CSE/STAT 416

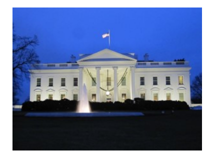
Hierarchical Clustering

Hunter Schafer
University of Washington
July 31, 2019

Clustering




SPORTS


WORLD NEWS

Define Clusters

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

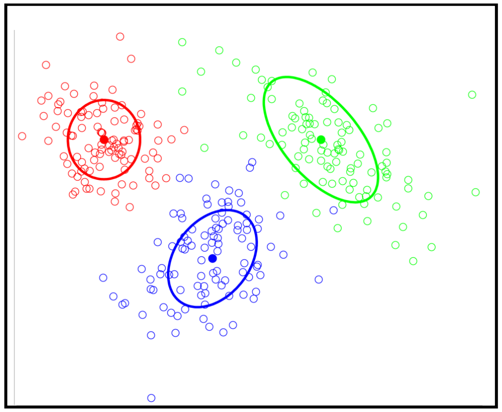
- The location of its center (**centroid**)
- Shape and size of its **spread**

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

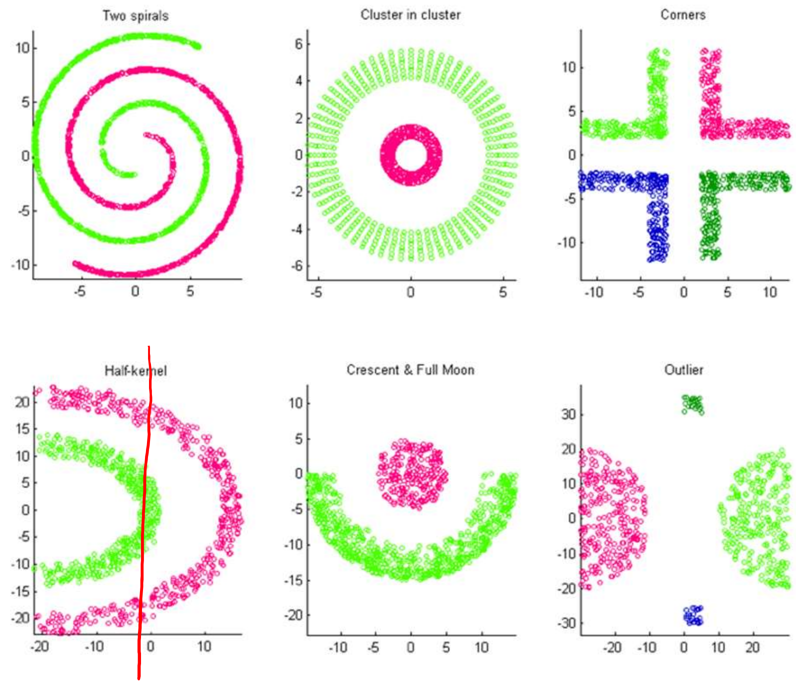
- Based on distance of assigned examples to each cluster



Not Always Easy

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

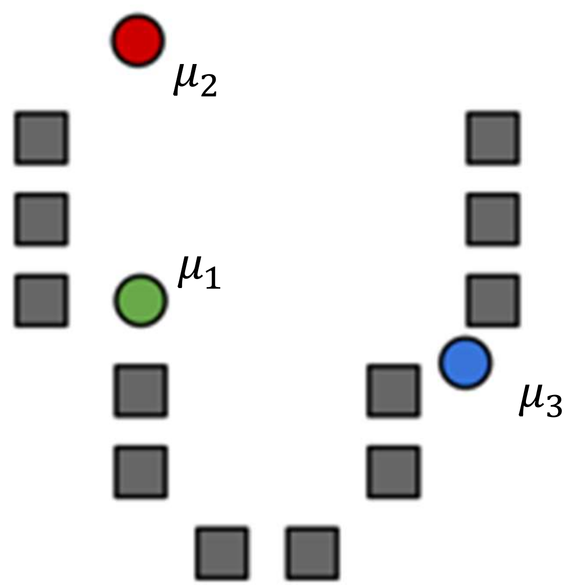
- Distance does not determine clusters



Step 0

Start by choosing the initial cluster centroids

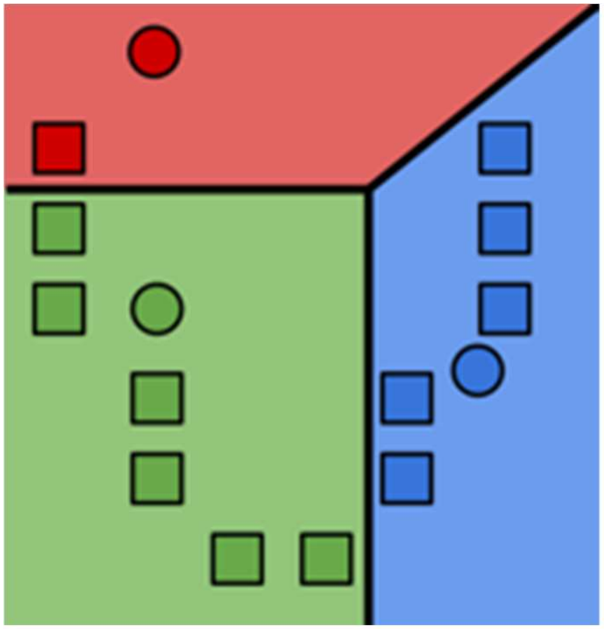
- A common default choice is to choose centroids at random
- Will see later that there are smarter ways of initializing



Step 1

Assign each example to its closest cluster centroid

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

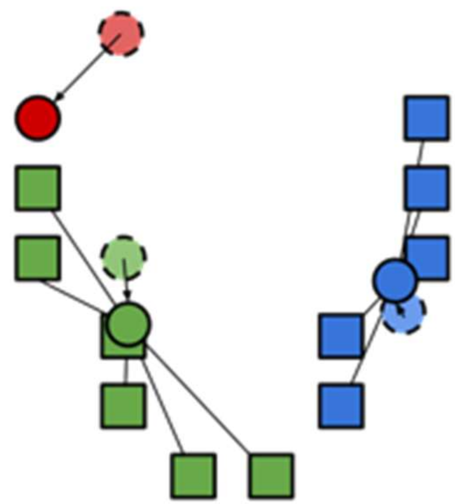


Step 2

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

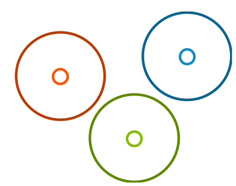
1. Choose first cluster μ_1 from the data uniformly at random
2. For the current set of centroids (starting with just μ_1), compute the distance between each datapoint and its closest centroid
3. Choose a new centroid from the remaining data points with probability of x_i being chosen proportional to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

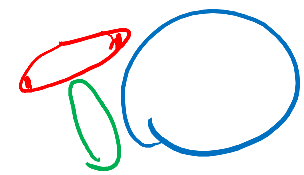
Problems with k-means

In real life, cluster assignments are not always clear cut

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

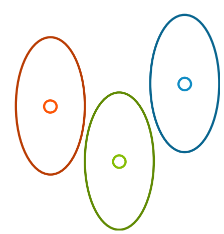
Because we minimize Euclidean distance, k-means assumes all the clusters are spherical





