Hierarchical Clustering

Moving away from strict partitions

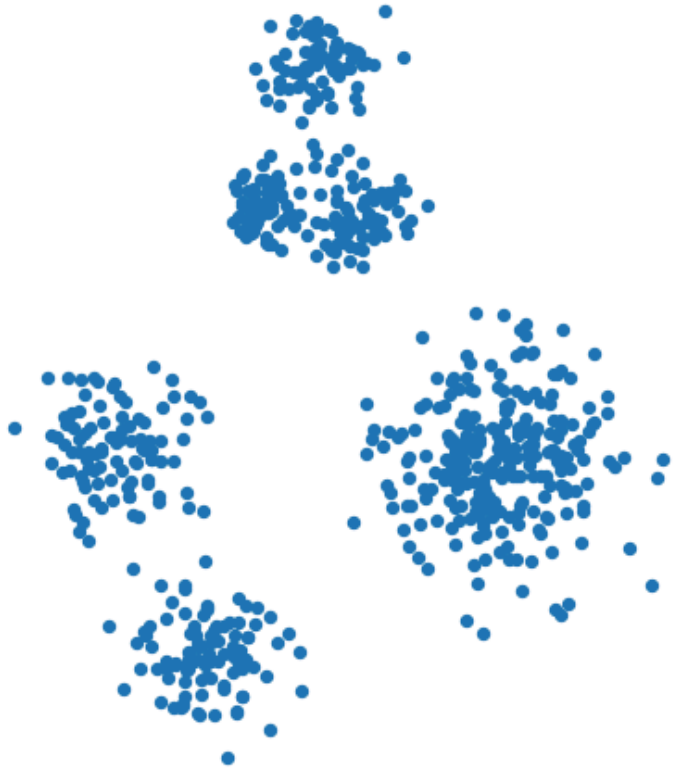
- Code

Example

How Many Clusters?

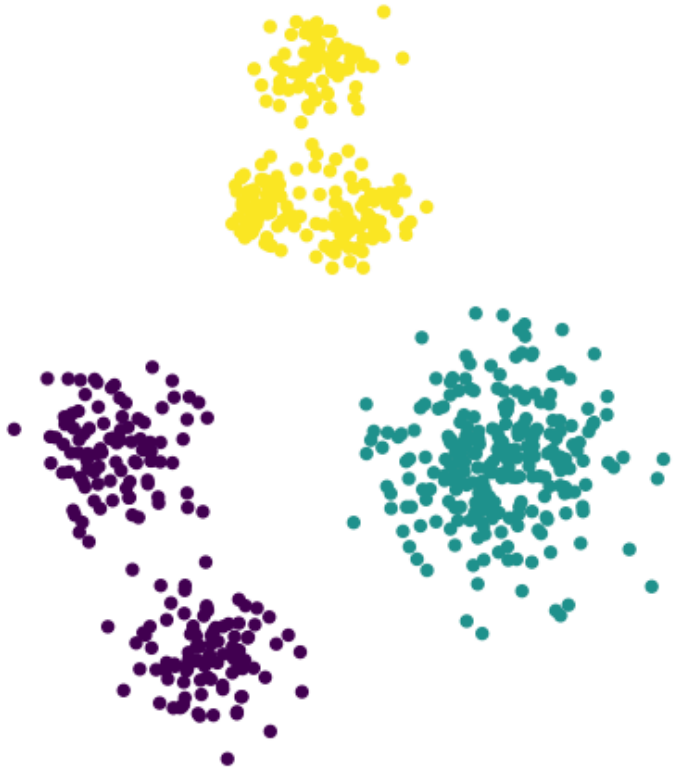
How many clusters would you say there are here?

▶ Code



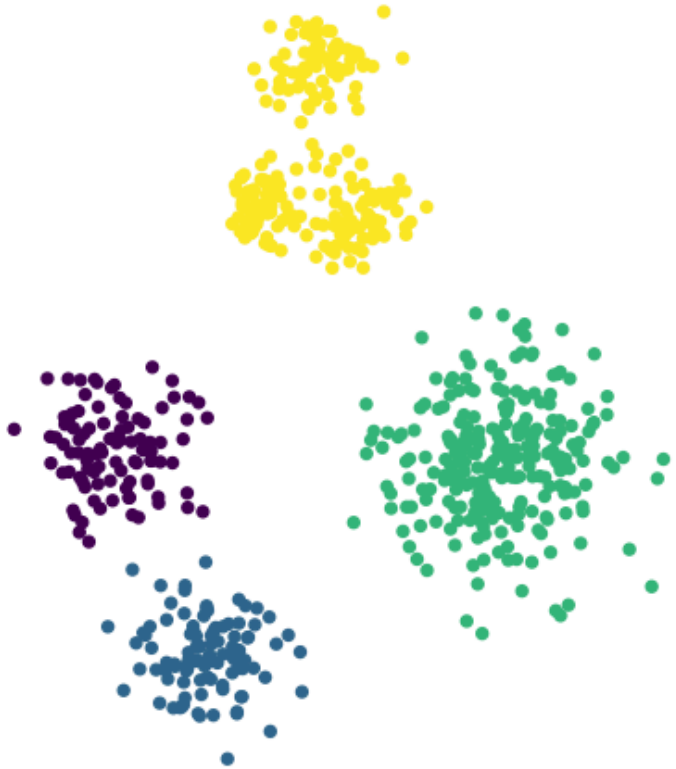
Three clusters?

▶ Code



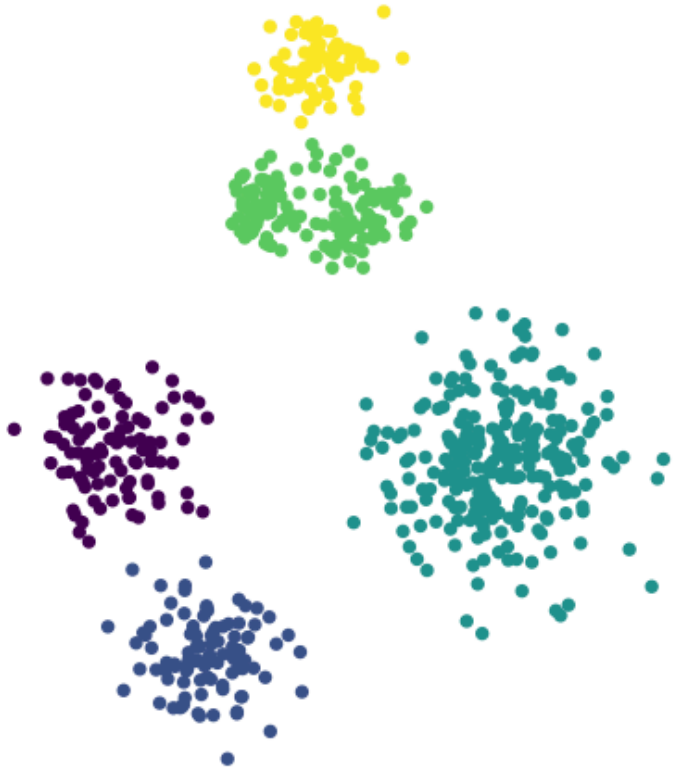
Four clusters?

▶ Code



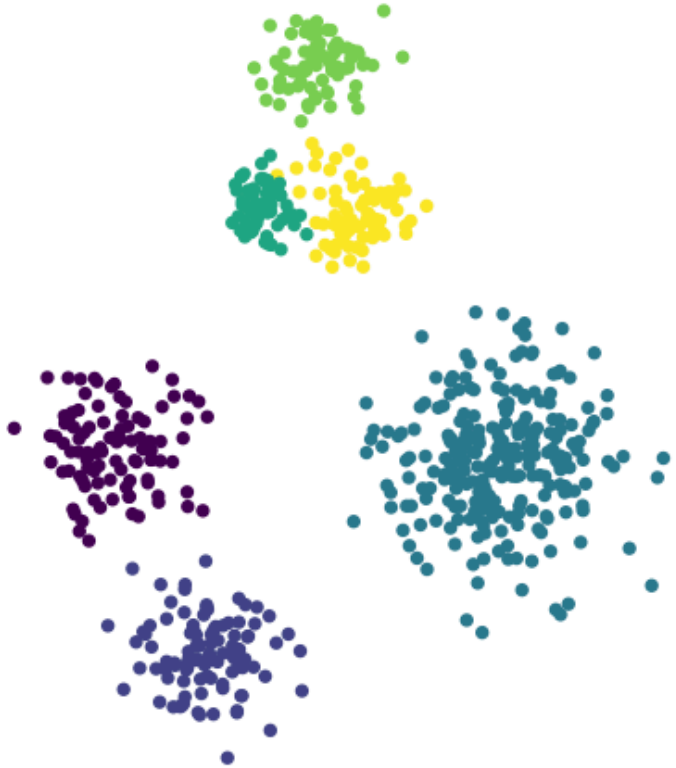
Five clusters?

▶ Code

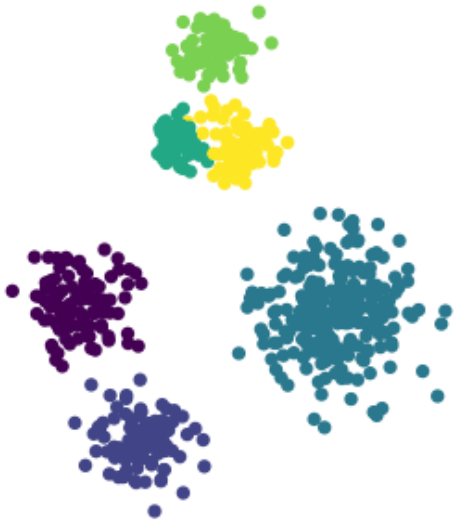


Six clusters?

