EEP 596: Adv Intro ML || Lecture 9

Dr. Karthik Mohan

Univ. of Washington, Seattle

February 2, 2023

Last time

1. k-means
2. k-means ++

Euclidean Distance, Starting points

Fixes the starting point issue
- Better Initialization

Assign (K)

Centroid (C)

How to choose the number of clusters? (K)

1. Prior Knowledge (E.g. #species of whales)
2. Inter-cluster Distance ↑
   Intra-cluster Distance ↓

Apply:
- Outlier/Anomaly Detection
- Cloud Infra Monitoring
- Finding Representative Embeddings

Today

1. **Clustering** k-means recap
2. **Clustering** Agglomerative Clustering
3. **Data Visualization** tSNE for Data Visualization

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance. (kMeans)

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.
5. k-means++ - Improvement on k-means and yields better quality clusters

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.
5. k-means++ - Improvement on k-means and yields better quality clusters
6. Both k-means and k-means++ suffer from the clusters being spherical in nature. What if the true cluster shapes look different? Next lecture: Kernel k-means and Agglomerative clustering!

k-means summary

1. k-means - A generic clustering algorithm that can take N data points and group them into K clusters based on Euclidean distance.
2. Clusters make sense if cluster division makes sense based on Euclidean distance.
3. k-means has a computational complexity of $O(Nd)$ for C step and $O(Nkd)$ for A step
4. Counter examples of clusters that may not be clustered well by k-means. Can use other methods for this.
5. k-means++ - Improvement on k-means and yields better quality clusters
6. Both k-means and k-means++ suffer from the clusters being spherical in nature. What if the true cluster shapes look different? Next lecture: Kernel k-means and Agglomerative clustering!
7. Clustering can help with cold-start problem. E.g. recommending new products!

Clustering in 2 dimensions - tSNE!

Data visualization method for clusters in 2 dimensions

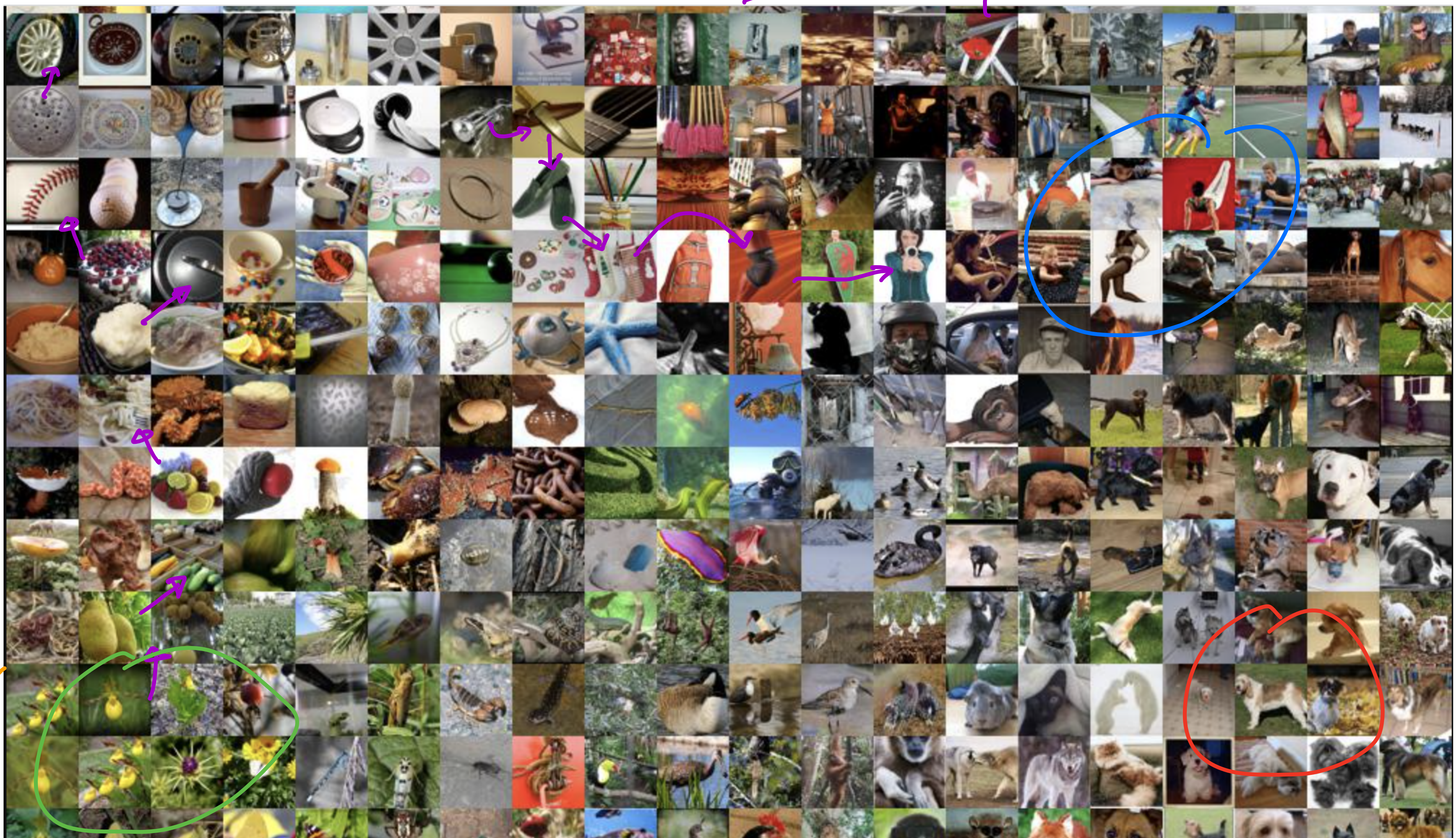
Clustering for Data Visualization

Images

Let's say we had 1000 images and wanted to "cluster" them onto a super-grid of images so that similar images are closely placed on the super-grid and dis-similar are placed further away. k-means clustering will only get us half-way there!

"soft clustering"

Data Visualization: Stochastic Neighborhood Embeddings (SNE)!



SNE

↳ Stochastic Neighborhood Embedding

High-level Idea

Find an embedding of images in 2 dimensions that put similar images close to each other and dis-similar images further away from each other.

3) Embedding → $[\ ] \in \mathbb{R}^{128}$ or $\in \mathbb{R}^{256}$

2) Neighborhood

1) Stochastic

↳ Randomness

tSNE $\begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$

SNE

High-level Idea

Find an embedding of images in 2 dimensions that put similar images close to each other and dis-similar images further away from each other.

Soft clustering

We don't have a K here. But if you look at any neighborhood of the super grid of images - They will look similar! We can call this soft-clustering.

→ soft clustering

SNE

Similarity measure through Probabilities

Let $x_1, x_2, \ldots$ represent features of the data in their original dimensions (e.g. images).

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}$$

Euclidean Distance

$\frac{1}{\sqrt{2\pi}} e^{-x^2/2\sigma^2}$ ~ p.d.f. for $N(0,1)$

$p_{j|i}$ – prob of j being close to i

("prob of j given i")

$e^{-10} \approx 0$

$e^{0} \approx 1$

SNE

Similarity measure through Probabilities

Let $x_1, x_2, \ldots$ represent features of the data in their original dimensions (e.g. images).

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}$$

Low-dimensional embedding Probabilities

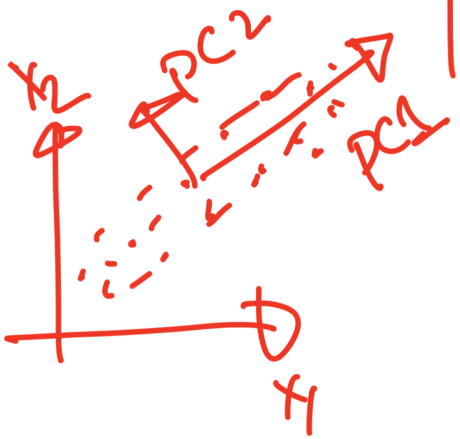
Let $y_1, y_2, \ldots$ represent features of the data in lower (embedded) dimensions (e.g. 2 dimensions).

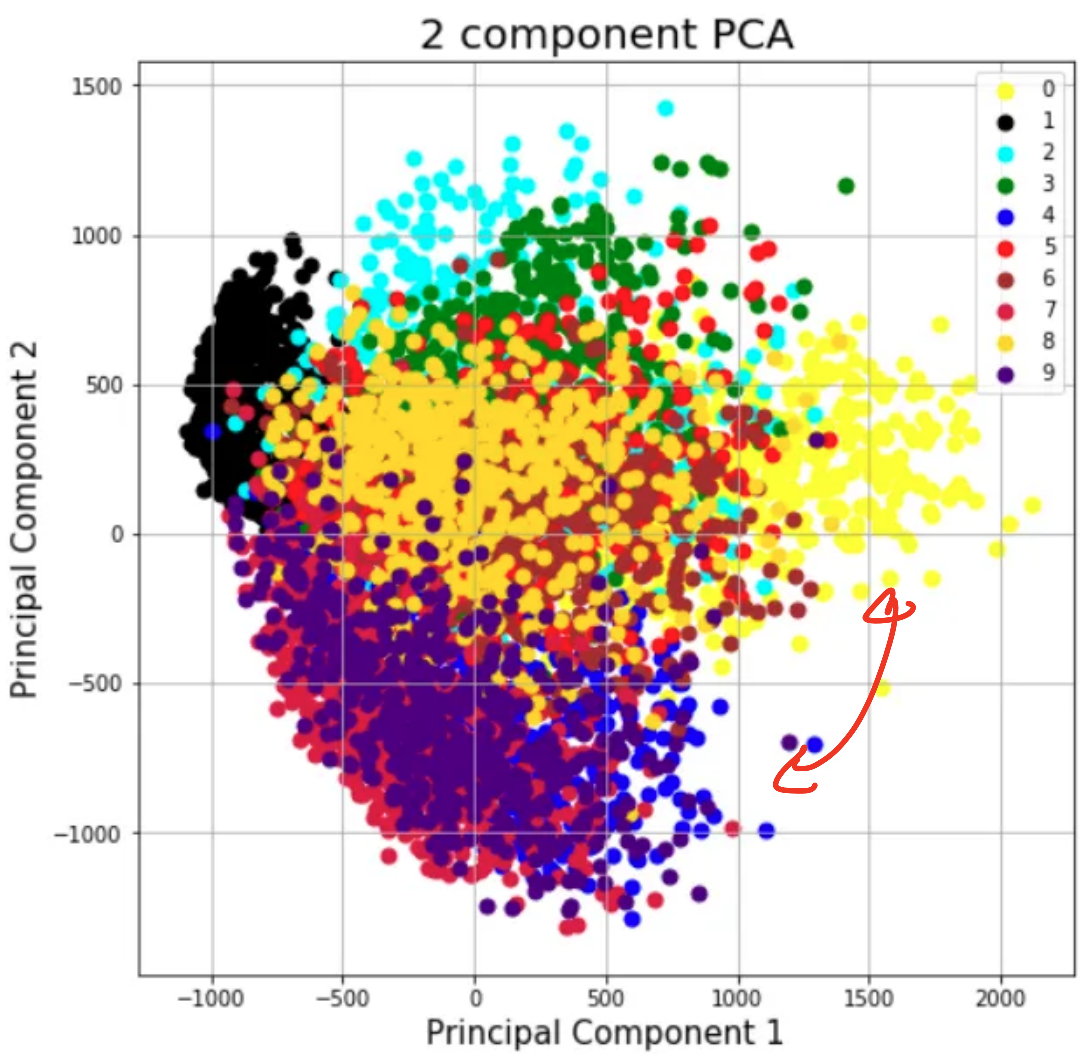
$$q_{j|i} = \frac{e^{-\|y_i - y_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|_2^2 / 2\sigma_i^2}}$$