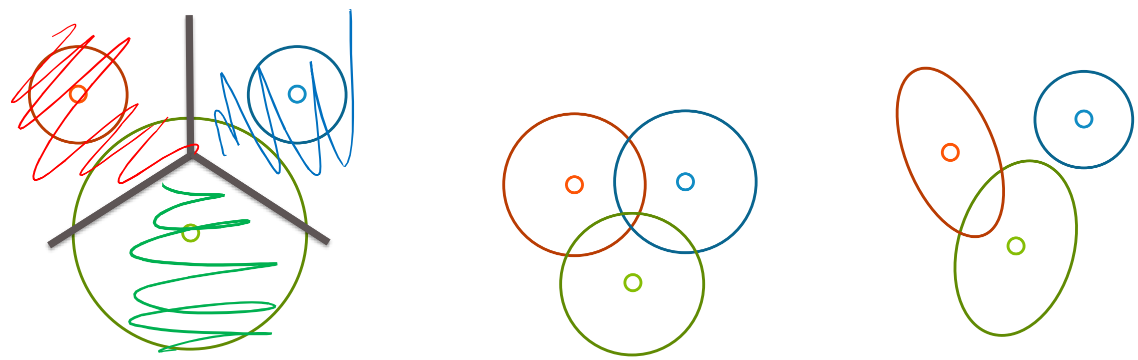
We can change this with weighted Euclidean distance

- 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

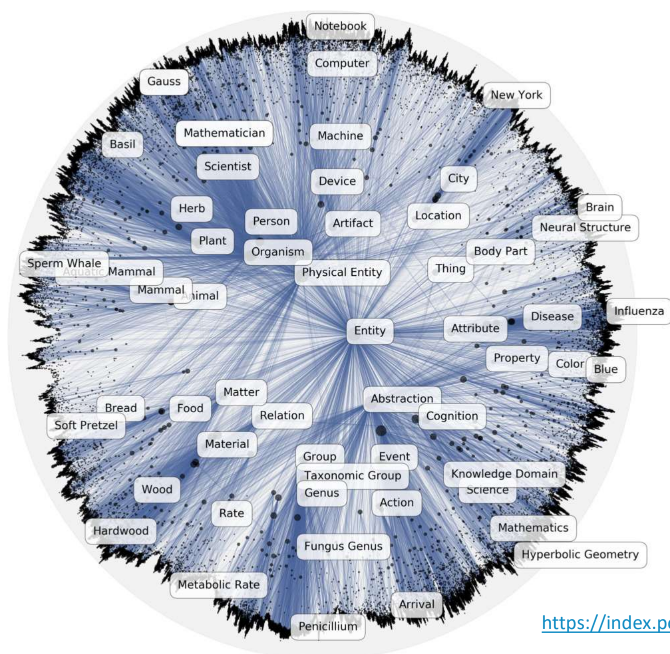
Allows **soft assignments** to clusters

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

Hierarchical Clustering

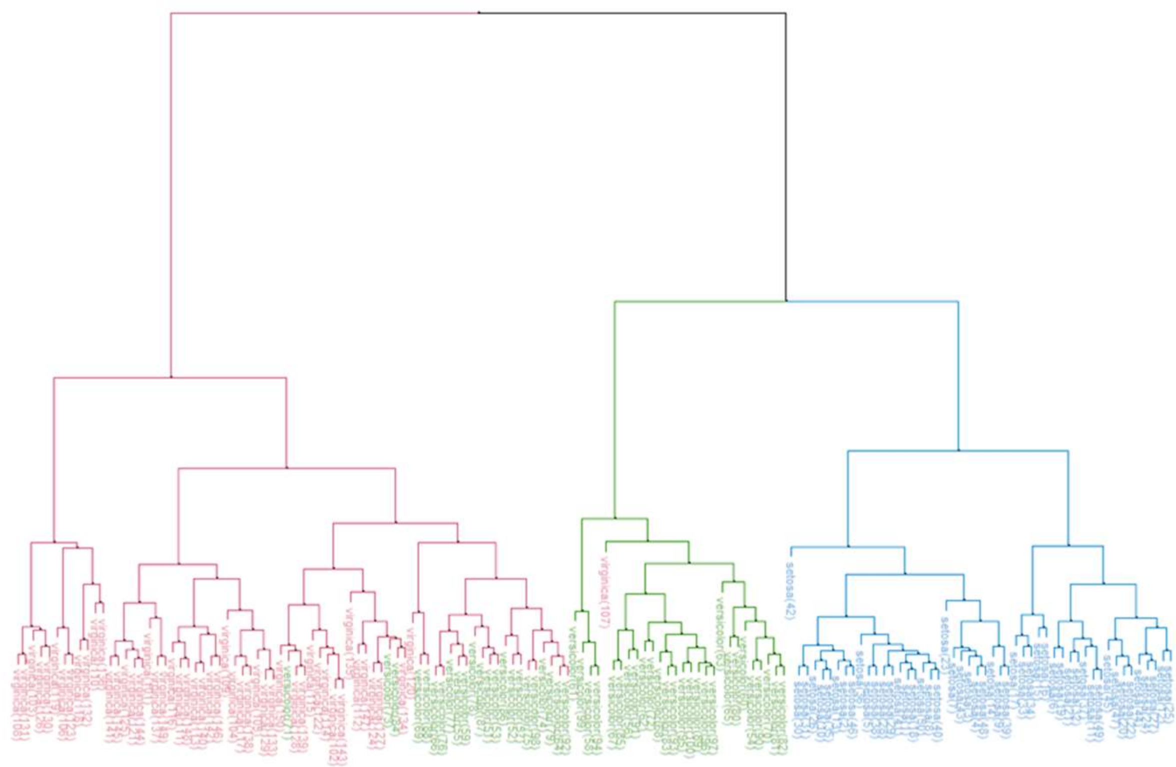
Nouns

Lots of data is hierarchical by nature



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

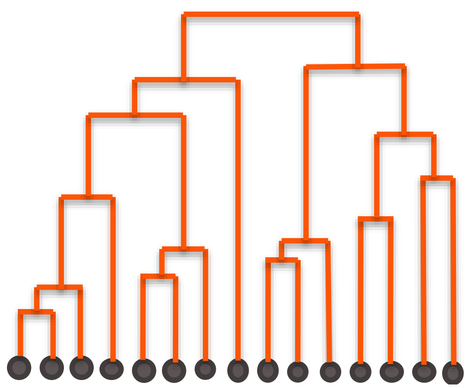
Species



Motivation

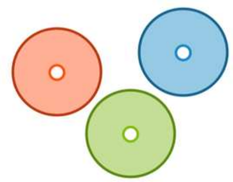
If we try to learn clusters in hierarchies, we can