▶ Code



▶ Code



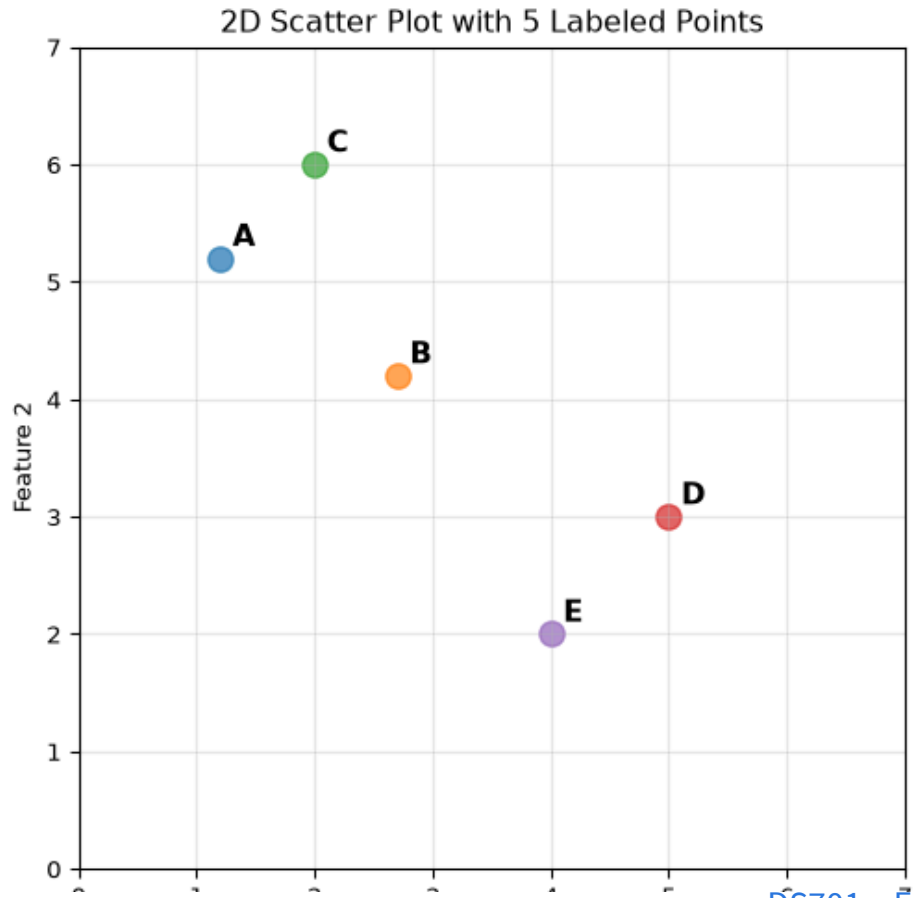
This dataset shows clustering on **multiple scales.**

Hierarchical Clustering Example

Hierarchical Clustering Example

Inspired by Hastie & Tibshirani lecture.

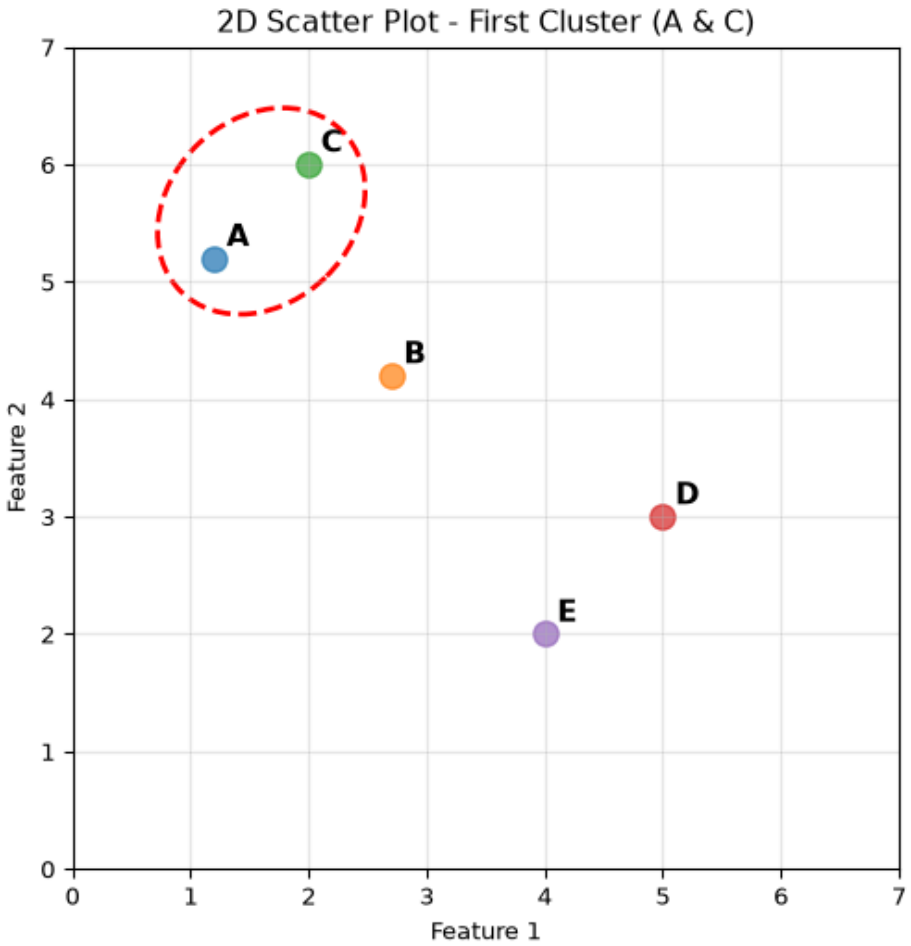
▶ Code



- Goal: Combine closest points/clusters into a new cluster
- Use distance metric to define closeness
- So-called bottom-up approach or *agglomerative* clustering
- Which 2 points are closest?

Example, cont.

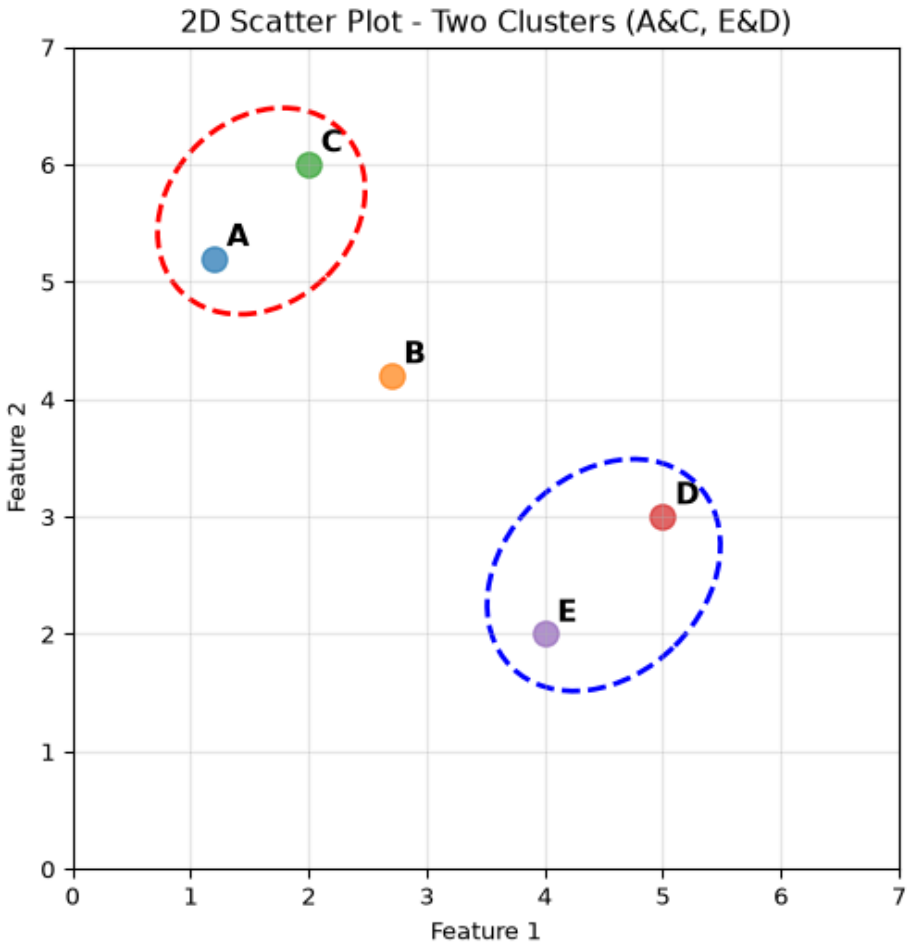
▶ Code



- Now we want to combine either the next closest pair of points or the point closest to a cluster
- How do we measure closeness of points to clusters?
- Or clusters to clusters?

