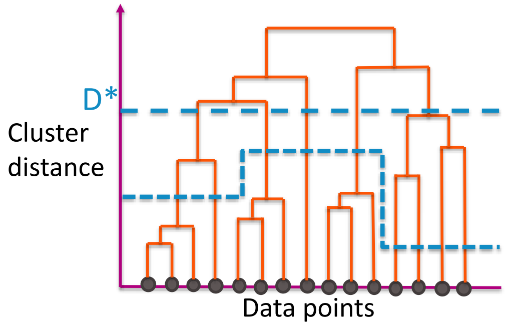
  - Complete Linkage:

$$\max_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$$

  - Centroid Linkage

$$d(\mu_1, \mu_2)$$

  - Others
- Where and how to cut dendrogram



Practical Notes

For visualization, generally a smaller # of clusters is better

For tasks like outlier detection, cut based on:

- Distance threshold
- Or some other metric that tries to measure how big the distance increased after a merge

No matter what metric or what threshold you use, no method is “incorrect”. Some are just more useful than others.

Computational Cost

Computing all pairs of distances is pretty expensive!

- A simple implementation takes $\mathcal{O}(n^2 \log(n))$

Can be much implemented more cleverly by taking advantage of the **triangle inequality**

- “Any side of a triangle must be less than the sum of its sides”

Best known algorithm is $\mathcal{O}(n^2)$

Missing Data

Missing Data

Data in the real-world is rarely clean or as nicely structured as data provided to you on a HW in class. You saw this in HW6!

One common way data can be messy (but not the only one!) is to have missing values.

- This usually takes the form of a NaN or some special value (e.g., an empty string or -1).

Just like how there isn't ever one right answer for modeling, how you deal with missing data will not have one right answer either!

- Usually depends on domain experience!

Missing Data

Missing data can happen at either

- Training time
- Prediction time (e.g., testing or after deploying)

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	?	high	risky
poor	5 yrs	low	safe
fair	?	high	safe

Loan application may be 3 or 5 years

Strategy 1: Skipping

The simplest strategy is just completely ignore missing values so you don’t have to deal with them.

This can take the form of

- Dropping rows with missing values
- Dropping features (columns) with missing values

Which to drop depends on how much data is missing / how important those entities are:

- If only one training row has a missing value, dropping it doesn’t seem to bad
- If only a few features (out of many) have missing values, maybe just drop those!

Strategy 1: Skipping (Pros/Cons)

Pros

- Very easy to understand/explain
- Can be applied to any model

Cons

- Might be removing useful information
- When is it better to remove examples vs. features?
- Doesn't help if data is missing at prediction time

Strategy 2: Sentinel Values

Idea: Replace missing data with some default value

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	UNK	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	UNK	high	risky
poor	5 yrs	low	safe
fair	UNK	high	safe

Strategy 2: Sentinel Values (Pros/Cons)

Pros

- Fairly simple to describe
- Efficient fix and works at prediction time as well
- Works well for categorical features (treats missingness as an important value in its own).

Cons

- Only works well for features that are already categorical. Numeric features have no clear sentinel value.

Strategy 3: Imputation

Use some heuristic (or learning) to fill in missing data with better guesses for their values.

A simple approach:

- Categorical features: Use most popular (mode) of non-missing values
- Numeric features: Use mean or median of non-missing values

Complex approach:

- Use a learning algorithm to learn relationships between the other features and the features with missing values. Fill in missing values with some learned model.
  - Many algorithms use a back-and-forth processes like EM used in clustering!

Strategy 3: Imputation (Pros/Cons)

Pros

- Usually easy to understand and implement (if using simple approach)
- Can be applied to any model
- Can be used at prediction time: Use same imputation rules

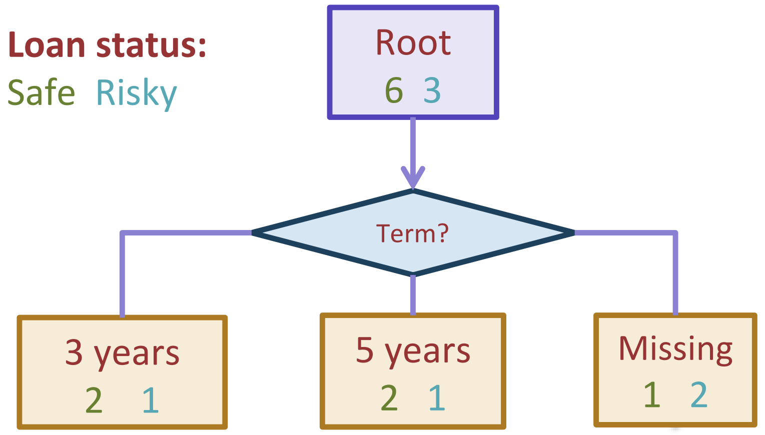
Cons

- May result in systematic errors (ask: why are certain values missing)
- Missing values could signal of their own and this removes them (e.g., credit-card fraud)

Strategy 4: Modify Algorithm

Use a new type of model that is robust to the presence of missing values.

For example, implement a new decision tree from scratch that can handle missing values (e.g., makes a new branch if a value is missing)



Strategy 4: Modify Algorithm (Pros/Cons)

Pros

- Very similar to sentinel values in terms of pros but can also handle numeric features
- Generally can have more accurate predictions

Cons

- Requires implementing a new type of model
  - Maybe easy for decision trees, but other types???

Miss Data - Recap

There are a lot of approaches to handling missing values. Note that missing values is a common source of messy data, is just one out of an infinite number of ways your data could be difficult to work with.

There will never be “one right strategy”, how you handle missing data is a modeling choice just like every other modeling you choice you make.

Concept Inventory

This week we want to practice recalling vocabulary. Spend 10 minutes trying to write down all the terms for concepts we have learned in this class and try to bucket them into the following categories.

Regression

Classification

Document Retrieval

Misc – For things that fit in multiple places or none of the above

You don't need to define/explain the terms for this exercise, but you should know what they are!

Try to do this for at least 5 minutes from recall before looking at your notes!