- 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

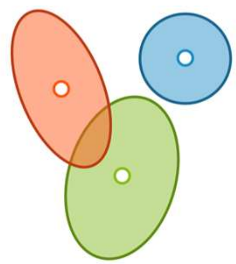


Finding Shapes

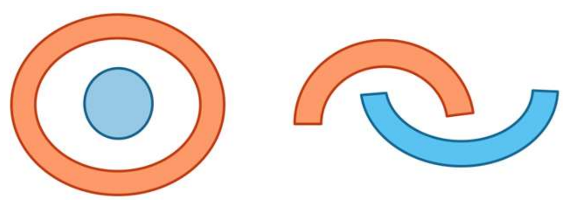
k-means



Mixture Models



Hierarchical Clustering



Types of Algorithms

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

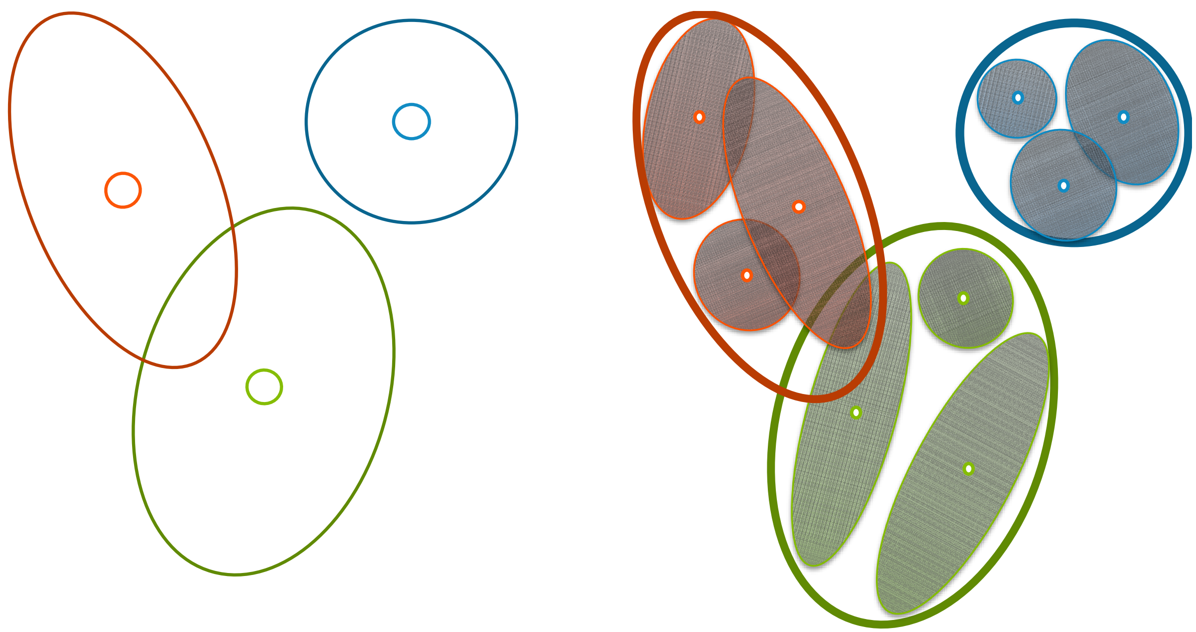
- Start with all the data in one big cluster and then recursively split the data into smaller clusters
  - Example: **recursive k-means**

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**

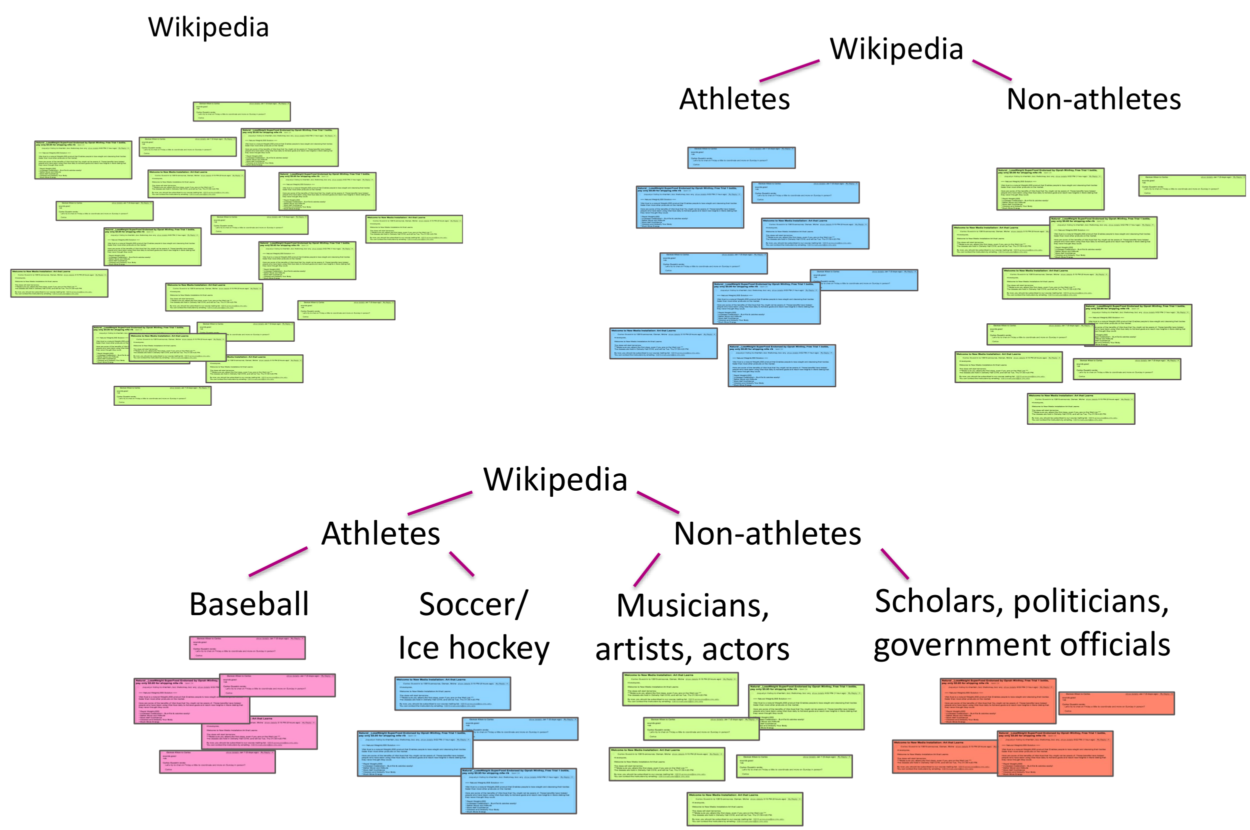
Divisive Clustering

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



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

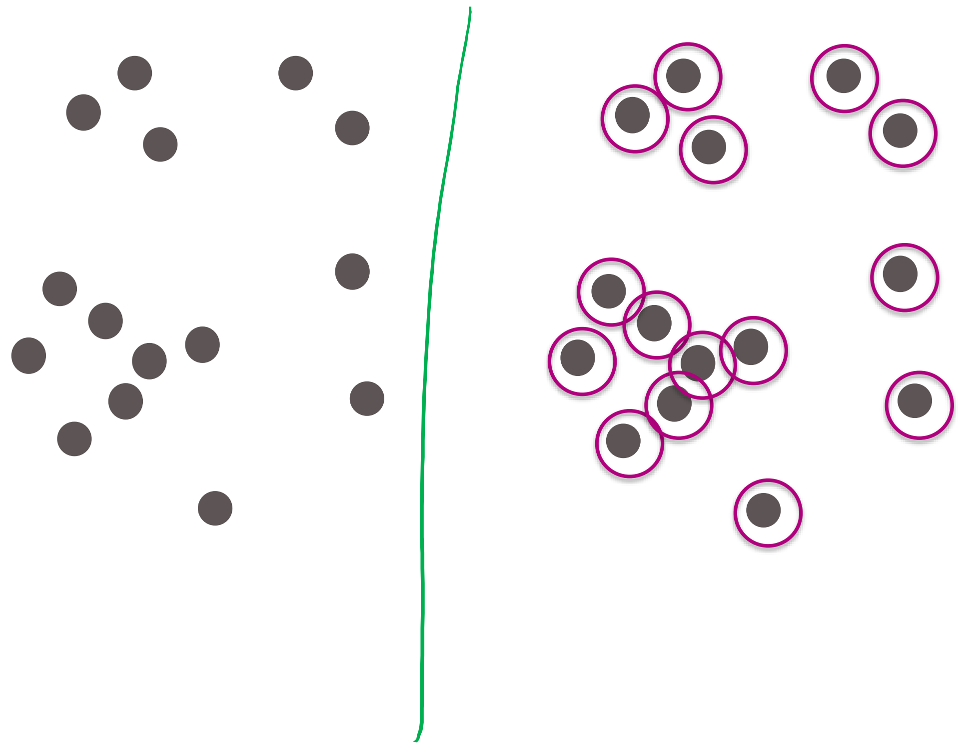
1. Initialize each point in its own cluster
2. Define a distance metric between clusters