MNIST digits data set

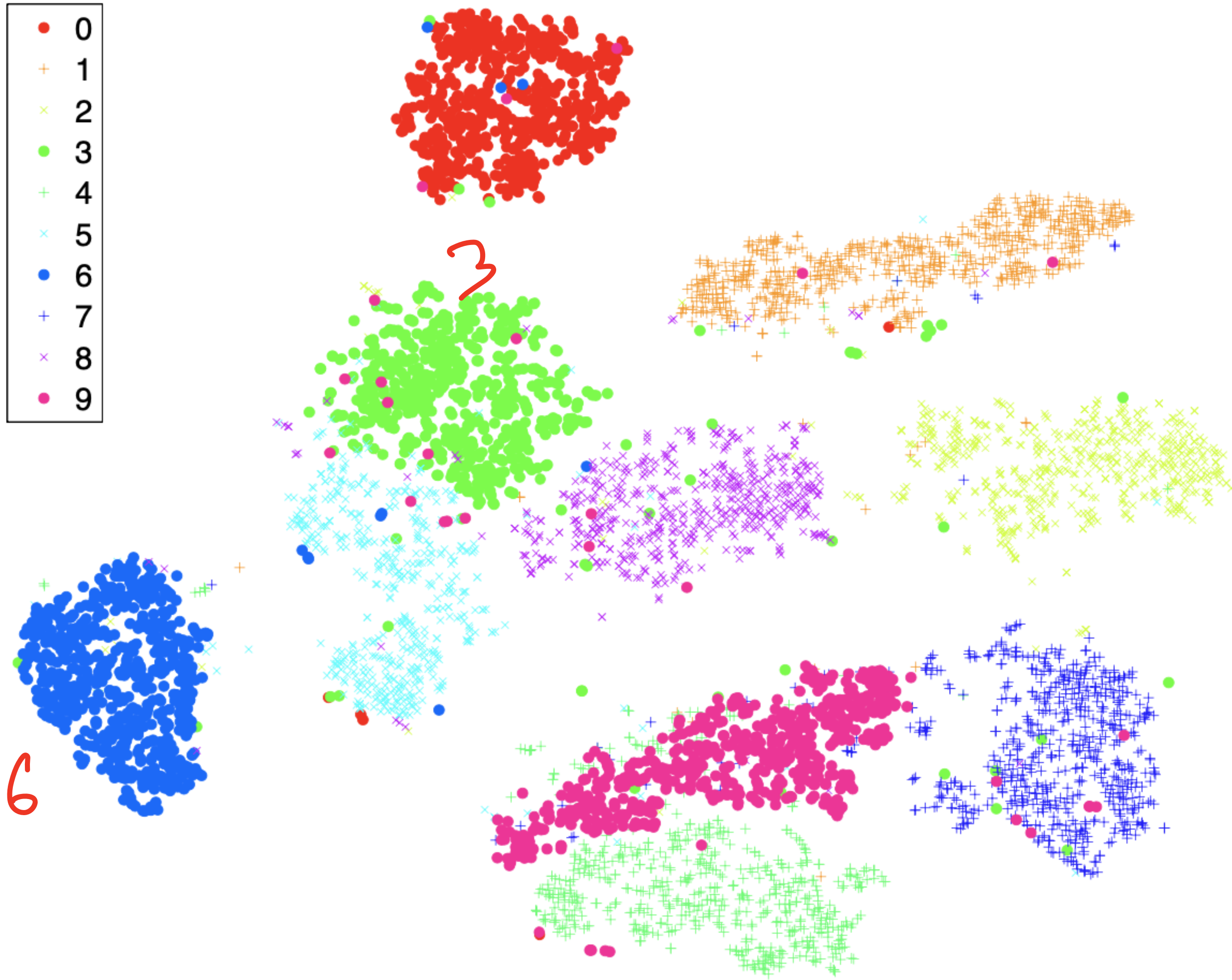
PCA on MNIST

1) PCA – Principal Component Analysis





MNIST tSNE embeddings



Estimating low-dimensional embeddings in SNE

A ~~similarity~~ Distance measure for Probabilities - KL Divergence

$$KL(p||q) = \sum_{i=1}^{d} p_i \log \frac{p_i}{q_i} \geq 0$$

KL(p||q) = 0 ⇒ p & q are the same

KL(p||q) ↑ ⇒ p & q are very different distn.

Loss fn. for Regression :- Quadratic

— " — for classification :- Cross-Entropy

— " — for t SNE :- KL Divergence (close to this)

Estimating low-dimensional embeddings in SNE

A similarity measure for Probabilities - KL Divergence

$$KL(p||q) = \sum_{i=1}^{d} p_i \log \frac{p_i}{q_i}$$

Loss function

$$L(y_1, y_2, \ldots, y_N) = \sum_{i=1}^{N} KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

N N

minimize L ⇒ obtain 2-dim embeddings through tSNE

Gradient and GD

Gradient

$$\frac{\partial L}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Gradient and GD

Gradient

$$\frac{\partial L}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

GD

$$y^{t+1} = y^t - \eta \frac{\partial L}{\partial y}(y^t)$$

Image Chain

ICE #1 (3 mins break out)

Let's say you want to create a video that has 1000 images (e.g. the one we looked at earlier) in a sequence so that the images in the video transforms smoothly from one to the next. How would you go about doing this if you learned a tSNE representation for the images?

Digits:- 000000001 1 1 1 2 2 2 2 2 2 2 . .

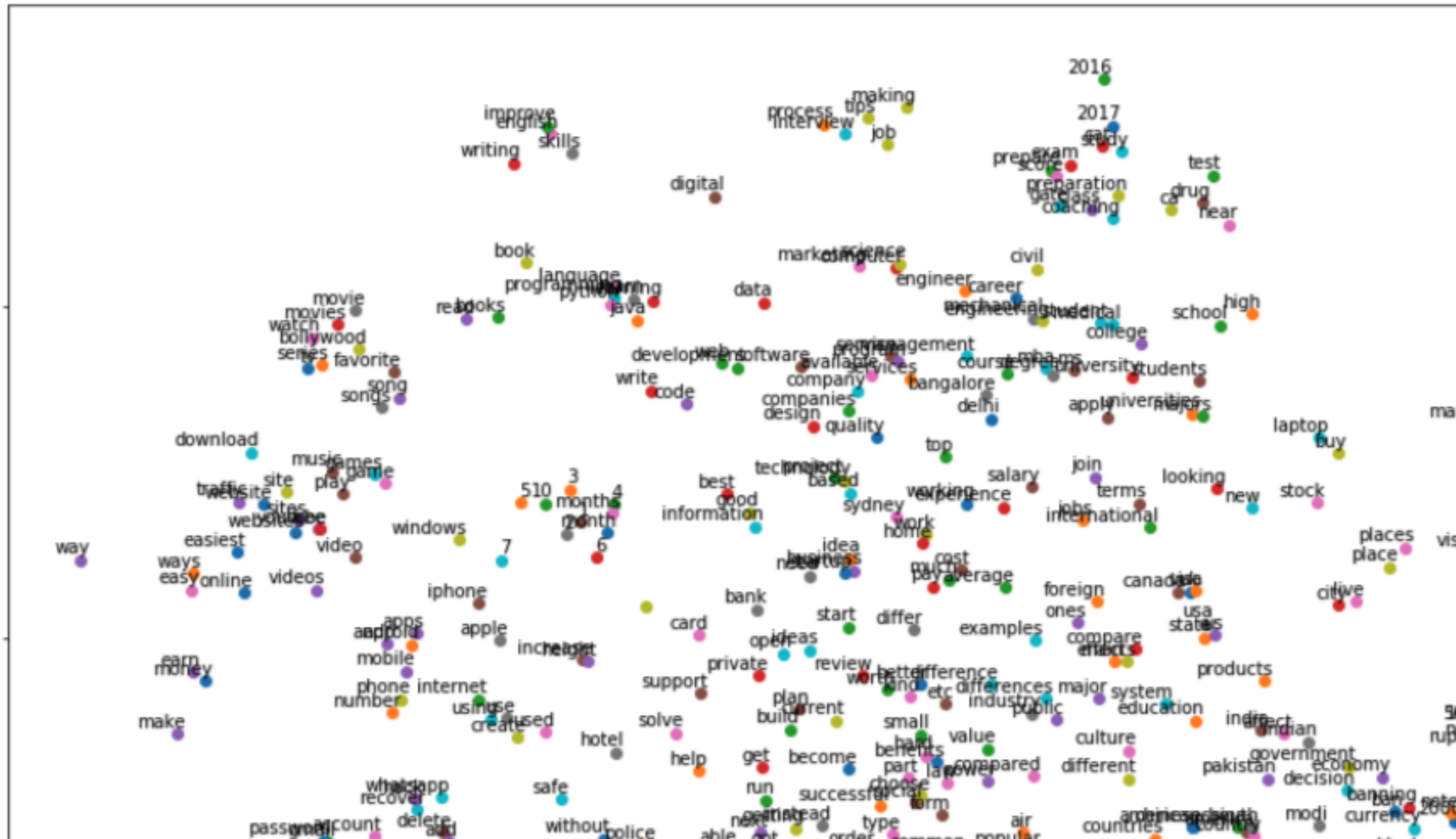
smooth transition

How do we create this grid?