Example, cont. (Part 2)

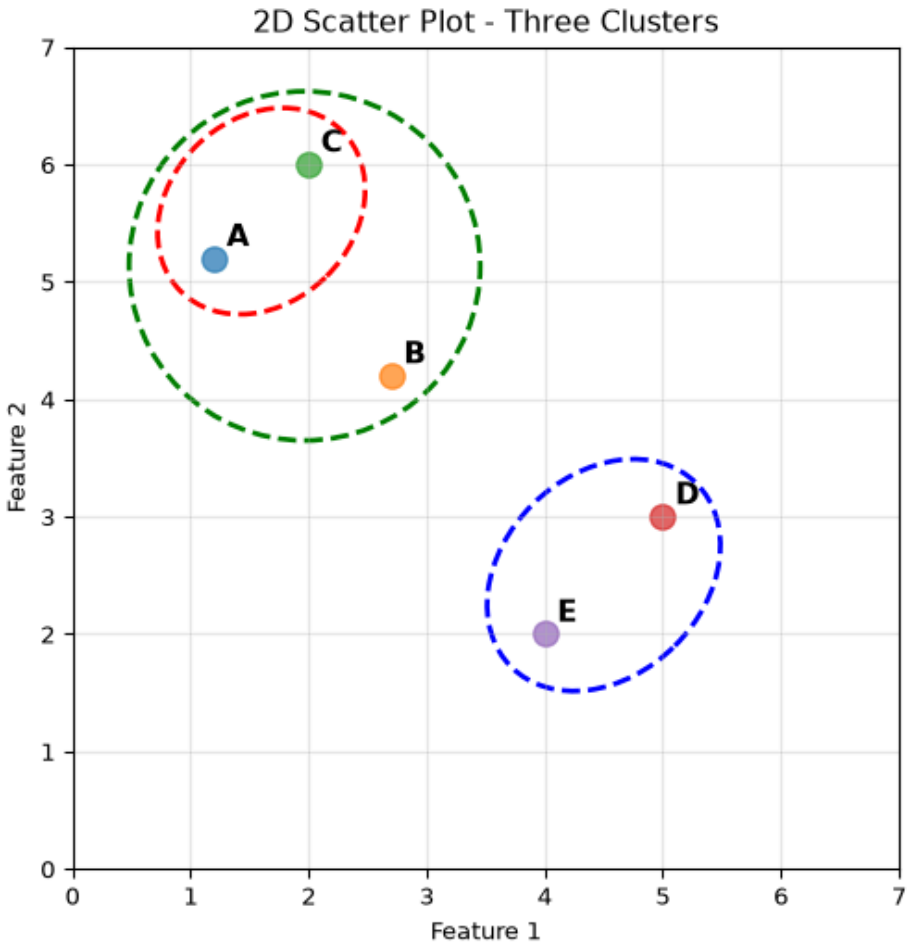
▶ Code



- Assume D and E are the next closest pair of points
- Now which cluster is B closest to?

Example, cont. (Part 3)

▶ Code

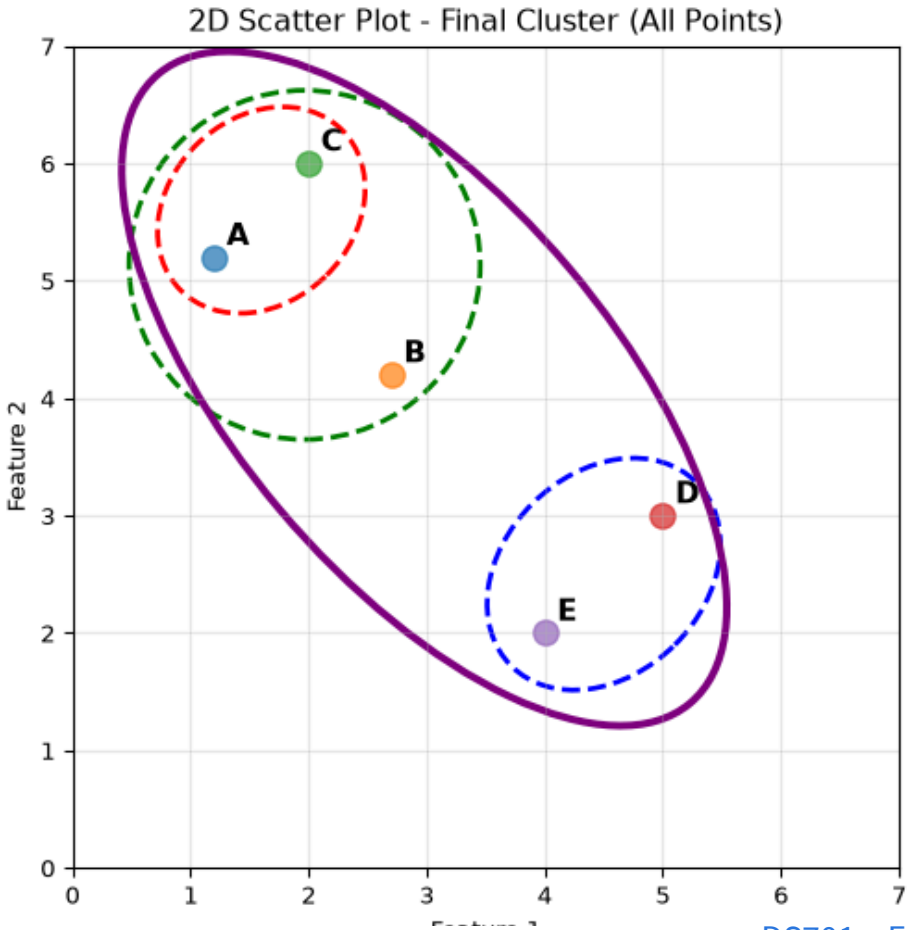


- We'll talk about measuring distances between clusters shortly
- But let's just say that B is closest to the cluster A, C, E and forms a new cluster
- and then finally…

Example, cont. (Part 4)