While there is more than one cluster

3. Merge the two closest clusters

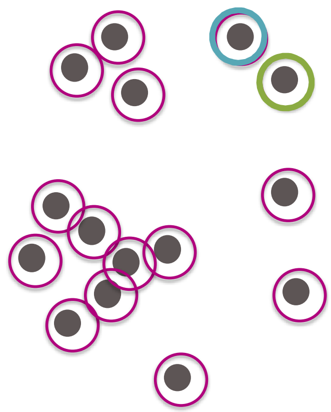
Step 1

1. Initialize each point to be its own cluster



Step 2

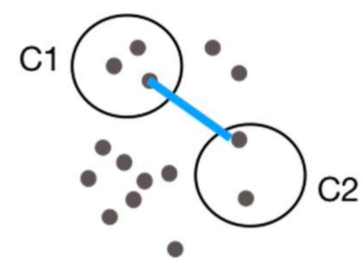
2. Define a distance metric between clusters



Single Linkage

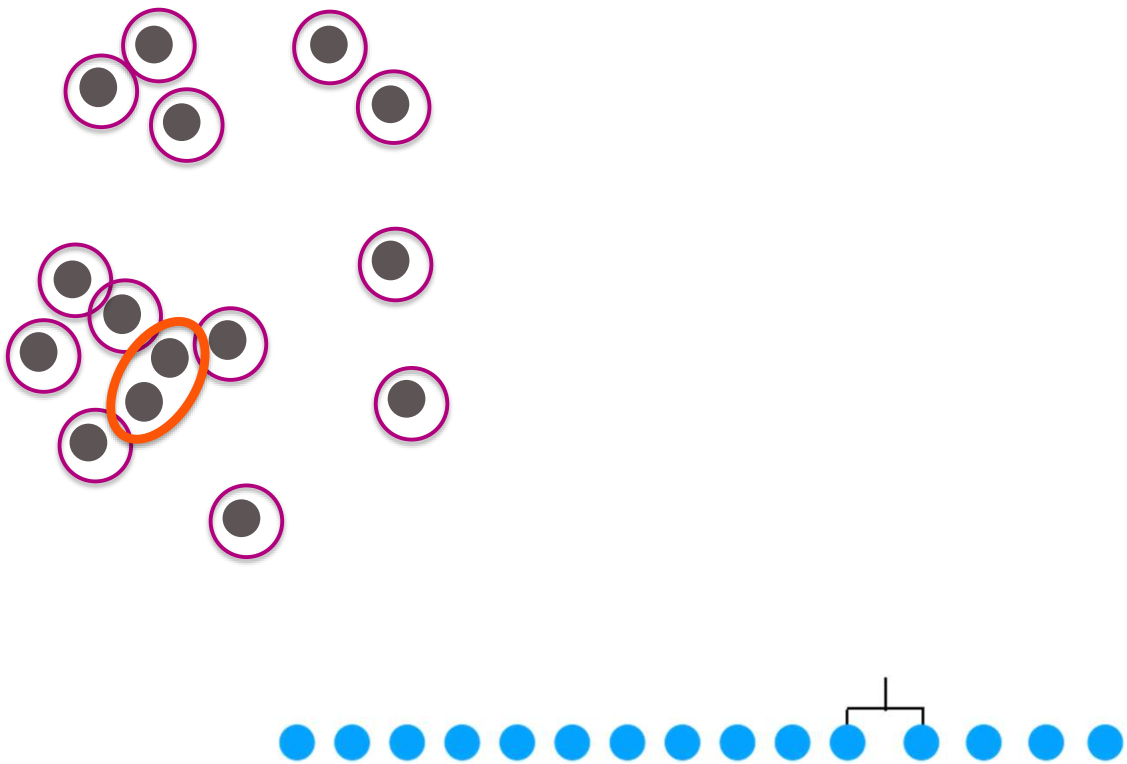
$$distance(C_1, C_2) = \min_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$$