tSNE Notebook Examples

Notebook
Fashion MNIST Notebook

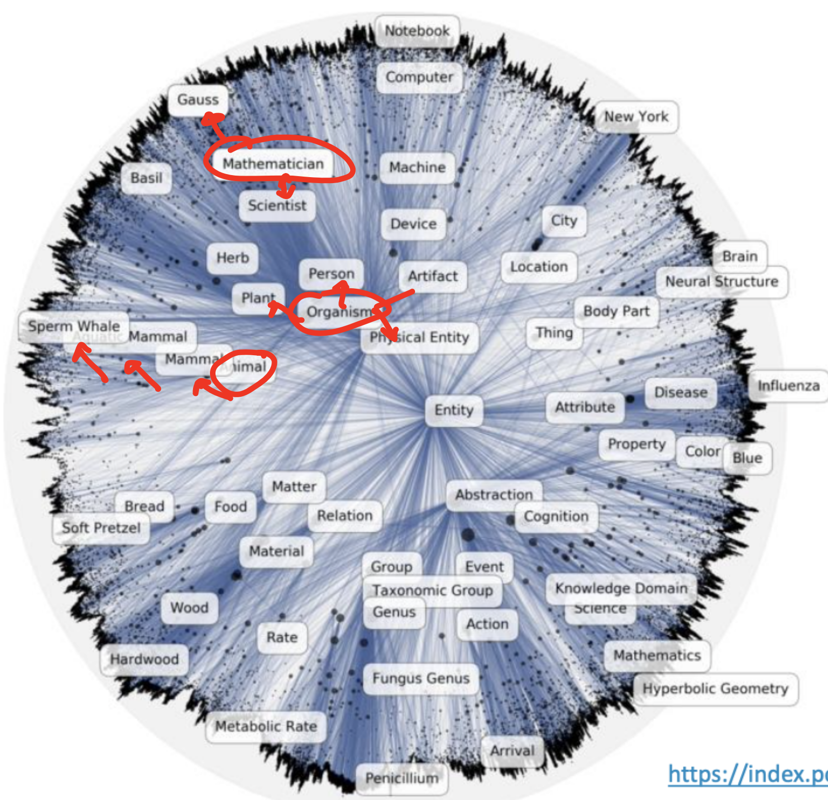
Word visualization based on word2vec



Hierarchical Clustering

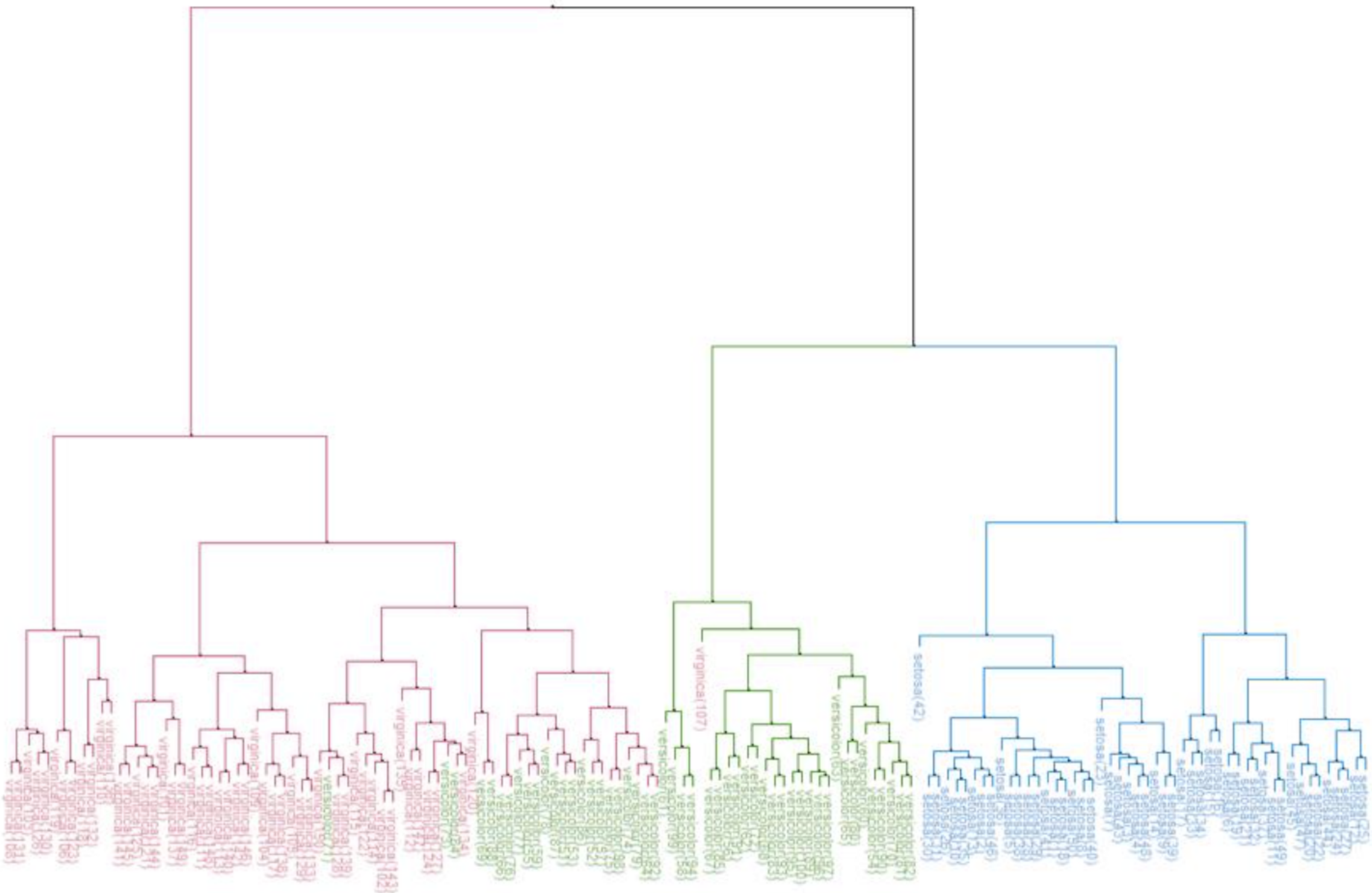
Example of Hierarchy: Nouns

Lots of data is hierarchical by nature



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

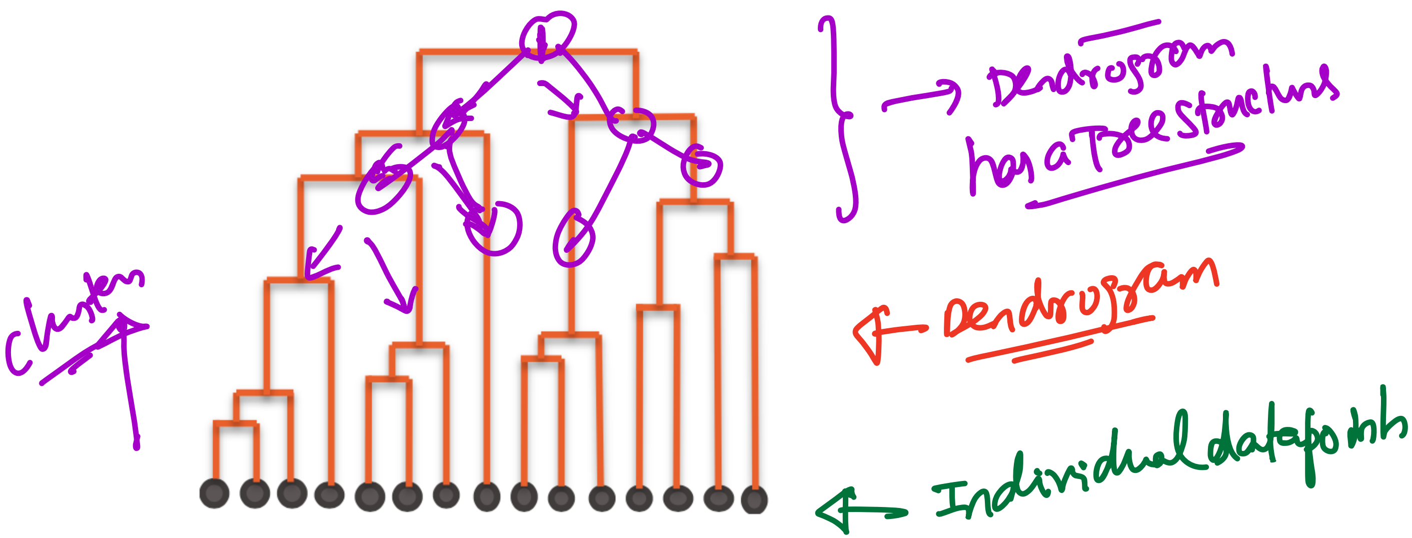
Example of Hierarchy: Species

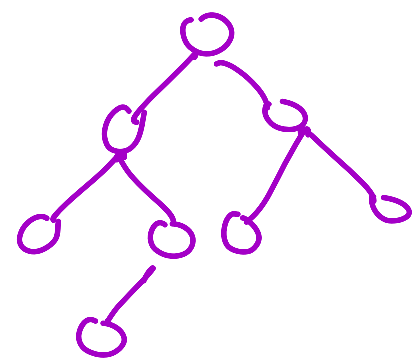


Motivation for Hierarchical Clustering

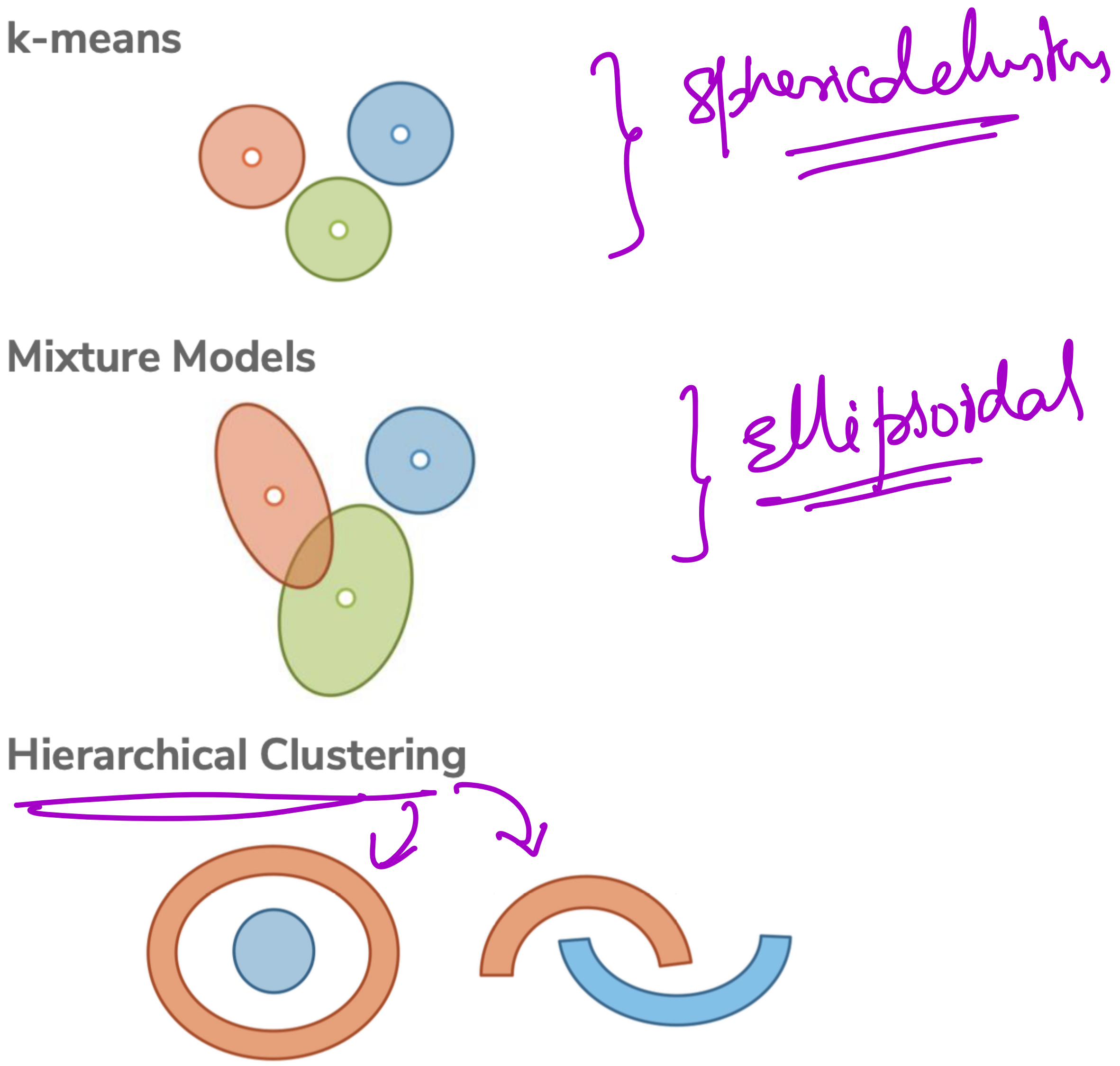
If we try to learn clusters in hierarchies, we can

- Avoid choosing the # of clusters beforehand (k)
- 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