▶ Code

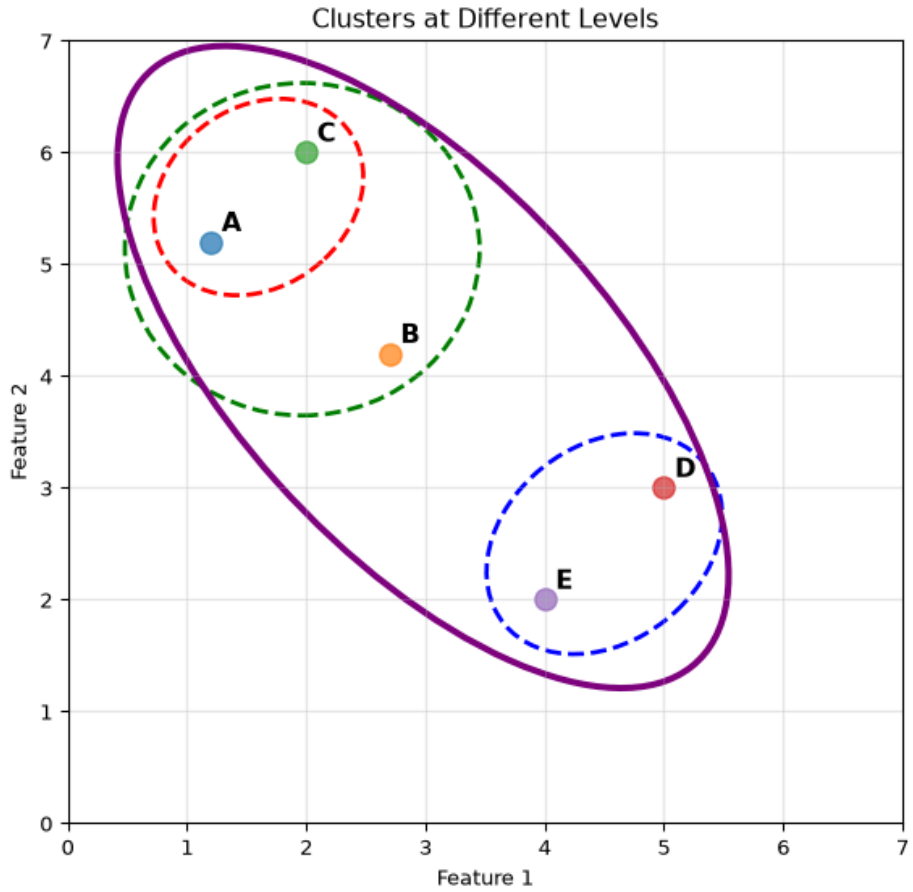
```
3.2 7 40 2.98 4.08
```

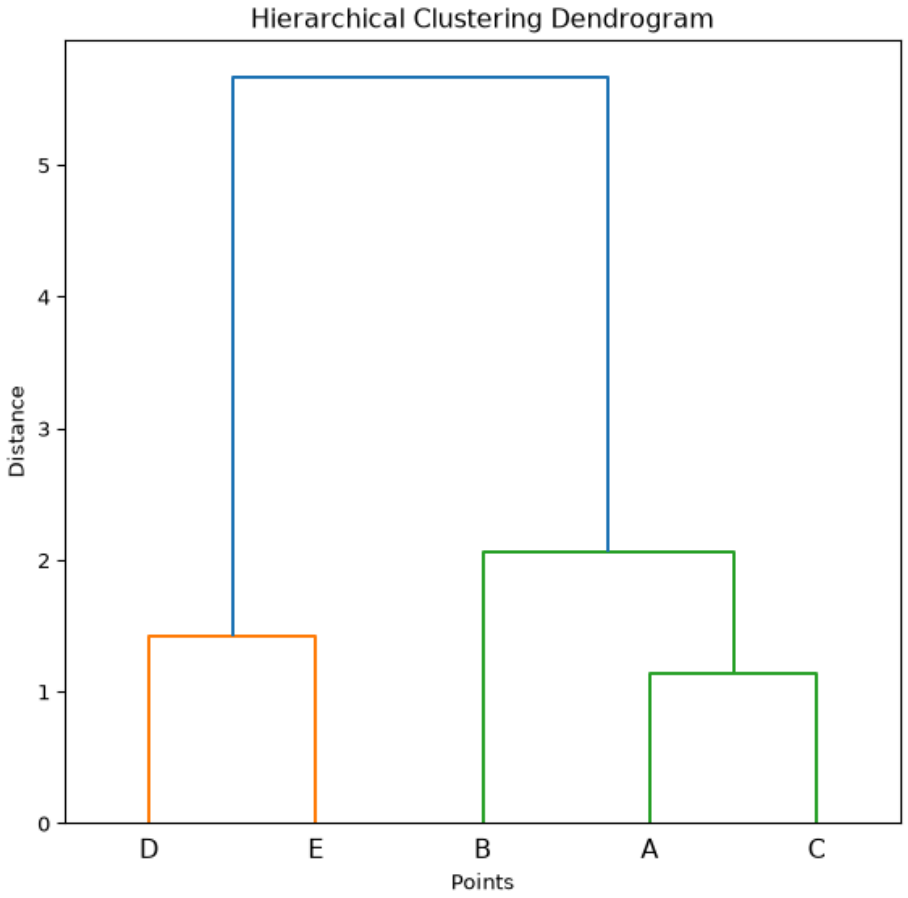


- We combine all points into a single cluster

Example: Dendrogram

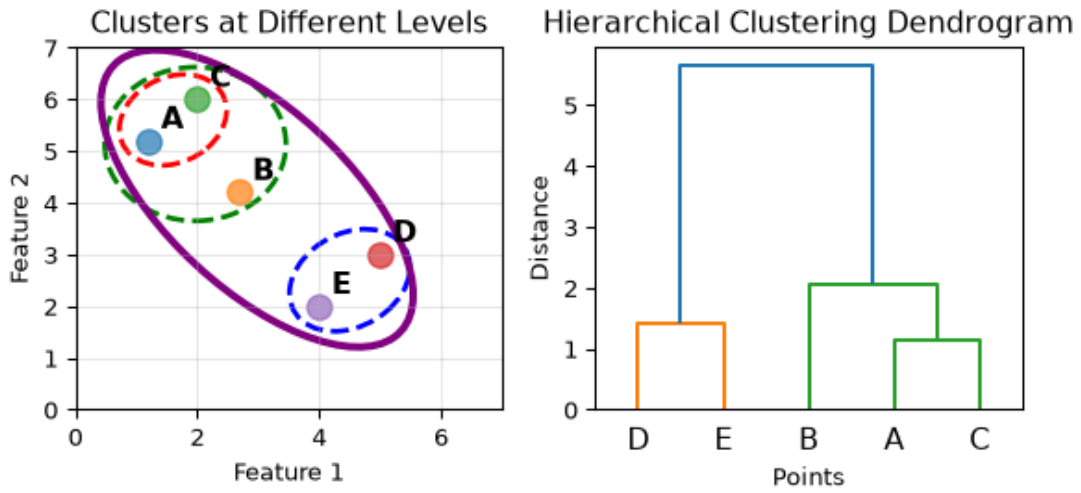
- Code





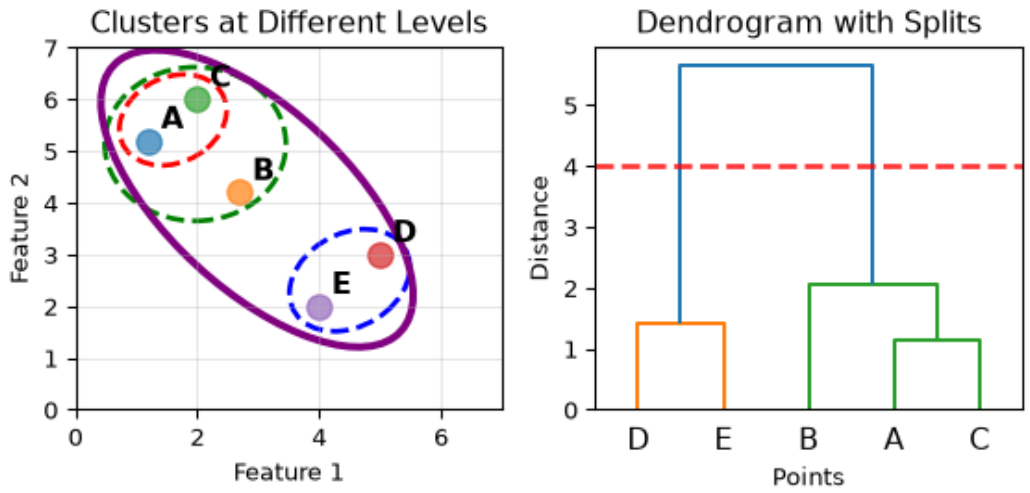
Example: Dendrogram Explained

- Code



Example: Dendrogram cont.

▶ Code



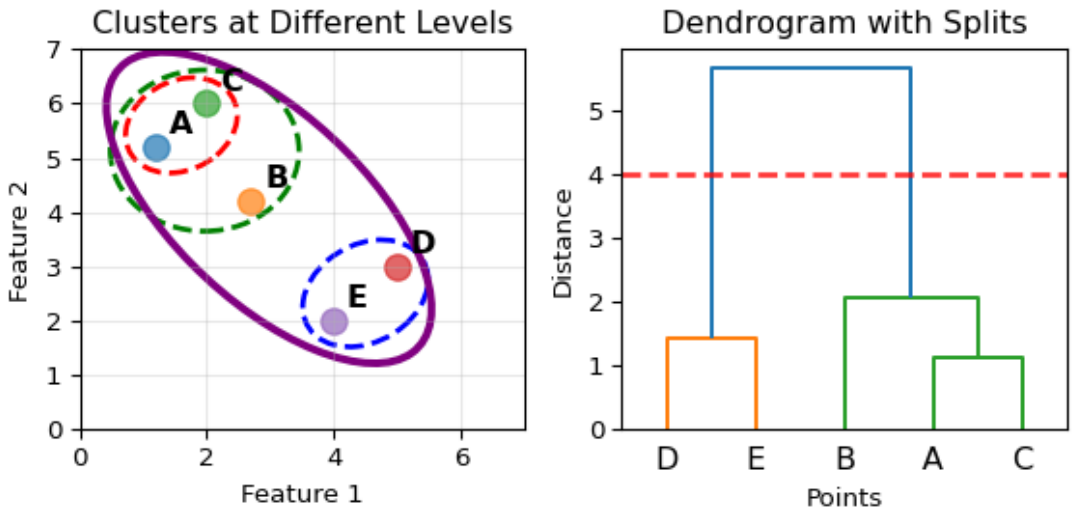
- We can now split the dendrogram at a certain level to get a clustering

Hierarchical Clustering

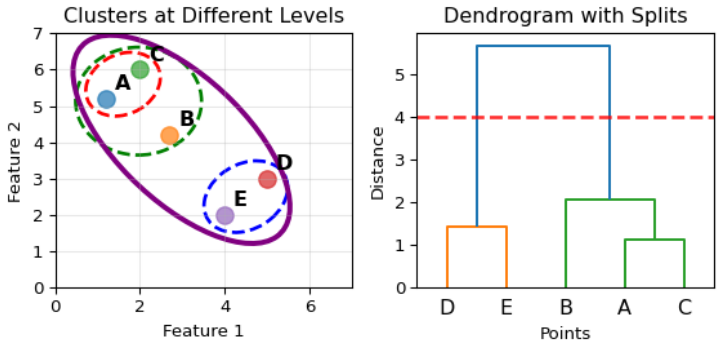
A hierarchical clustering produces a set of **nested** clusters organized into a tree.

A hierarchical clustering is visualized using a **dendrogram**

- A tree-like diagram that records the containment relations among clusters.



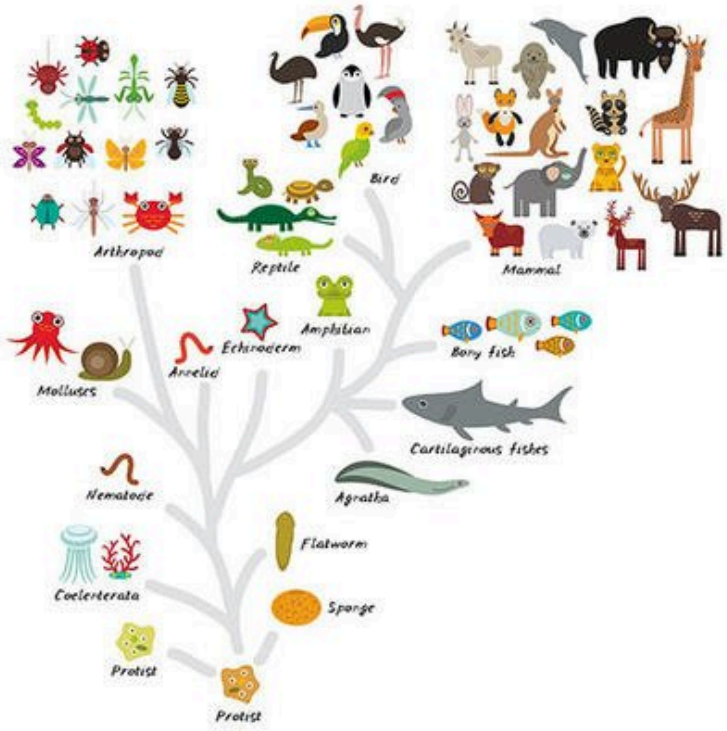
Strengths of Hierarchical Clustering



Hierarchical clustering has a number of advantages:

Another advantage

Another advantage is that the dendrogram may itself correspond to a meaningful structure, for example, a taxonomy.