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

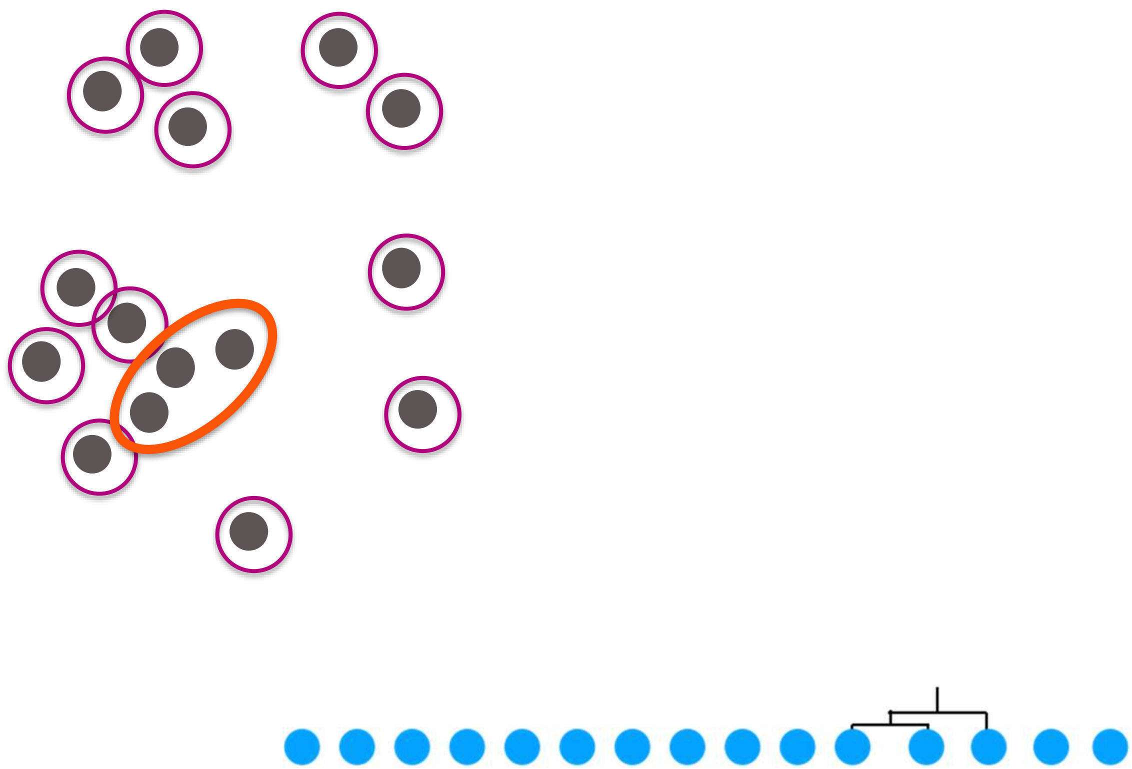


Step 3

Merge closest pair of clusters

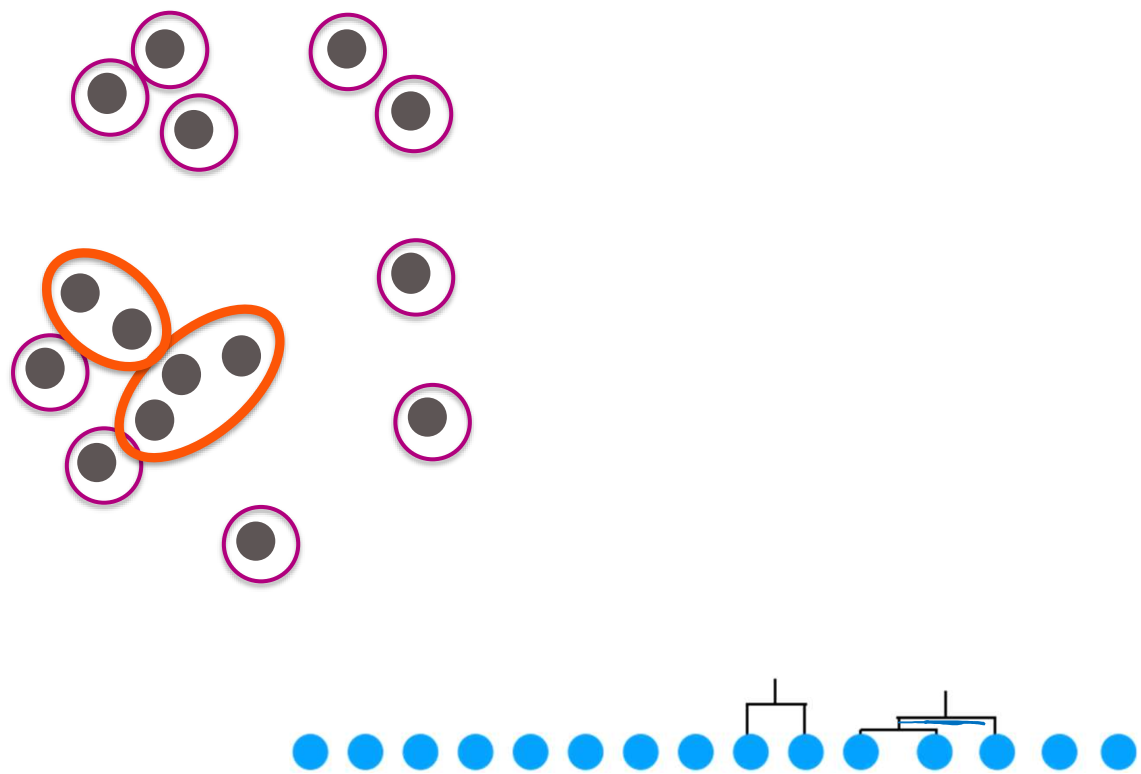


Repeat

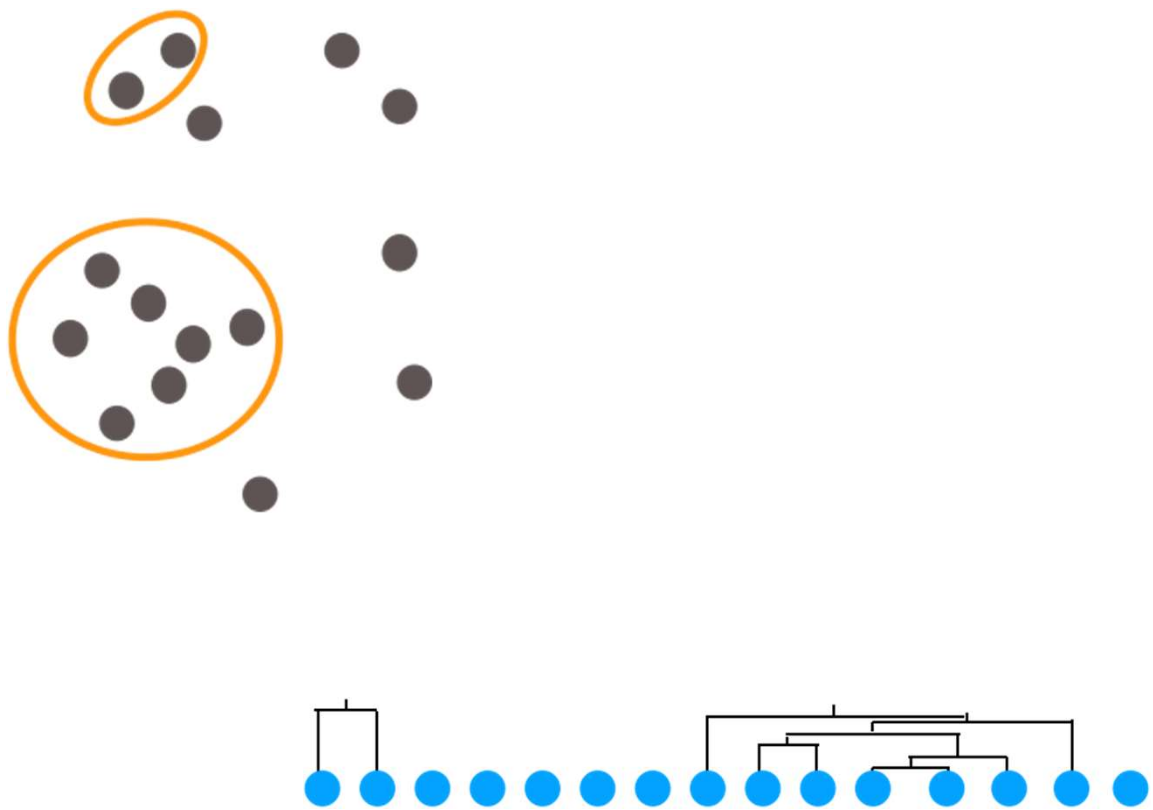


Repeat

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