Different shapes — Different algorithms



Types of Hierarchical Algorithms

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

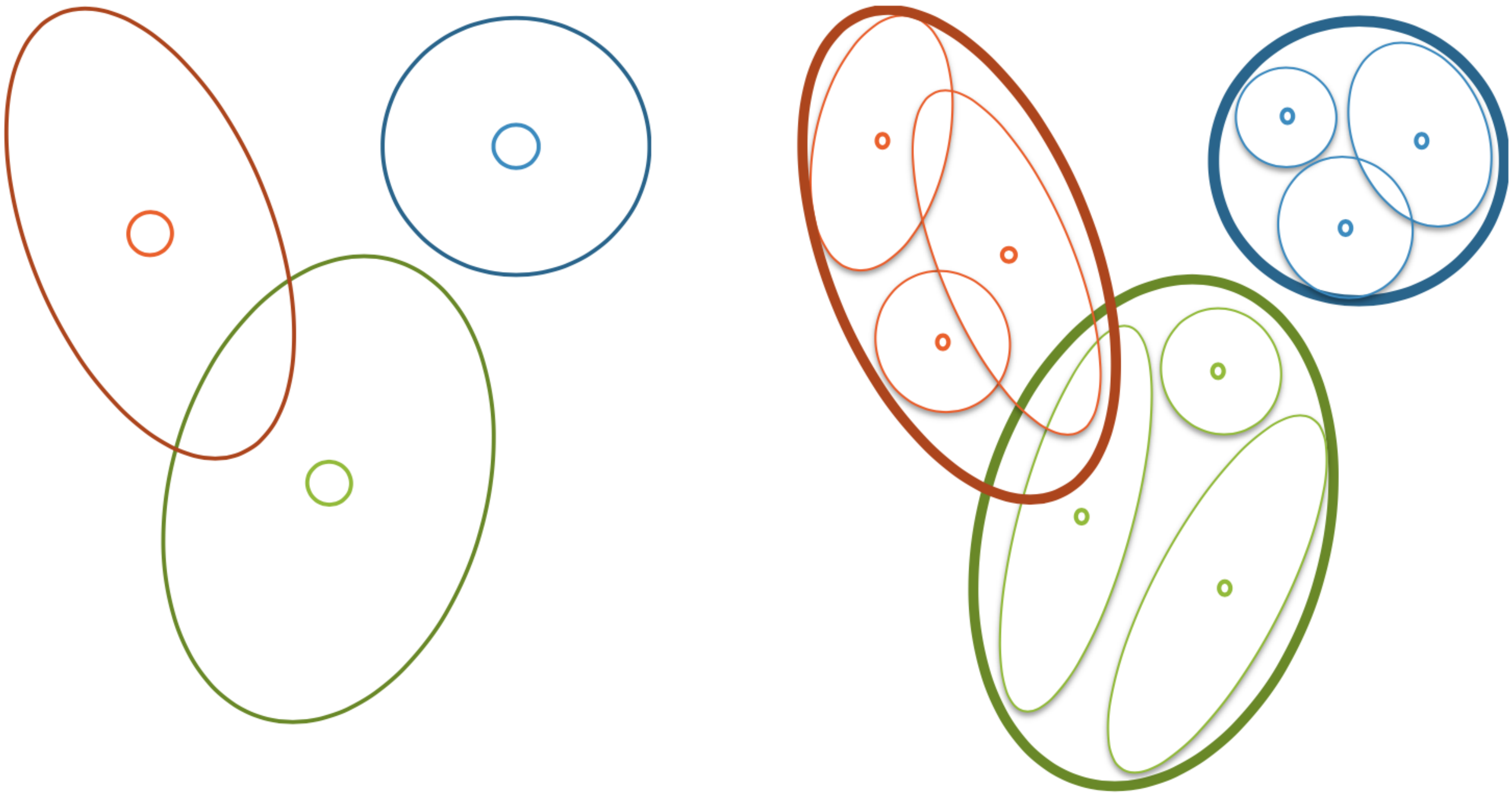
- Start with all the data in one big cluster and then recursively split the data into smaller clusters
  - Example: **recursive k-means**

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**

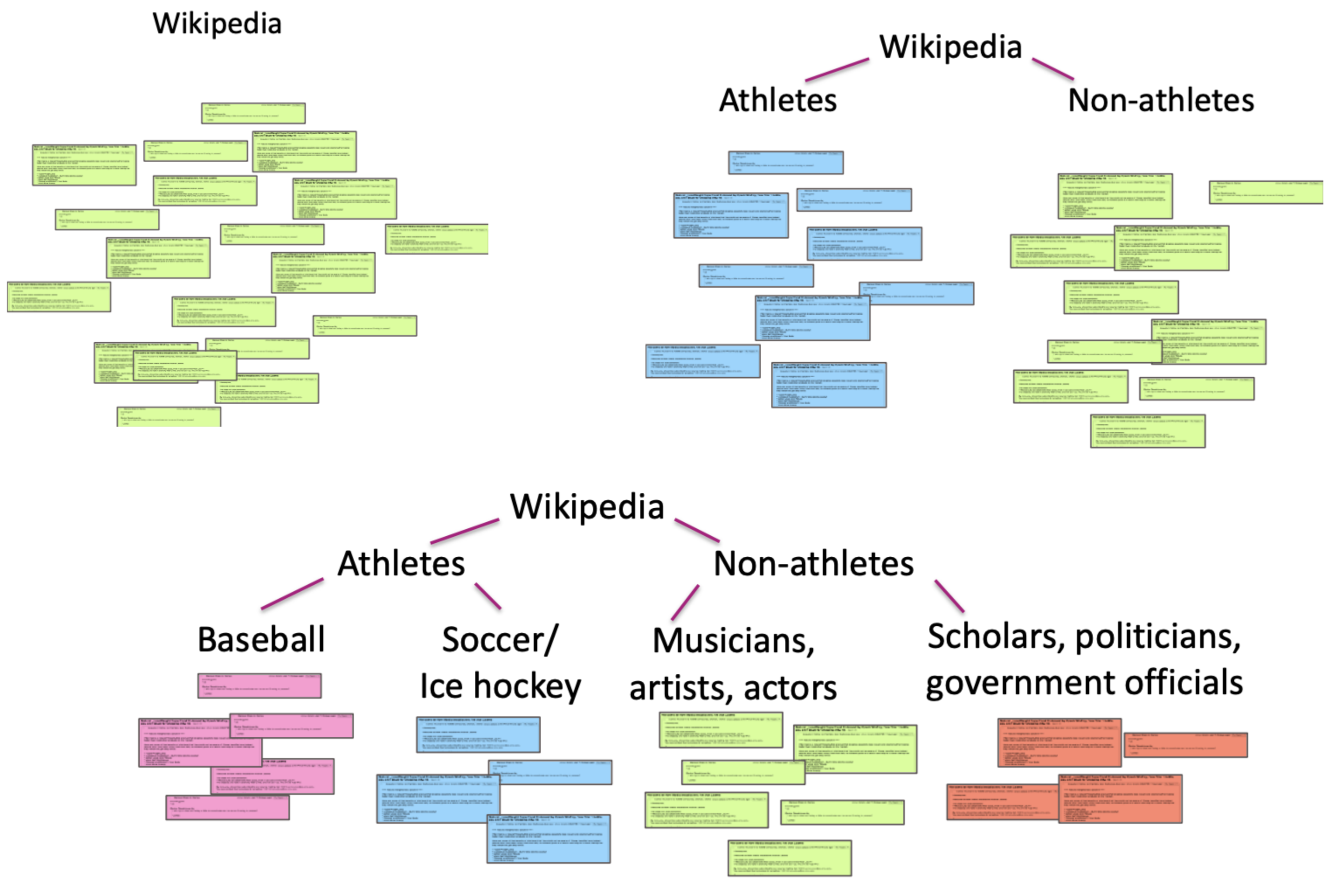
Divisive Clustering

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.



Wikipedia Example

Using Wikipedia



Hyper-parameters for Divisive Clustering

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

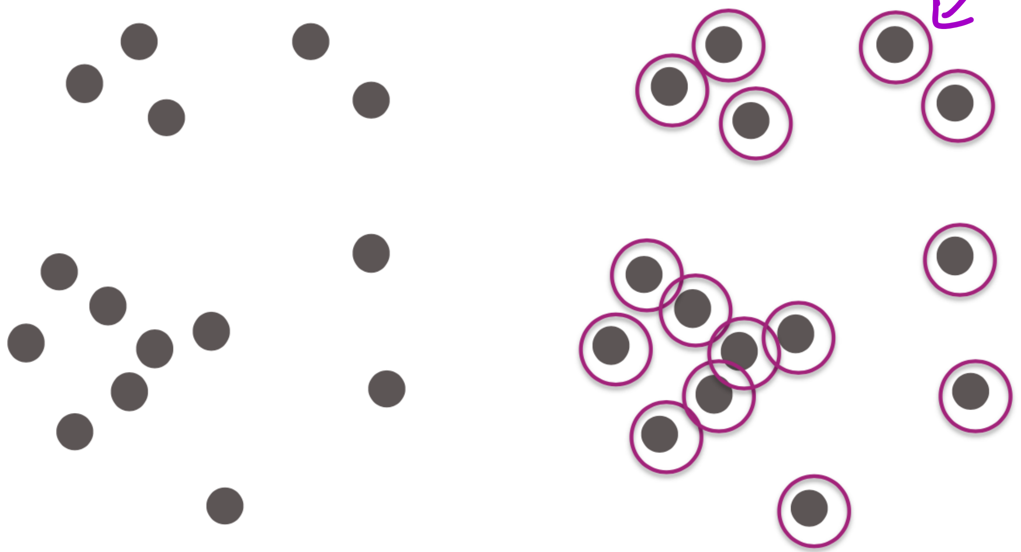
While there is more than one cluster

   3. Merge the two closest clusters

Agglomerative Clustering: Step 1

1. Initialize each point to be its own cluster

each pt. in its own cluster

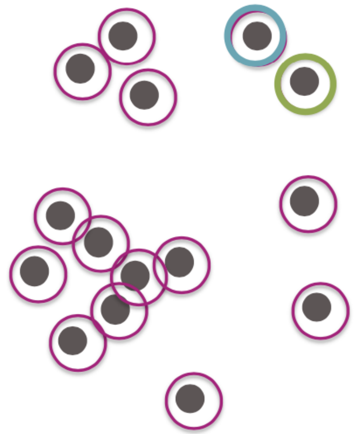


Agglomerative Clustering: Step 2

1. Distance bet points
Euclidean

2. Distance between clusters!

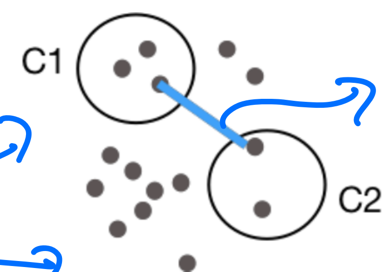
2. Define a distance metric between clusters



Single Linkage

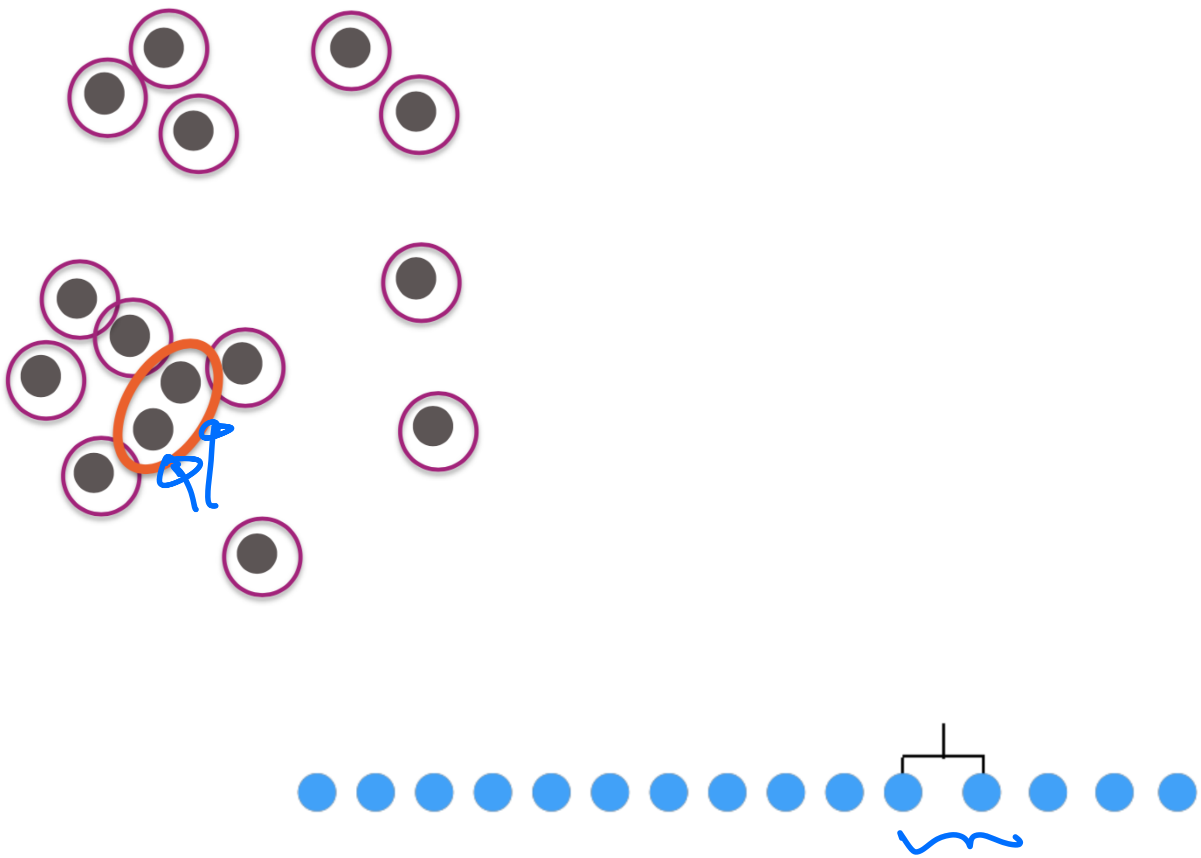
$$distance(C_1, C_2) = \min_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$$