Yet another advantage

- Many hierarchical clustering methods can be performed using either similarity (proximity) or dissimilarity (distance) metrics.
- This can be very helpful!
- Techniques like k-means rely on a specific way of measuring similarity or distance between data points.

Compared to k-means

Another aspect of hierachical clustering is that it can handle certain cases better than k-means.

Because of the nature of the k-means algorithm, k-means tends to produce:

- Roughly spherical clusters
- Clusters of approximately equal size
- Non-overlapping clusters

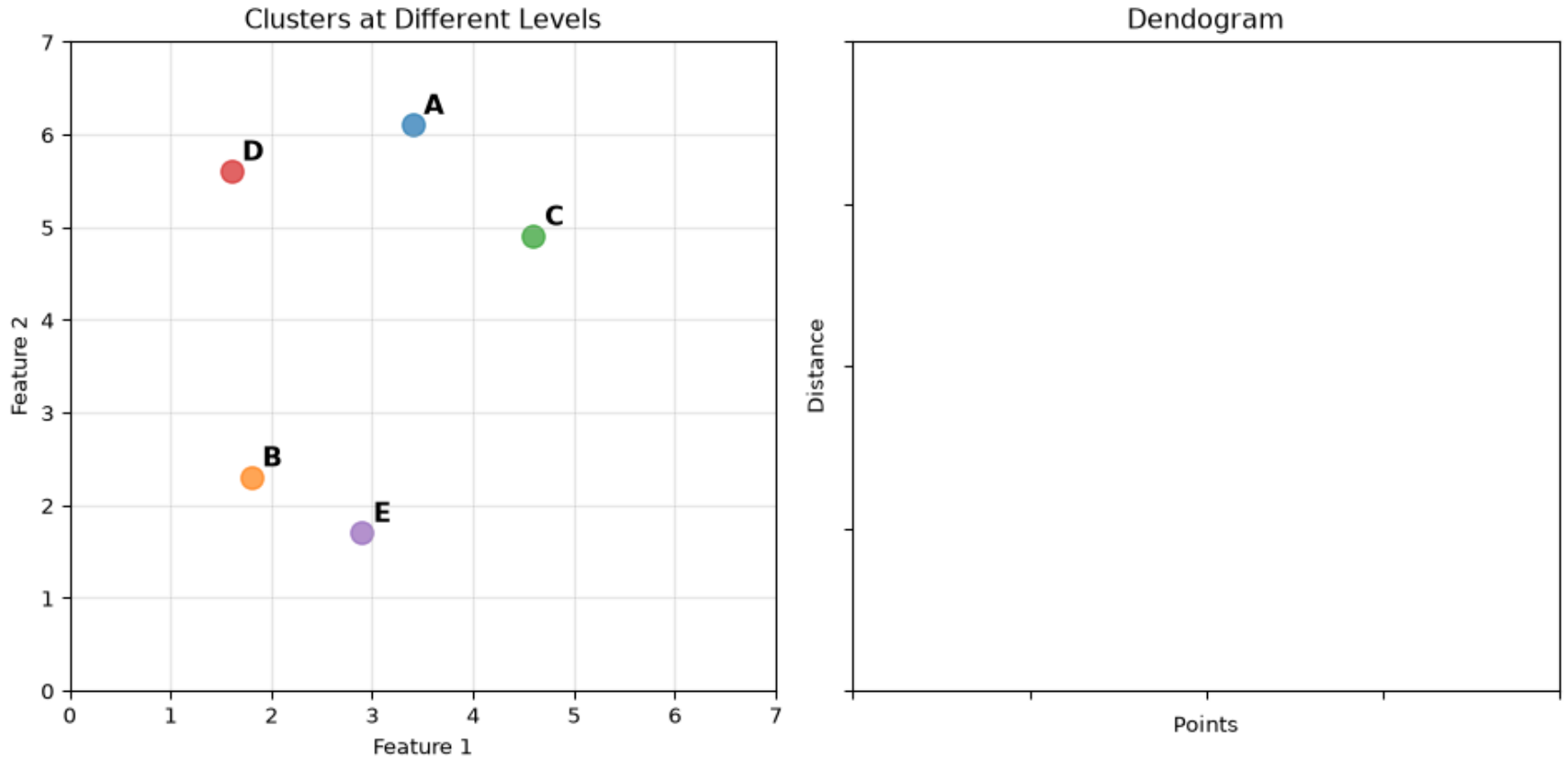
In many real-world situations, clusters may not be round, they may be of unequal size, and they may overlap.

Hence we would like clustering algorithms that can work in those cases also.

In-Class Exercise

Create a hierarchical cluster and dendogram for this data.

▶ Code



Hierarchical Clustering Algorithms

Hierarchical Clustering Algorithms

There are two main approaches to hierarchical clustering: “bottom-up” and “top-down.”

Some key points

- Agglomerative techniques are by far the more common.
- The key to both of these methods is defining **the distance between two clusters.**
- Different definitions for the inter-cluster distance yield different clusterings.

Hierarchical Clustering Algorithm Inputs

1. **Input data** as either
    - **2-D sample/feature matrix**
    - **1D condensed distance matrix** – upper triangular of pairwise distance matrix flattened
2. The cluster **linkage** method (**single**, complete, average, ward, …)
3. The **distance metric** (*Euclidean*, manhattan, Jaccard, …)

Hierarchical Clustering Algorithm Output

An $(n-1, 4)$ shaped **linkage matrix**.

Where **each row** is `[idx1, idx2, dist, sample_count]` is a single step in the clustering process and

`idx1` and `idx2` are indices of the cluster being merged, where

- indices $\{0 \ldots n-1\}$ refer to the original data points and
- indices $\{n, n+1, \ldots\}$ refer to each new cluster created

And `sample_count` is the number of original data samples in the new cluster.

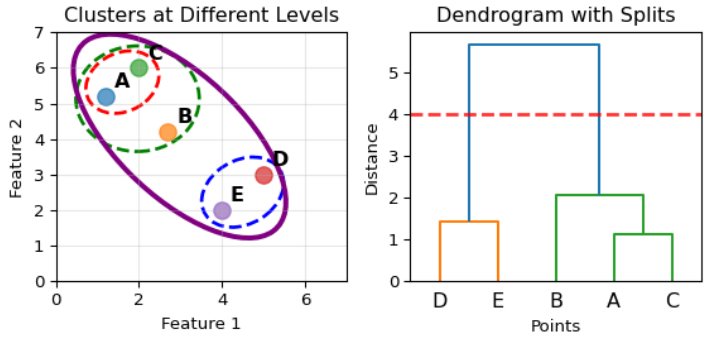
Linkage Matrix Example

Let’s calculate and display the linkage matrix for our 5 data points A-E:

▶ Code

```
Linkage Matrix for 5 data points A-E:
Format: [idx1, idx2, distance, sample_count]
==================================================
[[0.         2.         1.13137085 2.        ]
 [3.         4.         1.41421356 2.        ]
 [1.         5.         2.05588586 3.        ]
 [6.         7.         5.66085977 5.        ]]
==================================================

Interpretation:
Row 0: Merge points 0 and 2 (A and C) at distance 1.13
Row 1: Merge points 3 and 4 (D and E) at distance 1.41
Row 2: Merge point 1 (B) with cluster 5 (A&C) at distance 2.06
Row 3: Merge cluster 6 (A&B&C) with cluster 7 (D&E) at distance
5.66
```



Hierarchical Clustering Algorithm Explained

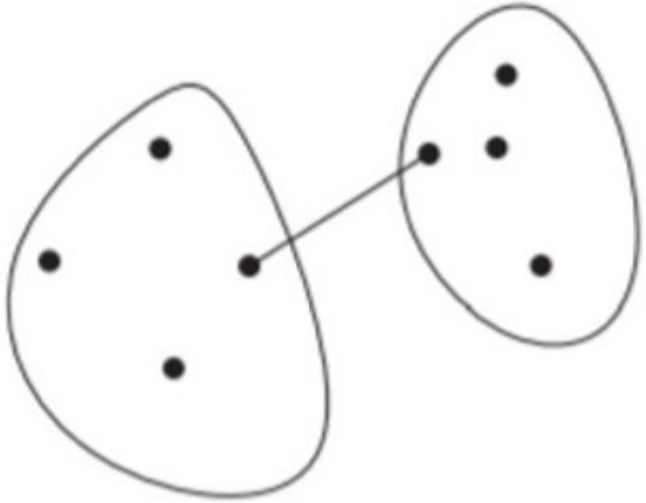
1. **Initialization**
    - Start with each data point as its own cluster.
    - Calculate the distance matrix if not provided
2. **Iterative Clustering**
    - At each step, the two closest clusters (according to linkage method and distance metric) are merged into a new cluster.
    - The distance between new cluster and the remaining clusters is added
    - Repeat previous two steps until only one cluster remains.

Defining Cluster Proximity

Given two clusters, how do we define the *distance* between them?

Here are three natural ways to do it.