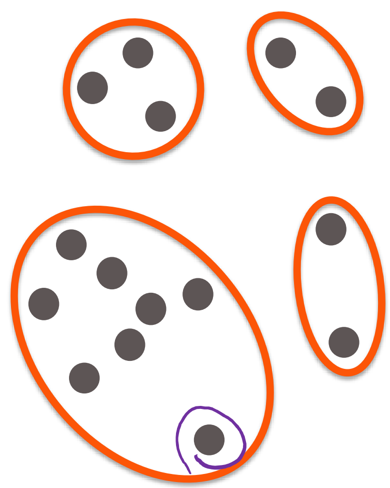


Repeat

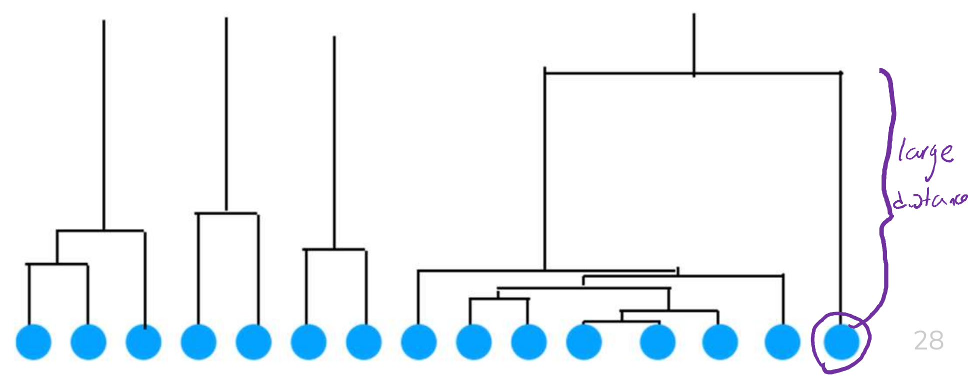


Repeat

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

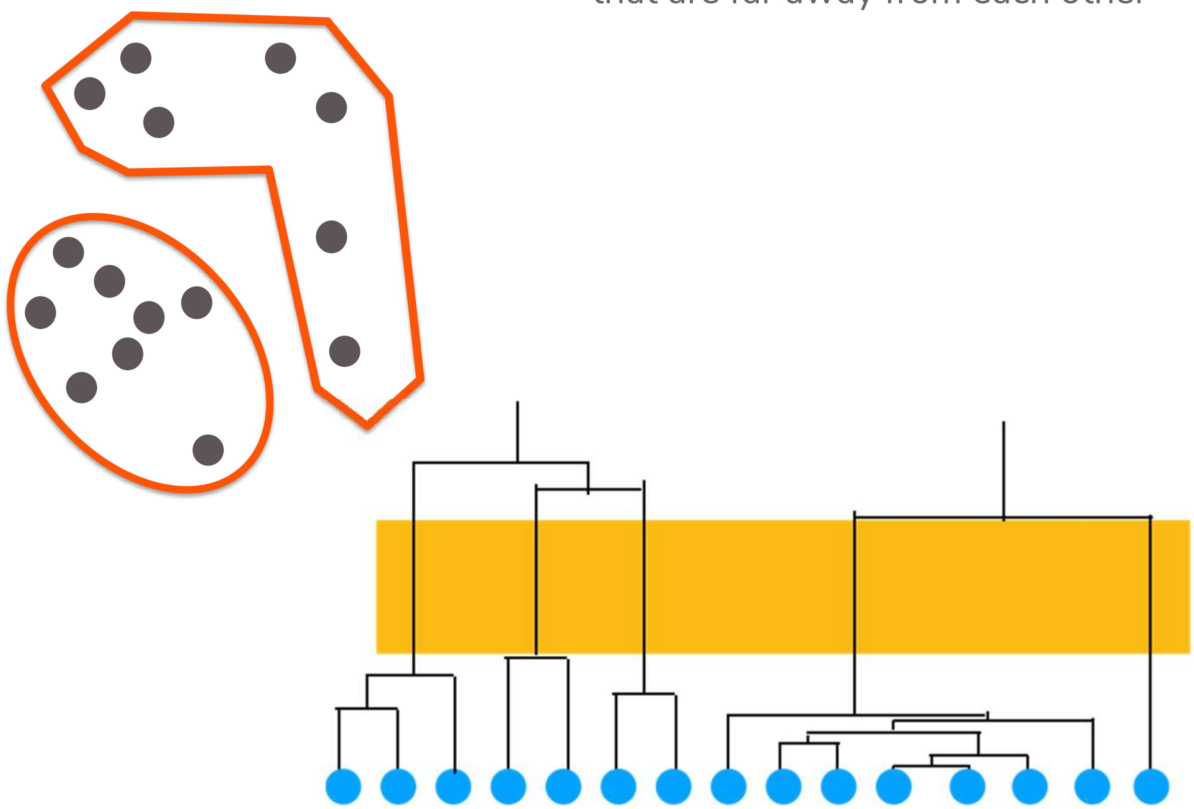


Can tell by seeing a point join a cluster with a really large distance.