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

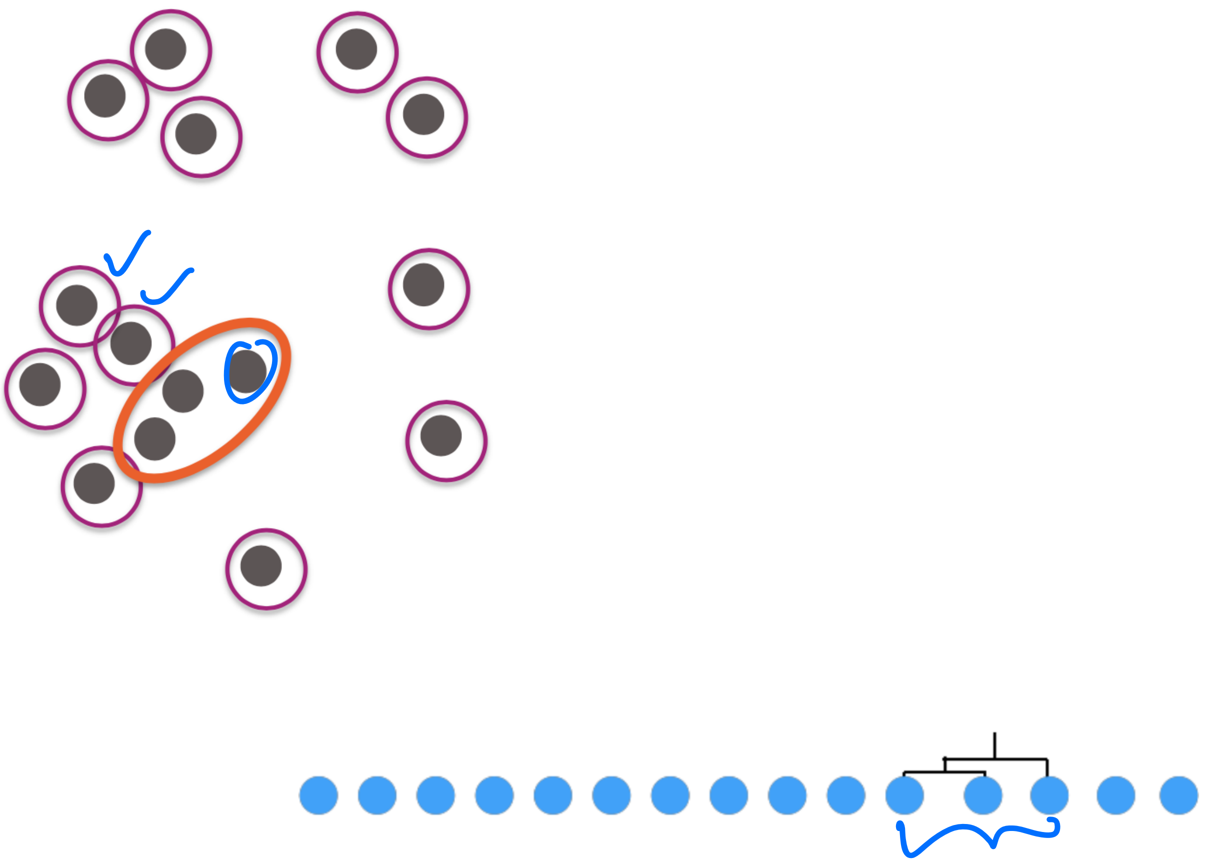


Agglomerative Clustering: Step 3

Merge closest pair of clusters

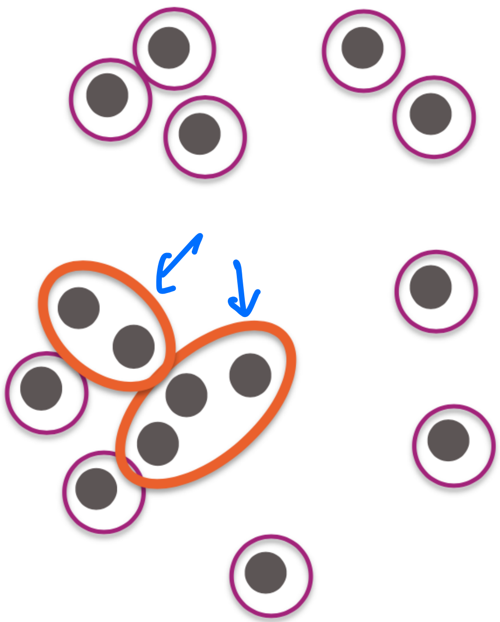


Agglomerative Clustering: Repeat

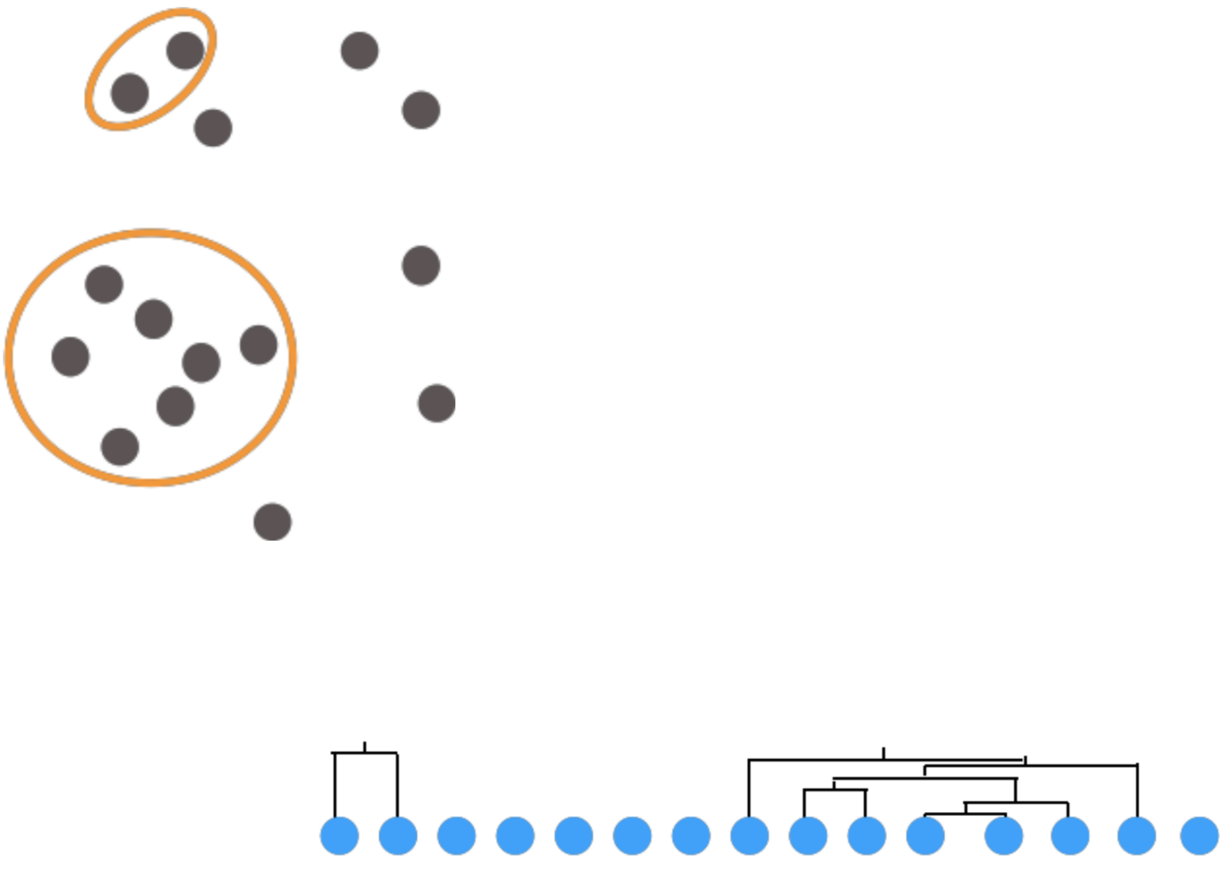


Agglomerative Clustering: Repeat

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



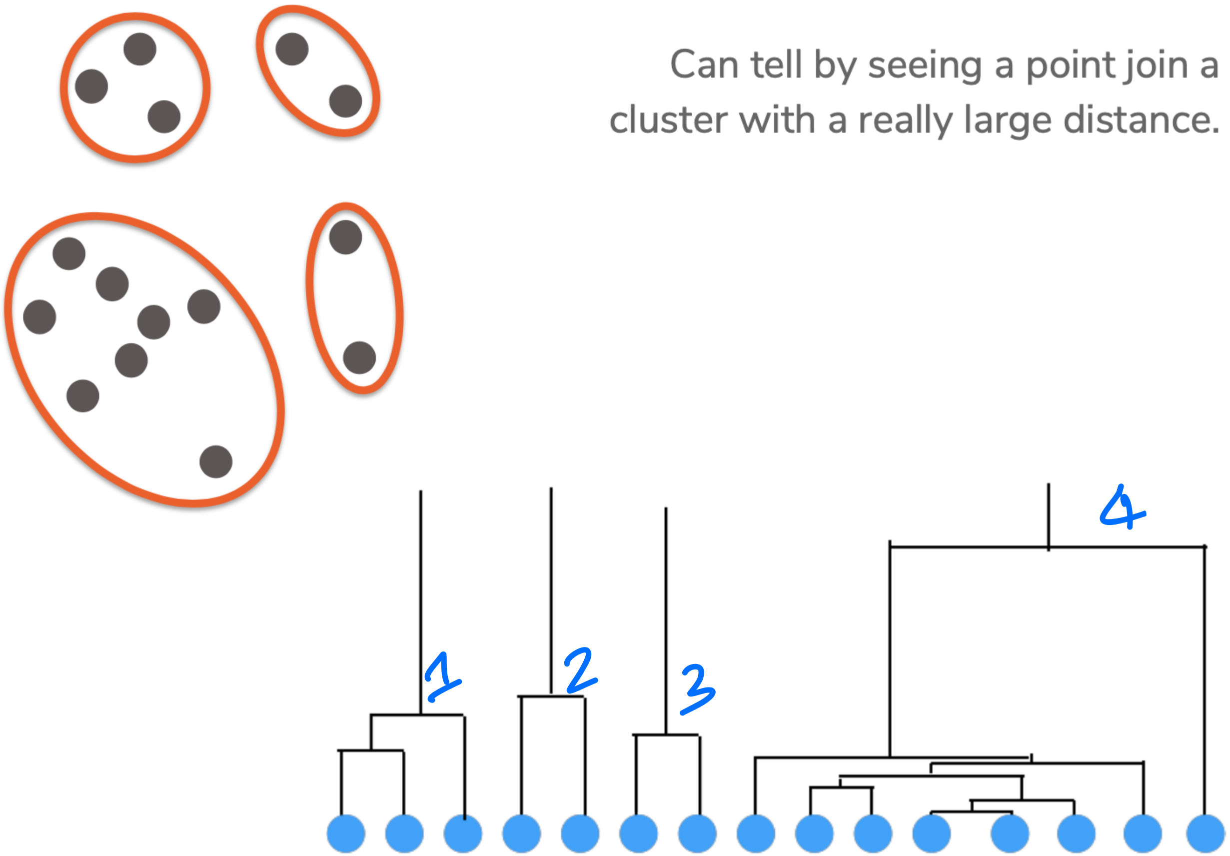
Agglomerative Clustering: Repeat



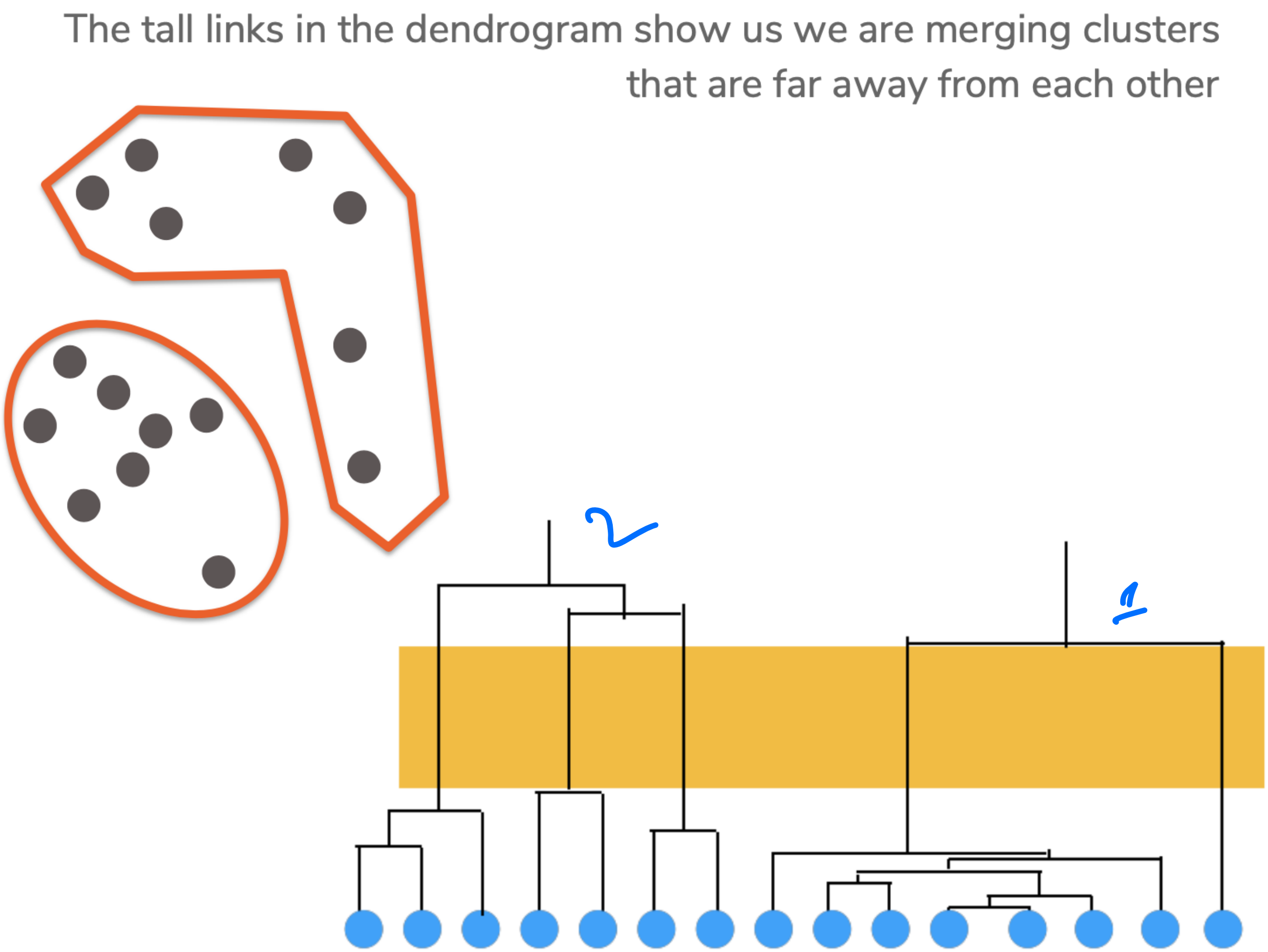