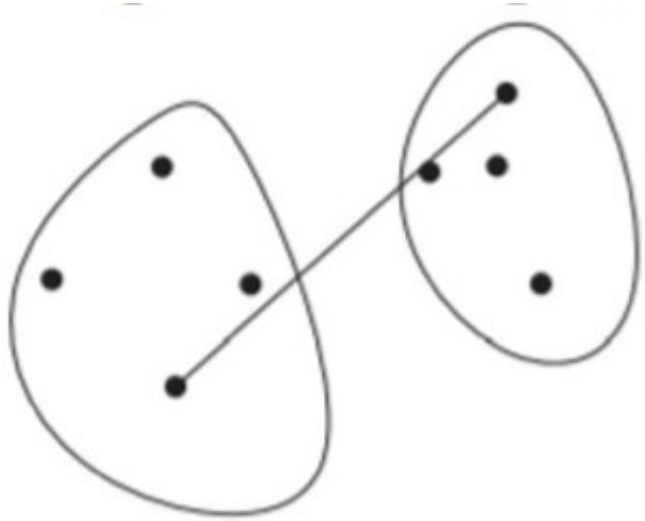
Cluster Proximity – Single-Linkage



Single-Linkage: the distance between two clusters is the distance between the closest two points that are in different clusters.

$$D_{\text{single}}(i,j) = \min_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

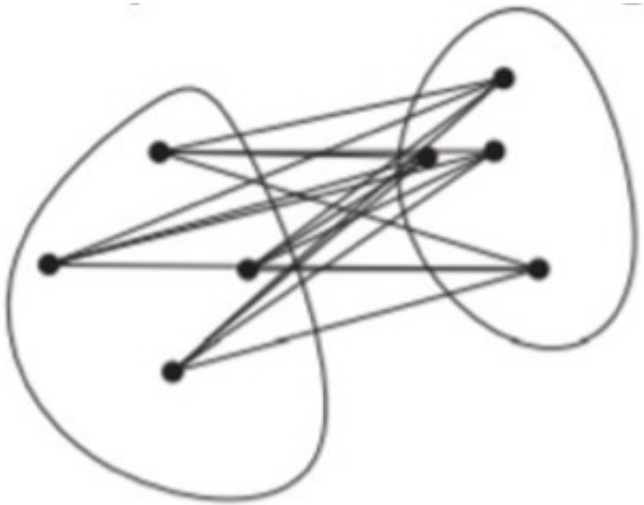
Cluster Proximity – Complete-Linkage



Complete-Linkage: the distance between two clusters is the distance between the farthest two points that are in different clusters.

$$D_{\text{complete}}(i,j) = \max_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

Cluster Proximity – Average-Linkage



Average-Linkage: the distance between two clusters is the average distance between all pairs of points from different clusters.

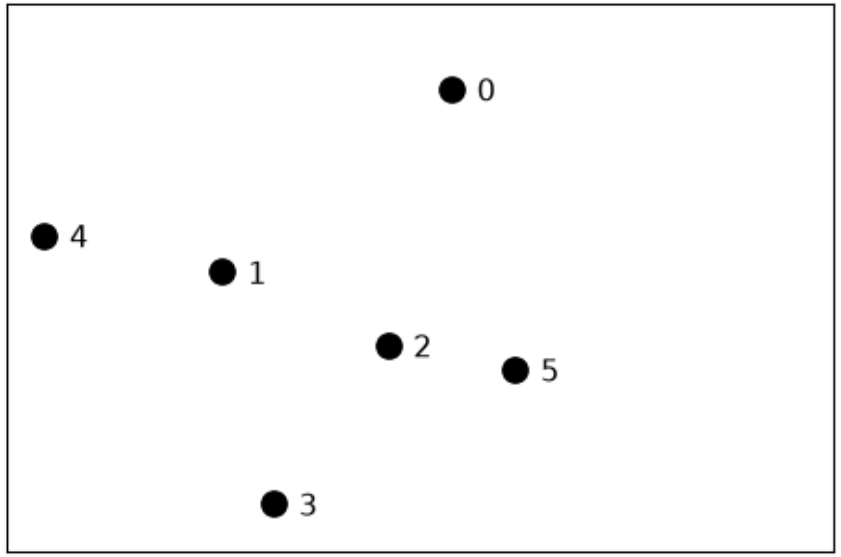
$$D_{\text{average}}(i,j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{x \in C_i,\, y \in C_j} d(x,y)$$

Cluster Proximity Example

Notice that it is easy to express the definitions above in terms of similarity instead of distance.

Here is a set of 6 points that we will cluster to show differences between distance metrics.

▶ Code



Cluster Proximity Example, cont.

We can calculate the distance matrix

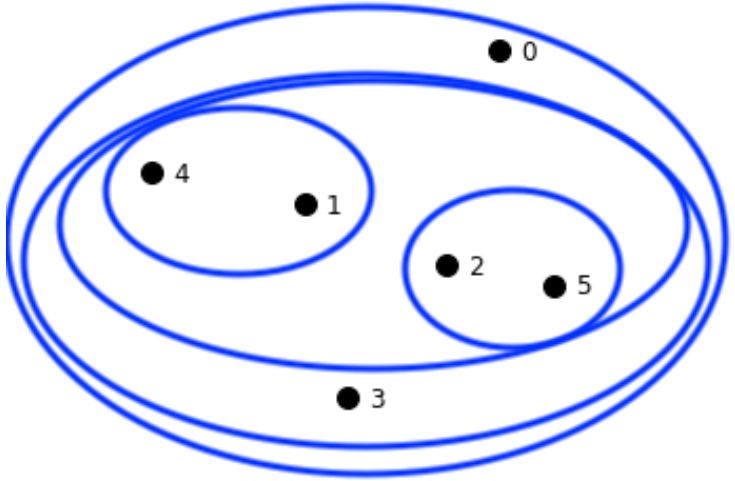
▶ Code

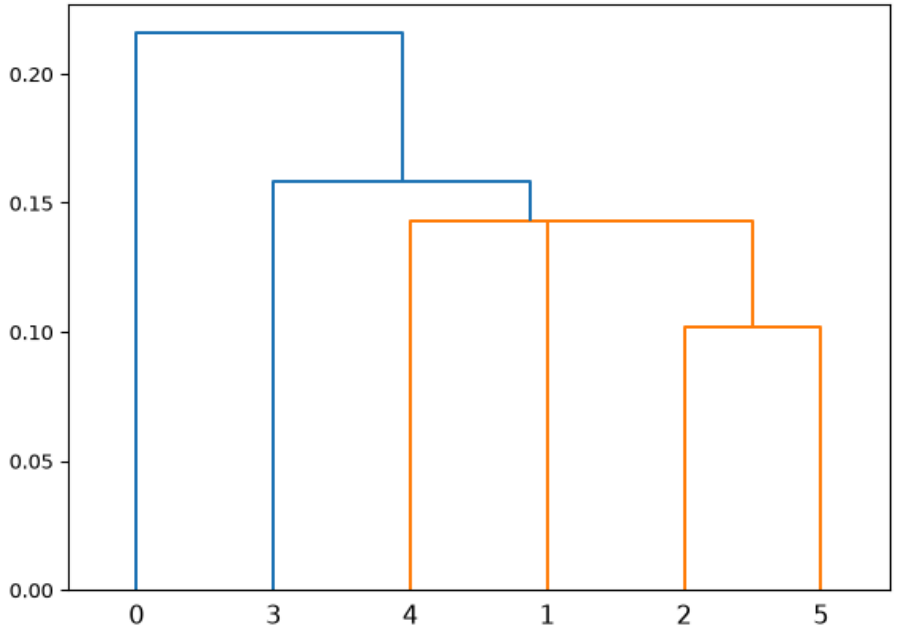
```
/var/folders/j_/hxxgy5dd7655k_416s6t9kgc0000gq/T/ipykernel_5598/1916159444.py:4: DeprecationWarning:
`distance_matrix` is deprecated in favor of `scipy.spatial.distance.cdist` as of SciPy 1.18.0 and will be
removed in SciPy 1.20.0.
  D = pd.DataFrame(distance_matrix(X, X), index = labels, columns = labels)
/var/folders/j_/hxxgy5dd7655k_416s6t9kgc0000gq/T/ipykernel_5598/1916159444.py:4: DeprecationWarning:
`minkowski_distance` is deprecated in favor of `scipy.spatial.distance.minkowski` as of SciPy 1.18.0 and will be
removed in SciPy 1.20.0.
  D = pd.DataFrame(distance_matrix(X, X), index = labels, columns = labels)
/var/folders/j_/hxxgy5dd7655k_416s6t9kgc0000gq/T/ipykernel_5598/1916159444.py:4: DeprecationWarning:
`minkowski_distance_p` is deprecated in favor of `scipy.spatial.distance.minkowski` as of SciPy 1.18.0 and will
be removed in SciPy 1.20.0.
  D = pd.DataFrame(distance_matrix(X, X), index = labels, columns = labels)
```

	p0	p1	p2	p3	p4	p5
p0	0.00	0.23	0.22	0.37	0.34	0.24
p1	0.23	0.00	0.14	0.19	0.14	0.24
p2	0.22	0.14	0.00	0.16	0.28	0.10
p3	0.37	0.19	0.16	0.00	0.28	0.22