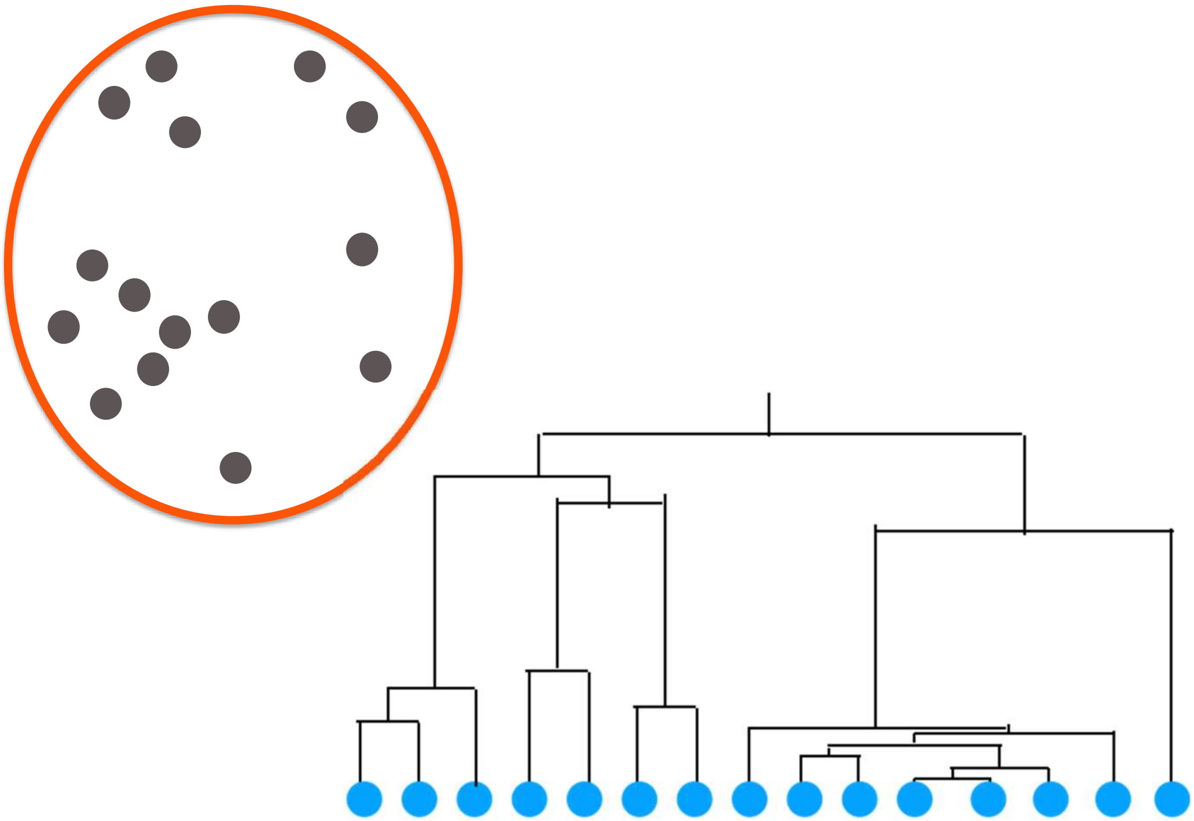
Repeat

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

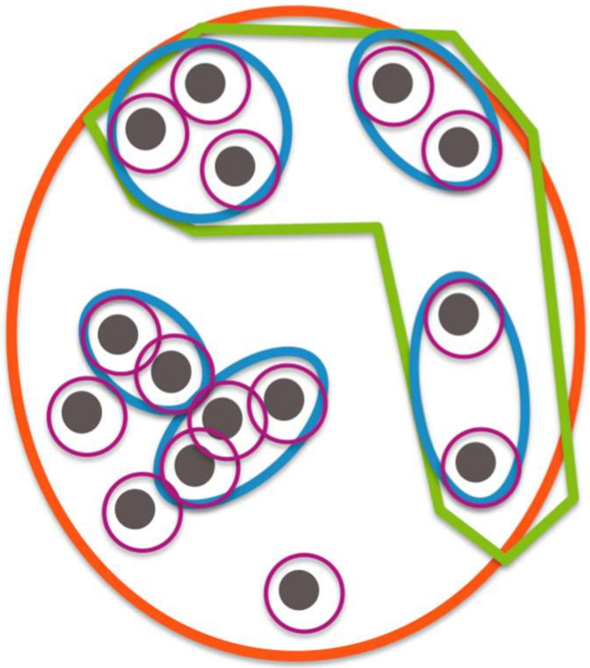


Repeat

Final result after merging all clusters

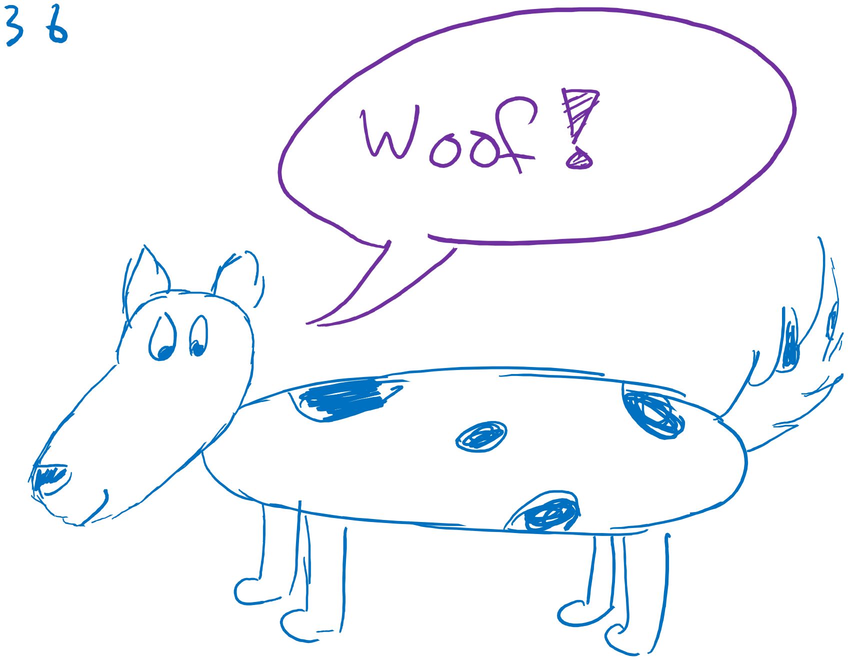


Final Result



Brain Break

10:36



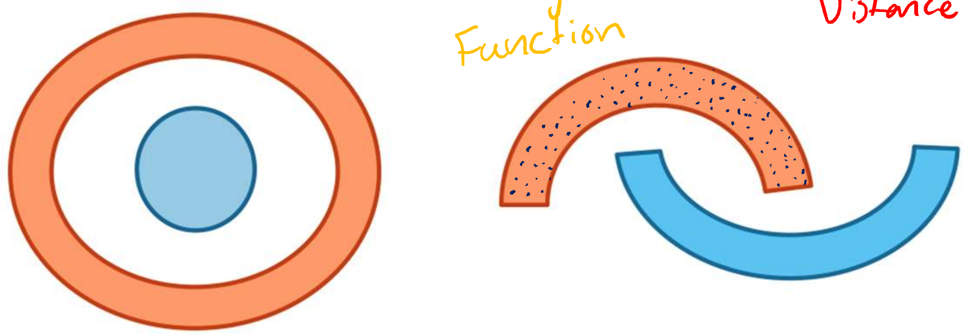
Agglomerative Clustering

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

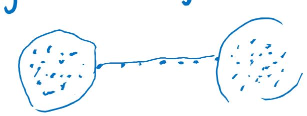
Single Linkage: $$\min_{x_i \in C_1, x_j \in C_2} \boxed{d(x_i, x_j)}$$

Linkage Function

Distance Metric



Single linkage can group long chains



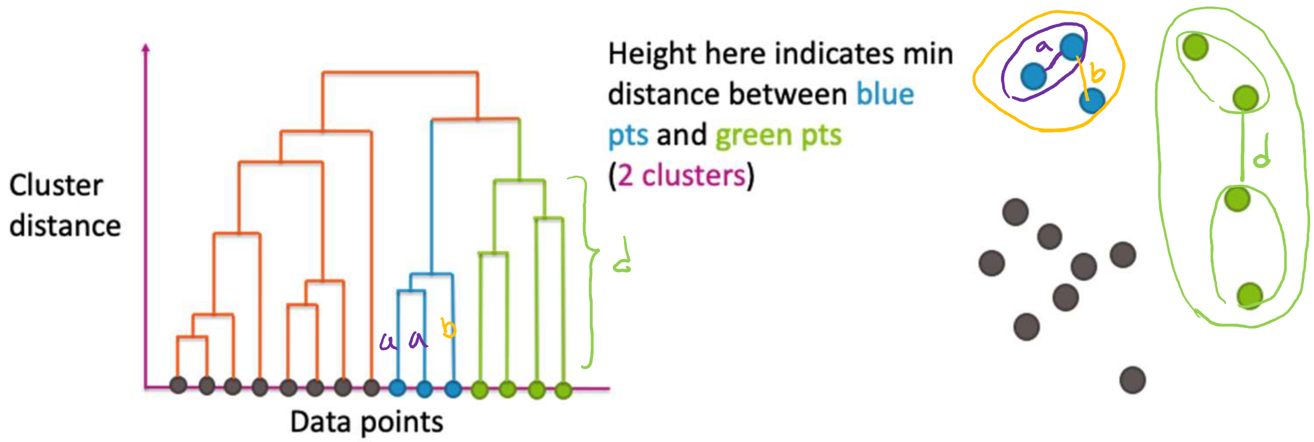
Okay in some contexts

Less desirable in others

Dendrogram

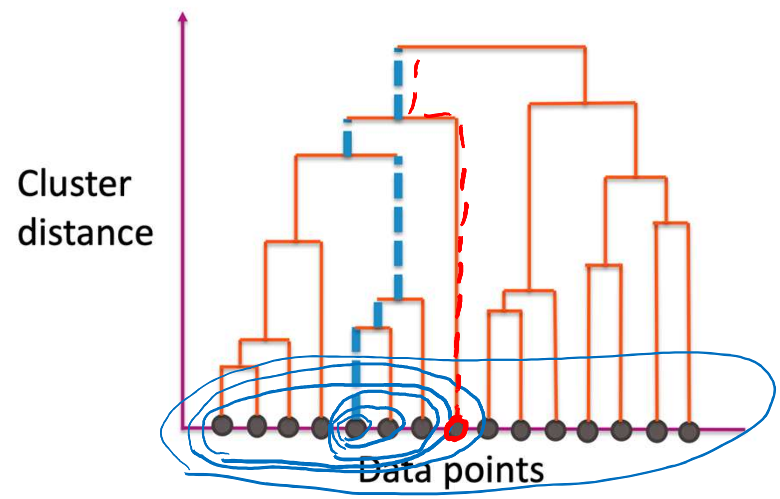
x-axis shows the datapoints (arranged in a very particular order)