Agglomerative Clustering: Repeat

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

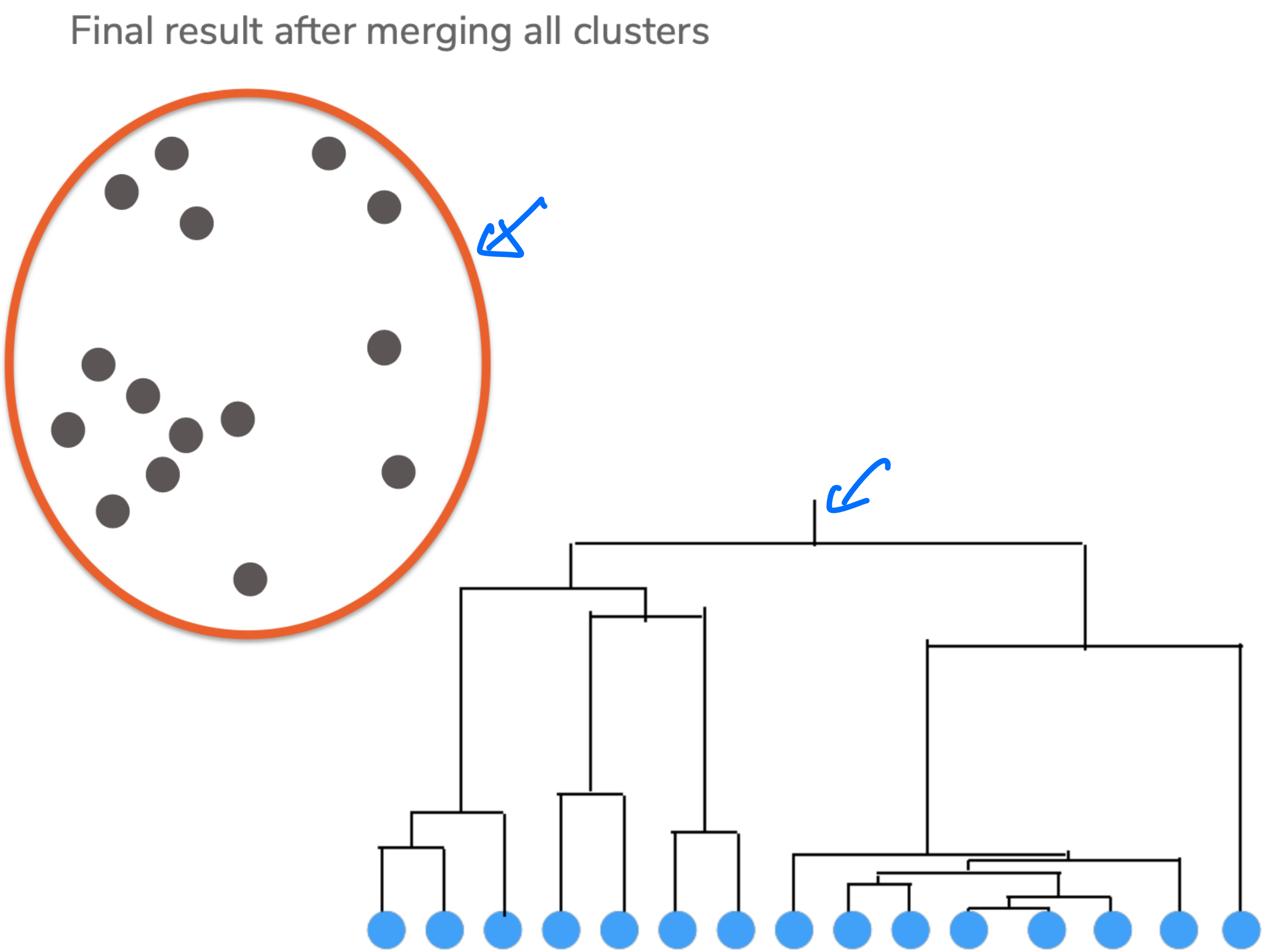
Can tell by seeing a point join a cluster with a really large distance.



Agglomerative Clustering: Repeat

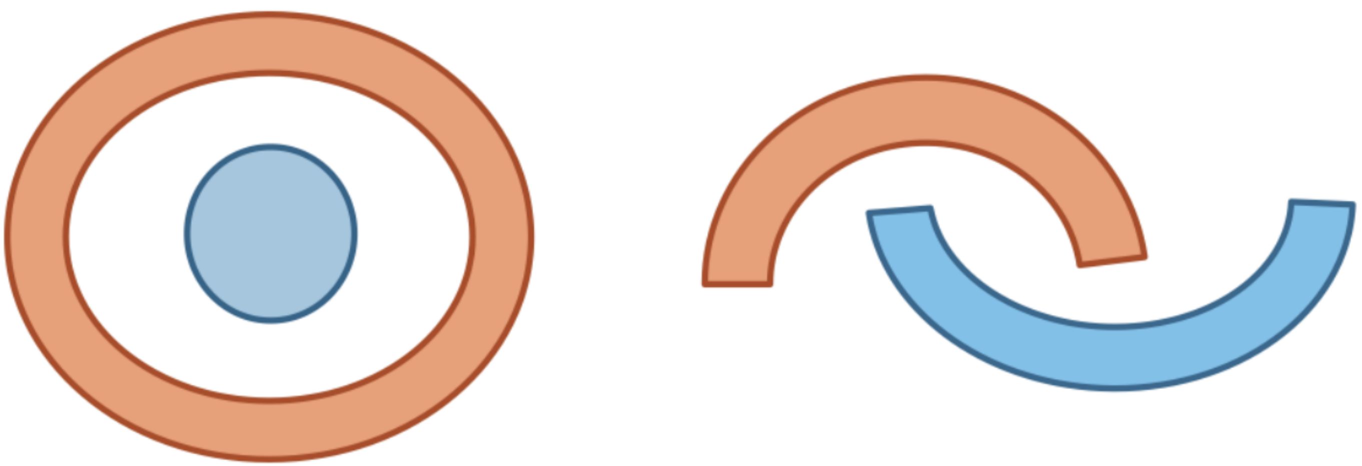


Agglomerative Clustering: Repeat



Agglomerative Clustering: Spiral and Donut!

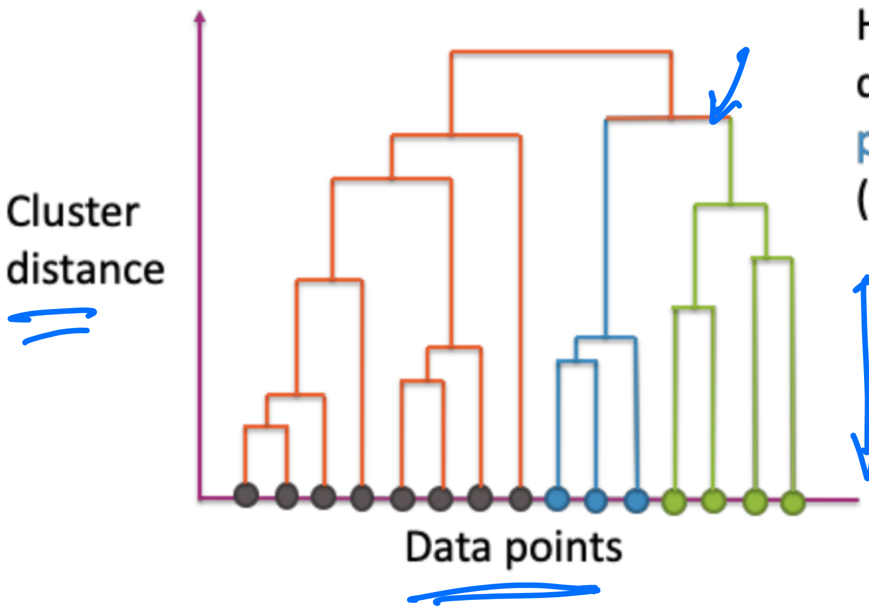
With agglomerative clustering, we are now very able to learn weirder clusterings like