Single-Linkage Clustering

▶ Code



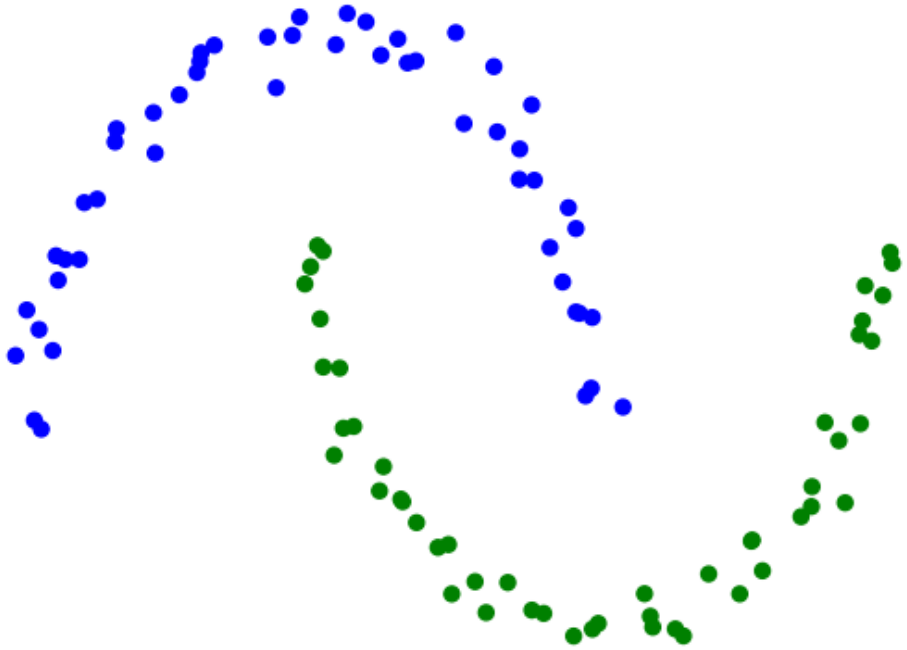


Single Linkage Clustering Advantages

Single-linkage clustering can handle non-elliptical shapes.

Here we use SciPy's fcluster to form flat custers from hierarchical clustering.

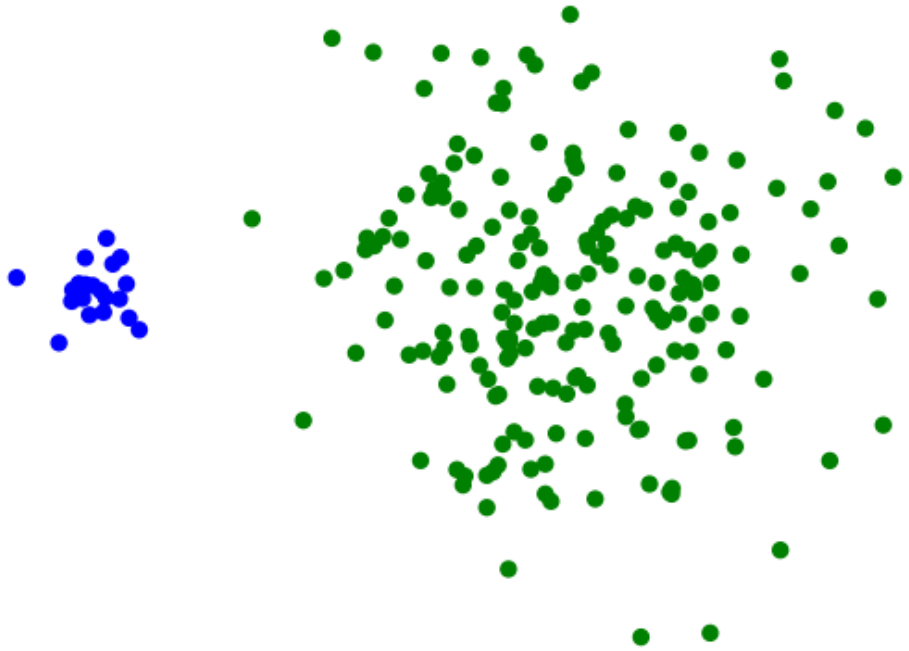
▶ Code



Single Linkage Advantages, cont.

Single-Linkage can find different sized clusters:

▶ Code

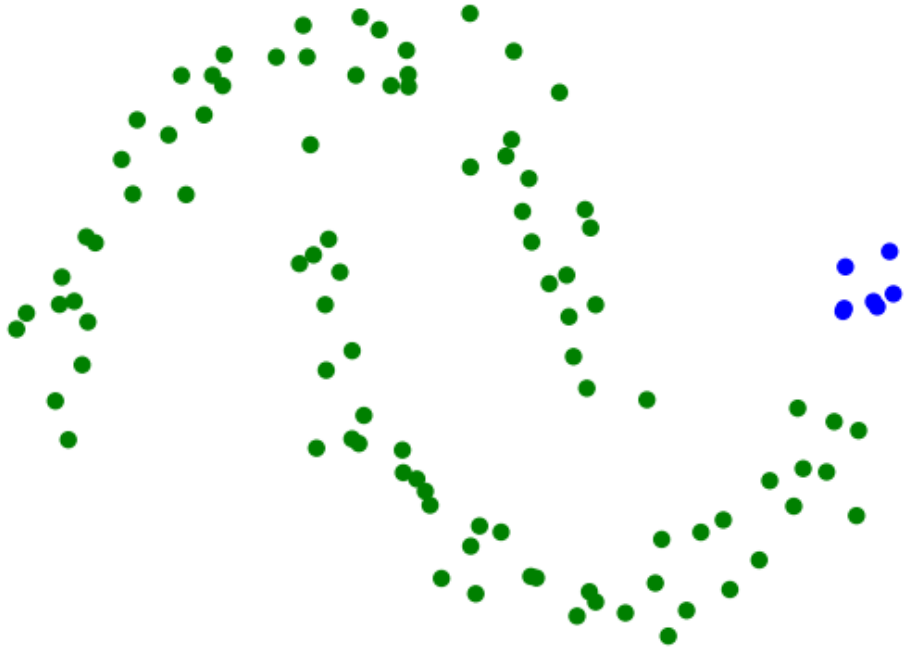


Single Linkage Disadvantages

Single-linkage clustering can be sensitive to noise and outliers.

The results can change drastically on even slightly more noisy data.

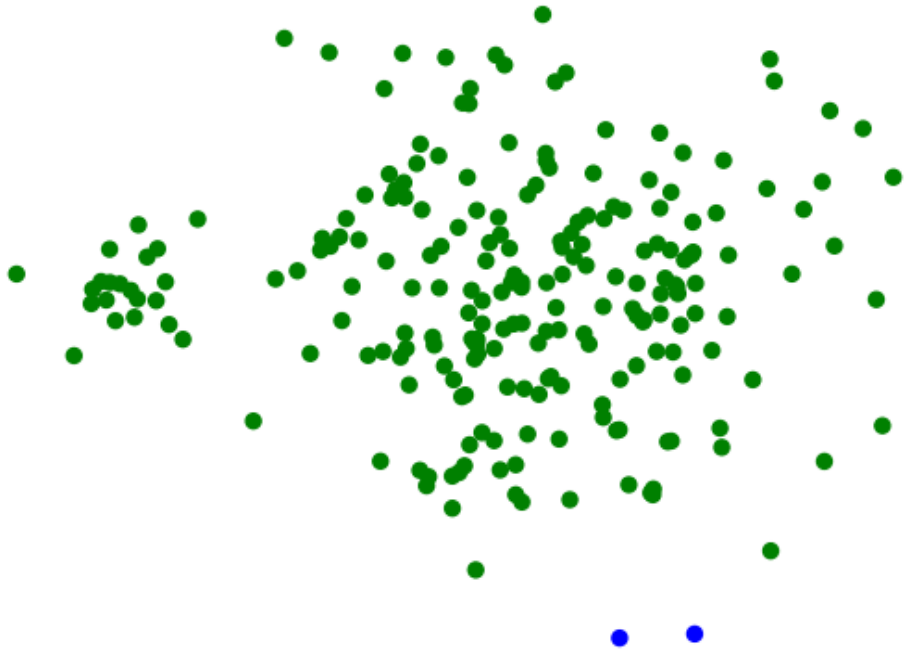
▶ Code



Single Linkage Disadvantages, cont.

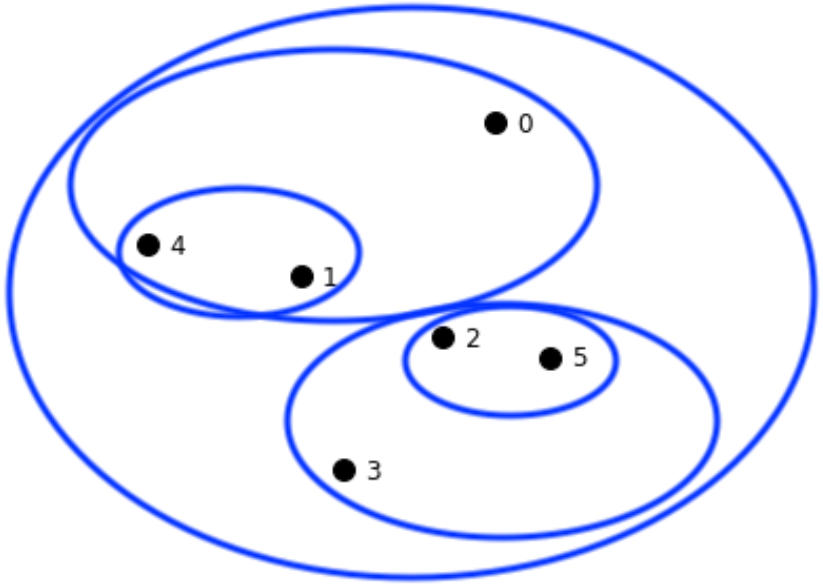
And here's another example where we bump the standard deviation on the clusters slightly.