y-axis shows distance between pairs of clusters



Dendrogram

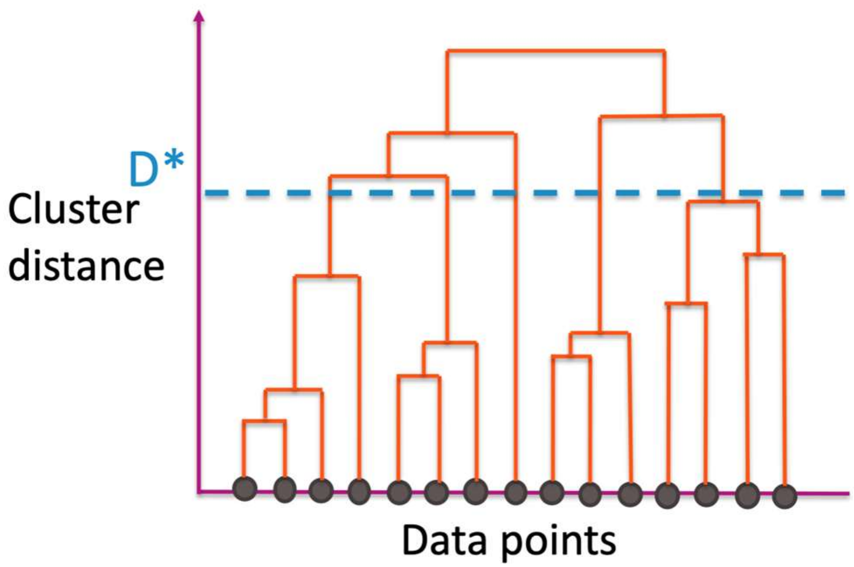
The path shows you all clusters that a single point belongs and the order in which its clusters merged



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



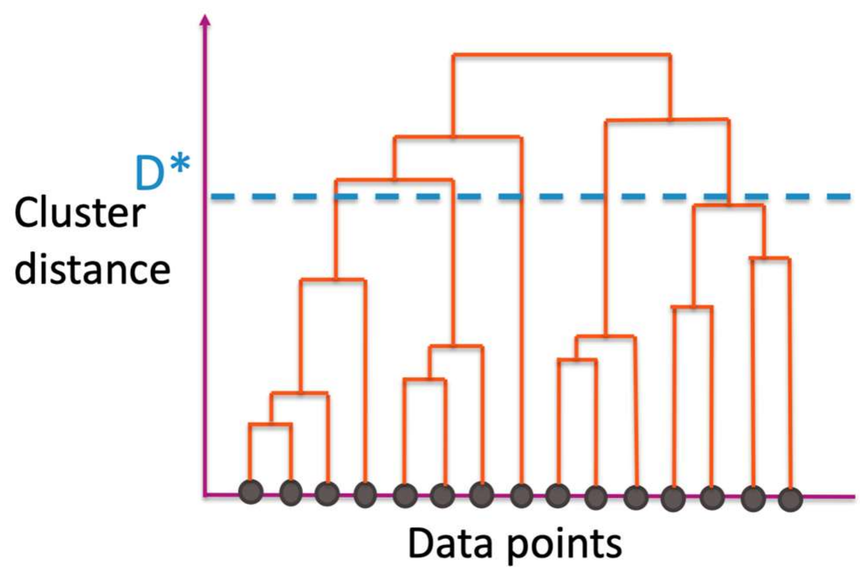
Poll Everywhere

Think

1 min

pollev.com/cse416

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



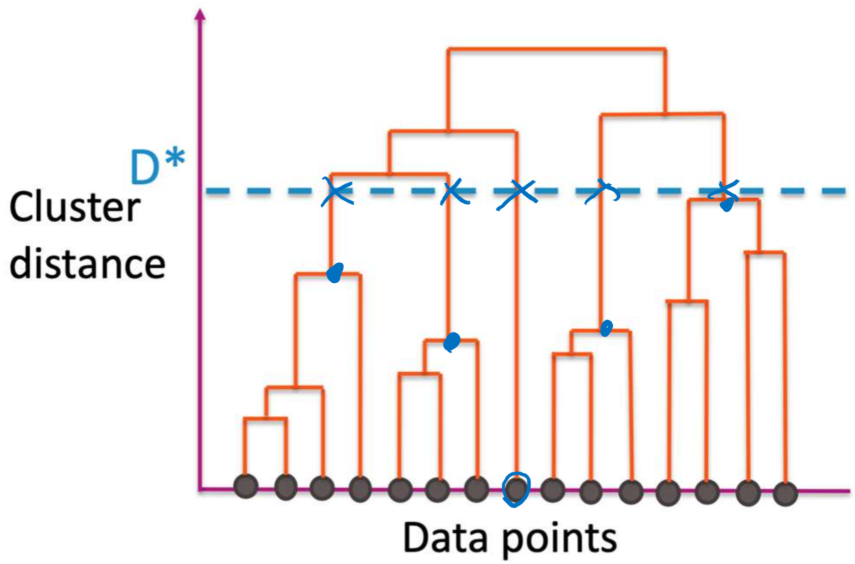
Poll Everywhere

Pair

2 min

pollev.com/cse416

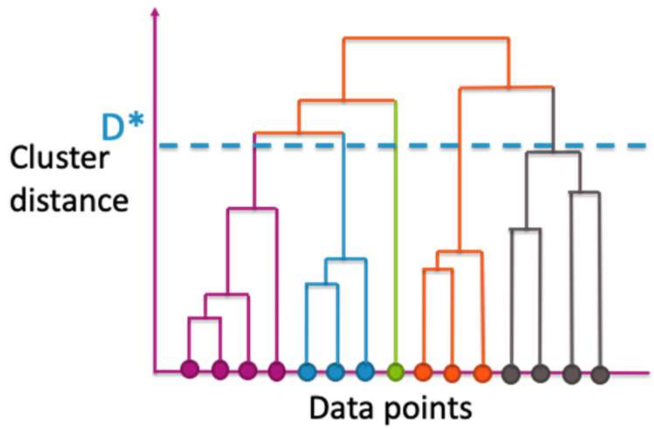
How many clusters would ~~be~~ we have if we use this threshold?

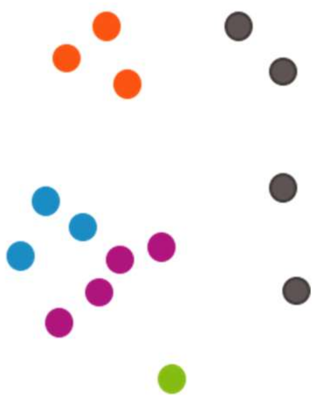


5

Cut Dendrogram

Every branch that crosses D^* becomes its own cluster

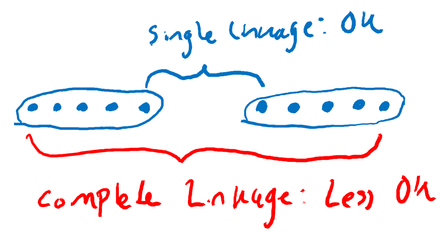


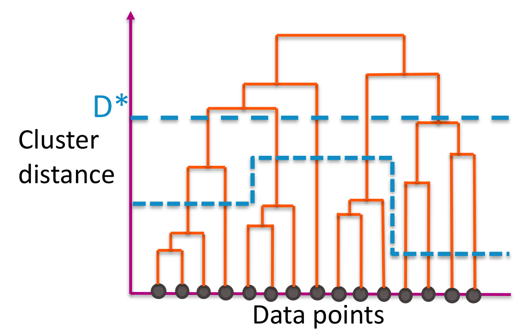


Choices to Make

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 $\mathcal{O}(n^2)$

Concept Inventory

11:50

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

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 before looking at your notes.