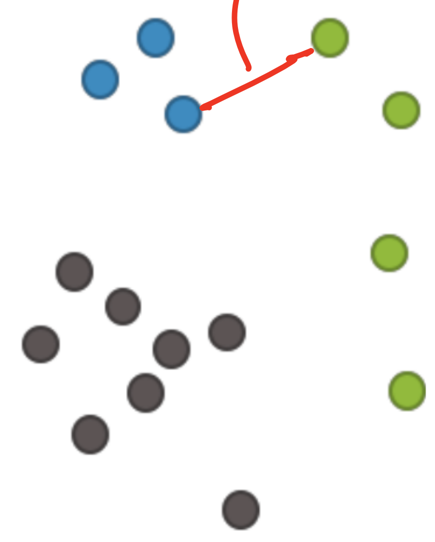
Dendrogram

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

y-axis shows distance between pairs of clusters



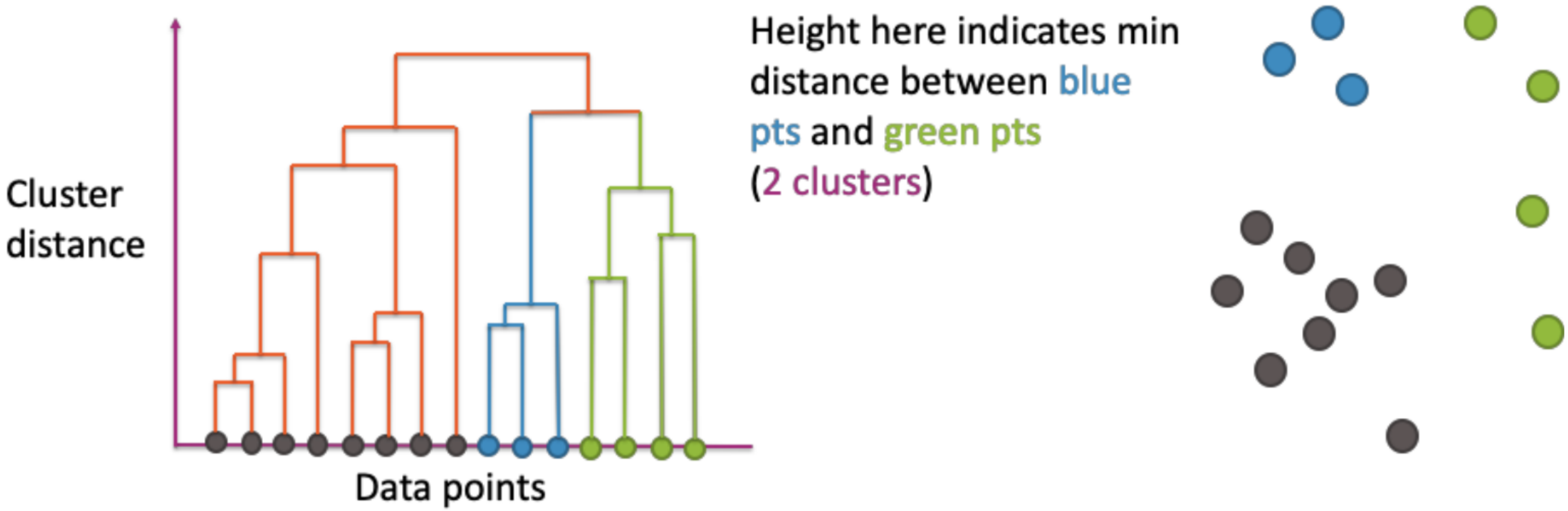
Height here indicates min distance between blue pts and green pts (2 clusters)



Dendrogram

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

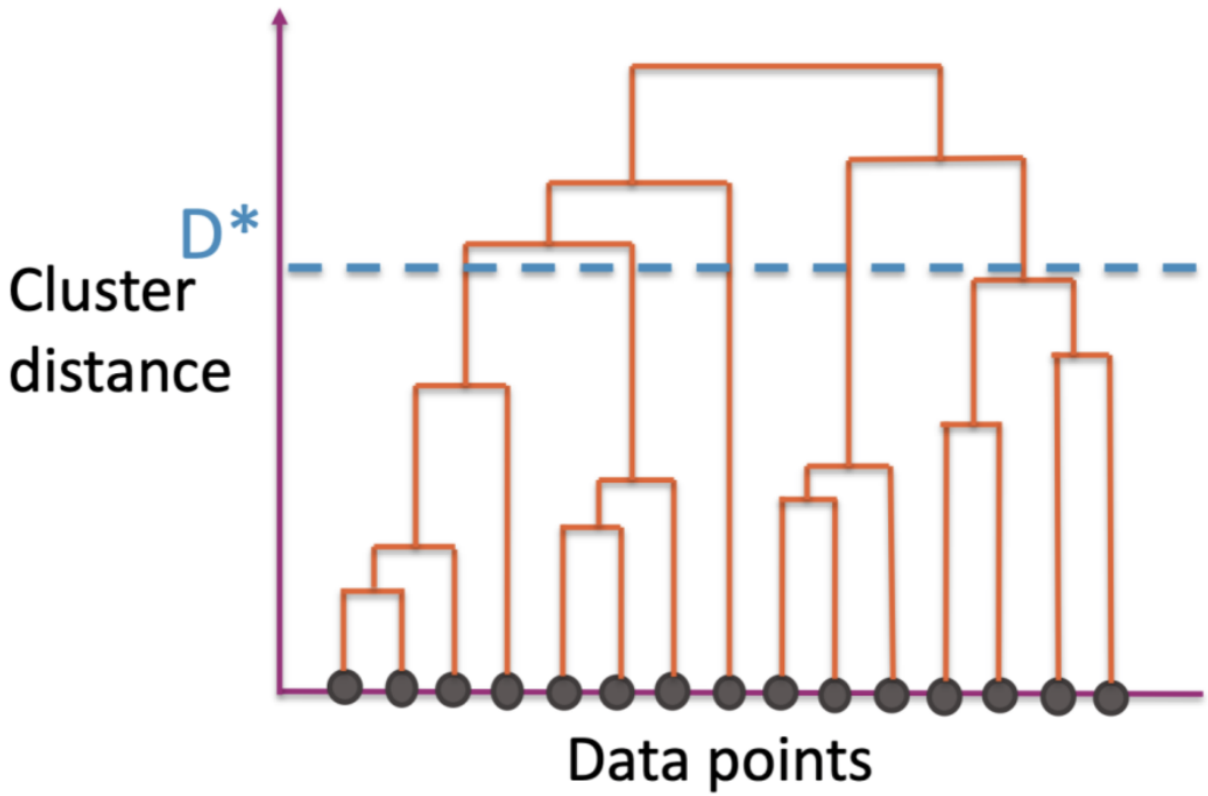
y-axis shows distance between pairs of clusters



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

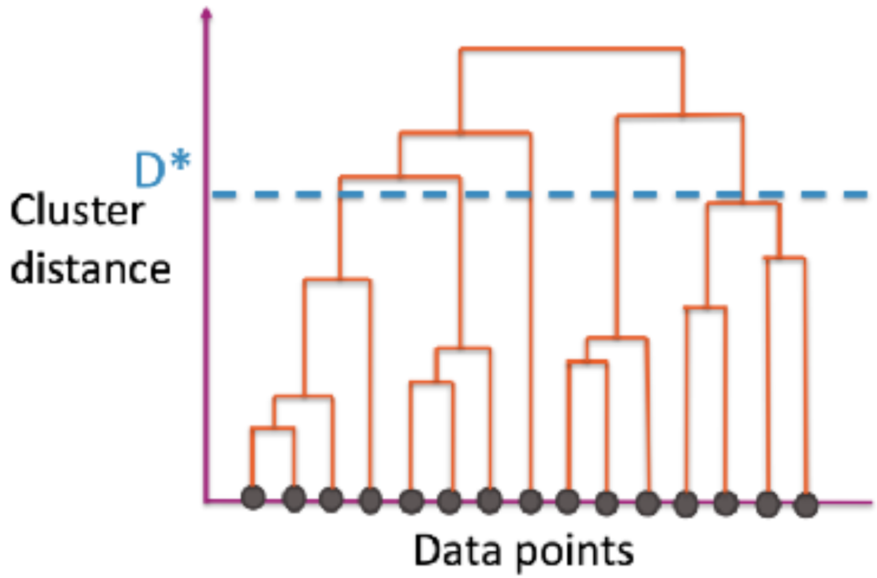


Dendrogram ICE

ICE #2

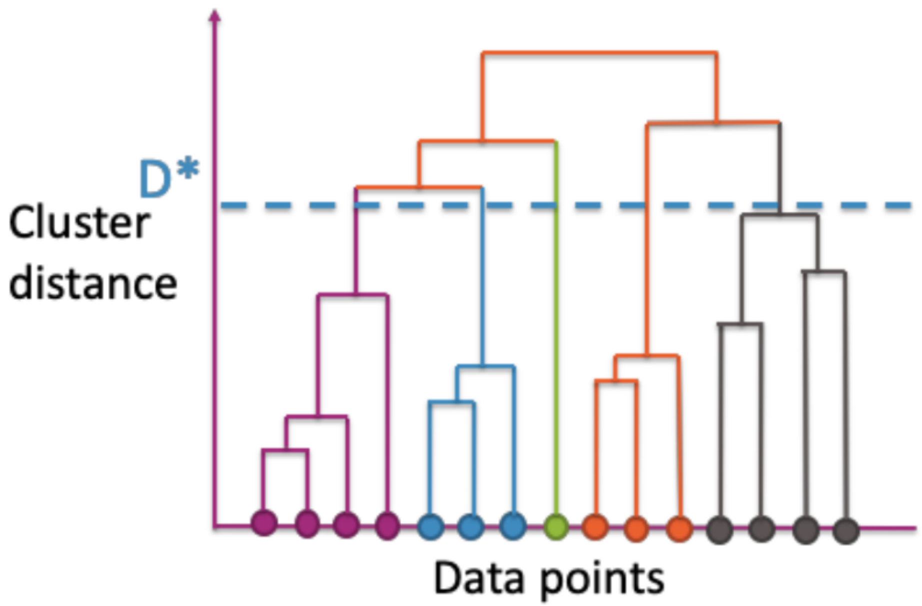
How many clusters would we have if we use this threshold to cut?

- a) 4
- b) 5
- c) 6
- d) 7



Cut Dendrogram

Every branch that crosses D^* becomes its own cluster



Agglomerative Clustering — Hyper-parameters

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

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

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

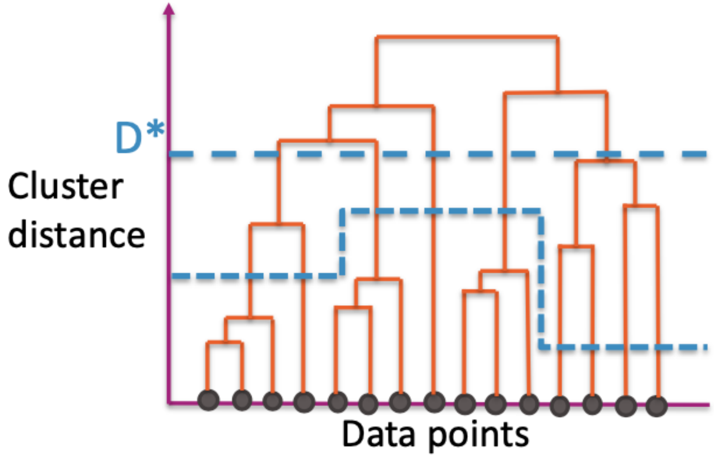
  - Complete Linkage:

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

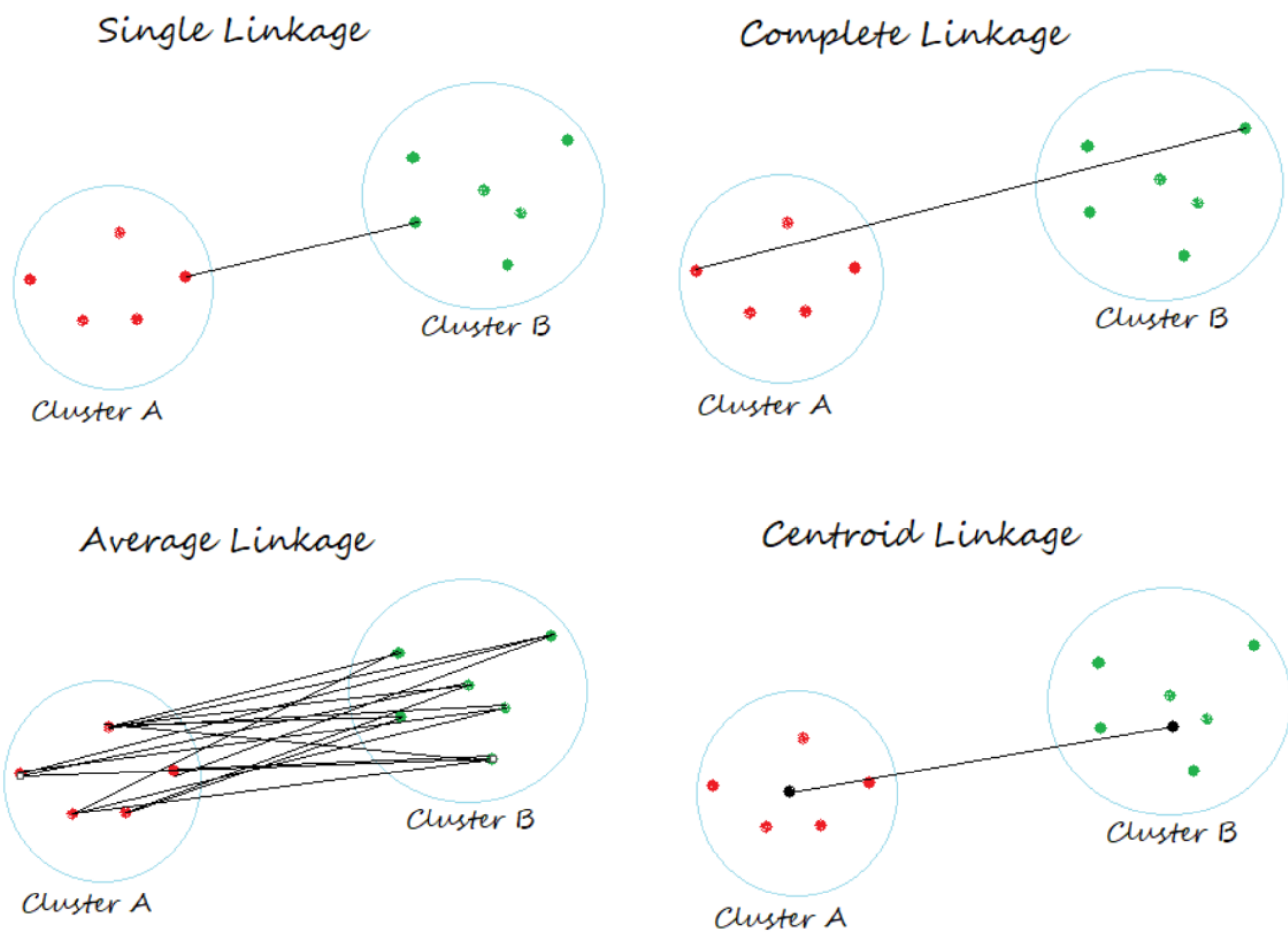
  - Centroid Linkage

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

  - Others
- Where and how to cut dendrogram



Linkage examples



Dendrogram ICE

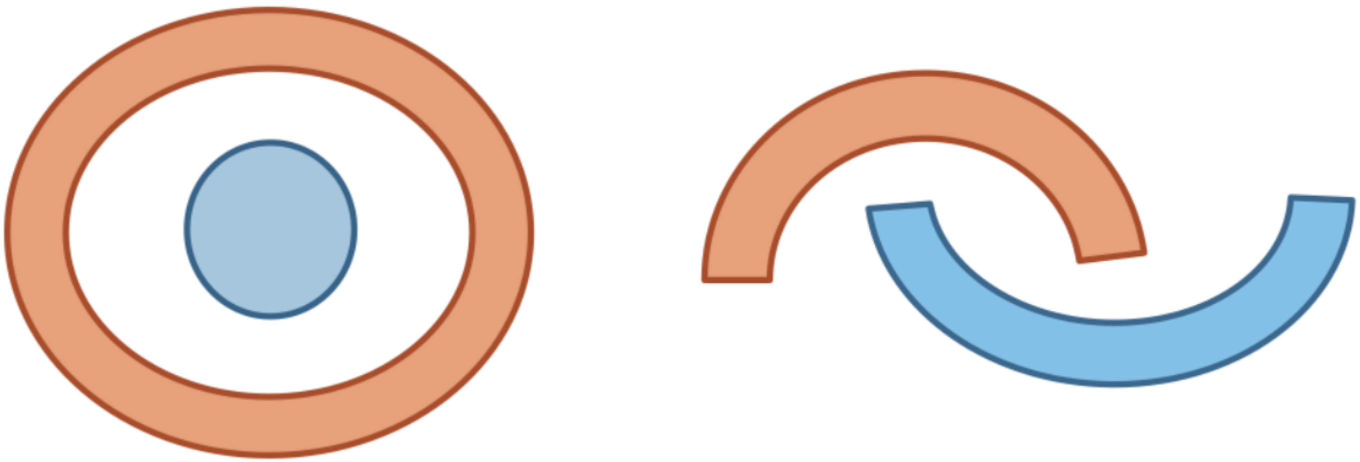
ICE #3

Which linkage function is more likely to detect spiral clusters?

a. Single Linkage
b. Centroid Linkage
c. Complete Linkage
d. Any Linkage

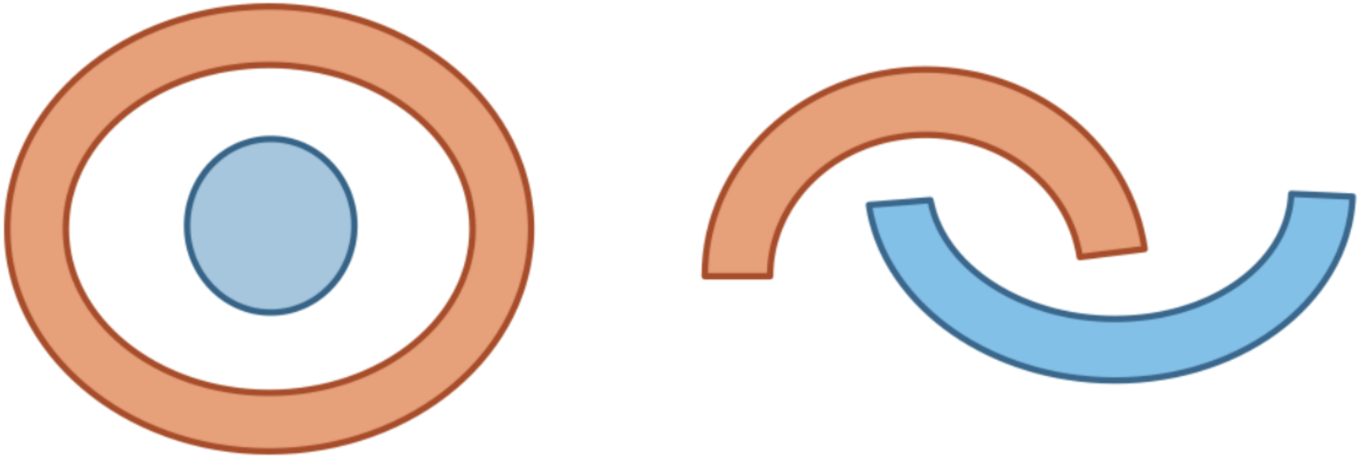
Centroid Linkage Applied to Spiral

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

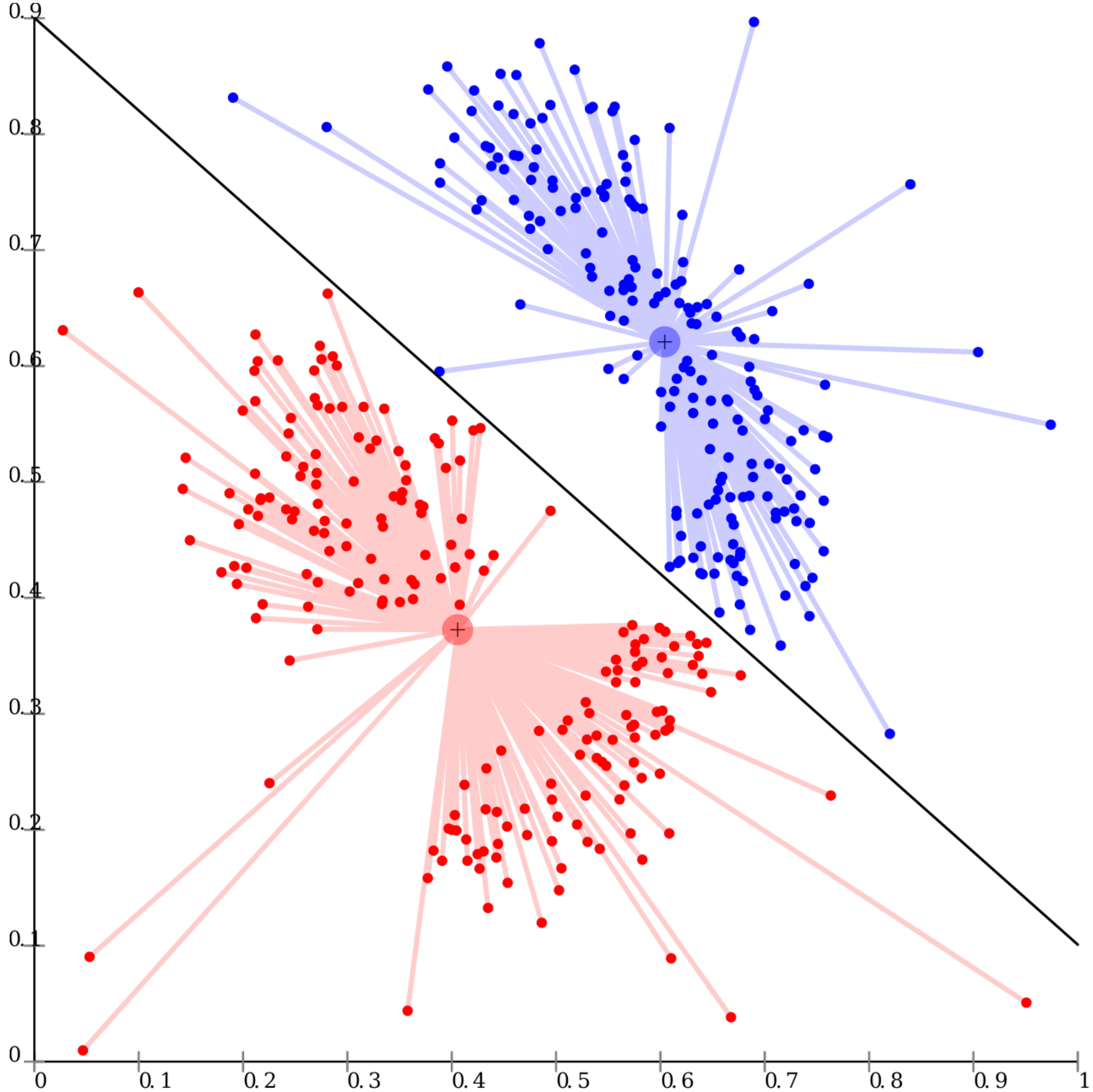


Single Linkage Applied to Spiral

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



Where Centroid Linkage Works!



Dunn Index - Metric that measures goodness of clusters

Dunn Index

$$D = \frac{\min_{1 \leq i < j \leq K} d(i,j)}{\max_{1 \leq j \leq K} d'(j)}$$

Dunn Index - Metric that measures goodness of clusters

Dunn Index

$$D = \frac{\min_{1 \leq i < j \leq K} d(i,j)}{\max_{1 \leq j \leq K} d'(j)}$$

ICE #4

Say you had a single-linkage and k-means clustering applied to a data set to produce K clusters each. Call them A and B. When would you say single-linkage produces better clustering than k-means?

a. $D(A) > D(B)$
b. $D(B) > D(A)$

Dendrogram

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.

Dendrogram

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

Comparison of Clustering Algorithms

Quick comparison

	k-means	Agglomerative Clustering
Computation	$O(Ndk)$	$O(N^2d)$
Type	Spherical	Arbitrary shapes

Few more points..

a. Weigh computational complexity with complexity of clustering - kmeans vs agglomerative
b. Agglomerative distance choices yield different sets of clusters (single linkage vs centroid)
c. Clustering in practice is an art
d. However, quality of clustering can be evaluated - E.g. through Dunn Index!