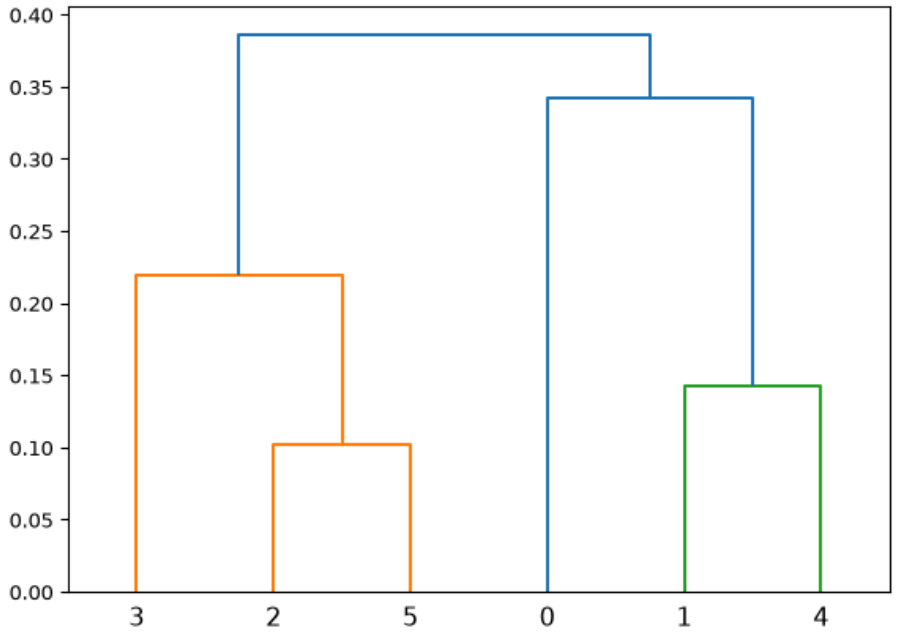
▶ Code



Complete-Linkage Clustering

▶ Code

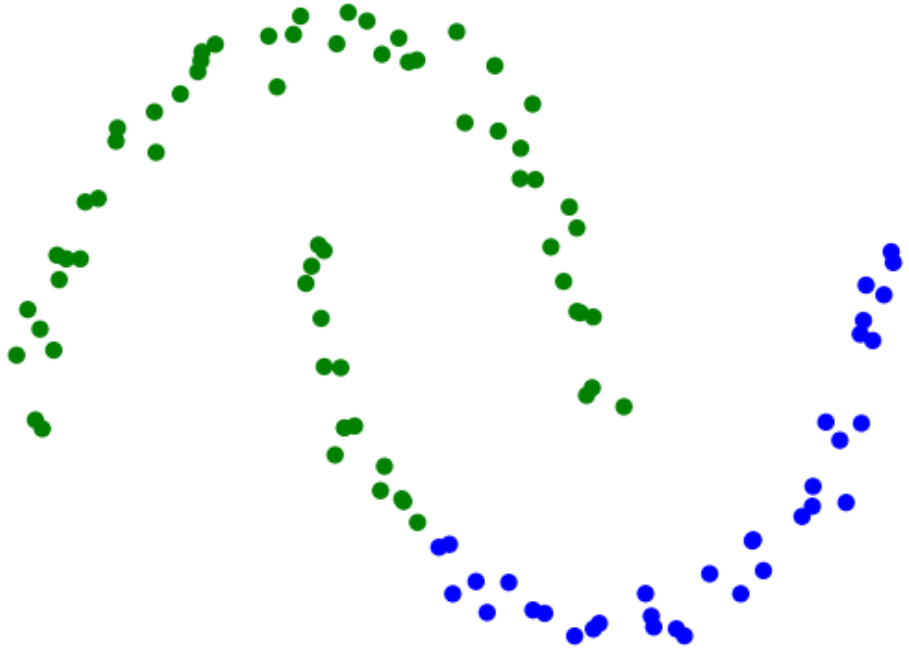




Complete-Linkage Clustering Advantages

Produces more-balanced clusters – more-equal diameters

▶ Code



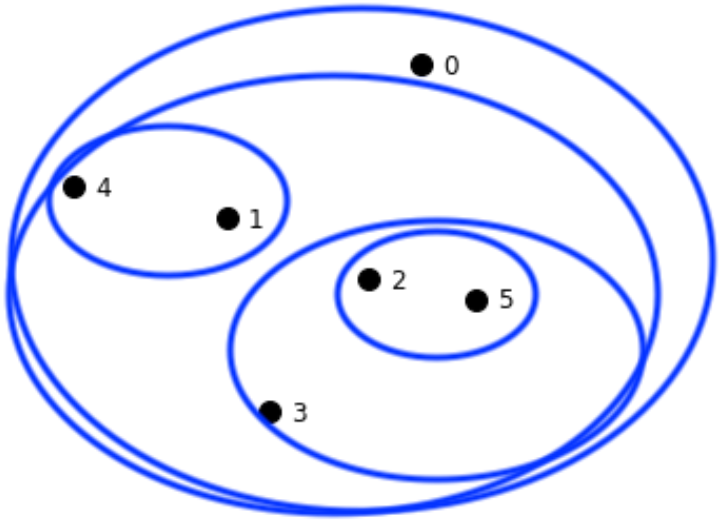
Complete-Linkage Clustering Disadvantages

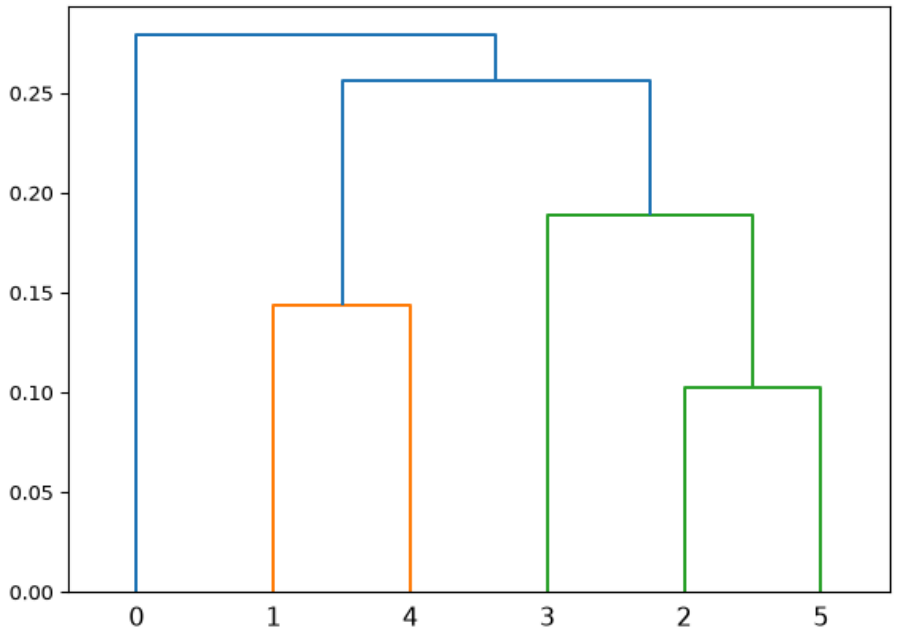
Some disadvantages for complete-linkage clustering are:

- Sensitivity to outliers
- Tendency to compute more compact, spherical clusters
- Computationally intensive for large datasets.

Average-Linkage Clustering

▶ Code





Average-Linkage Clustering Strengths and Limitations

Average-Linkage clustering is in some sense a compromise between Single-linkage and Complete-linkage clustering.

Strengths:

- Less susceptible to noise and outliers

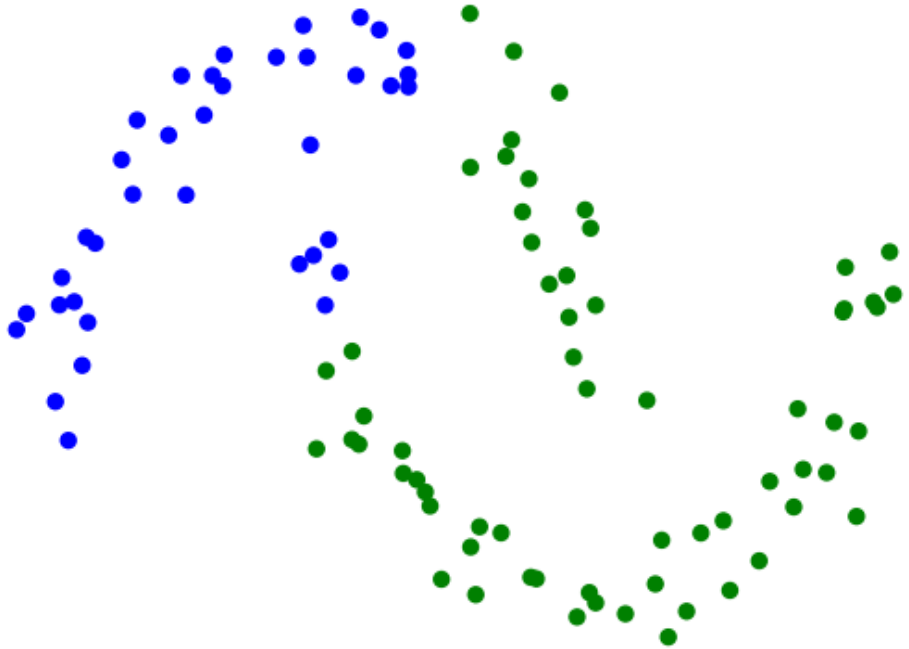
Limitations:

- Biased toward elliptical clusters

Average-Linkage Clustering Strengths and Limitations, cont.

Produces more isotropic clusters.

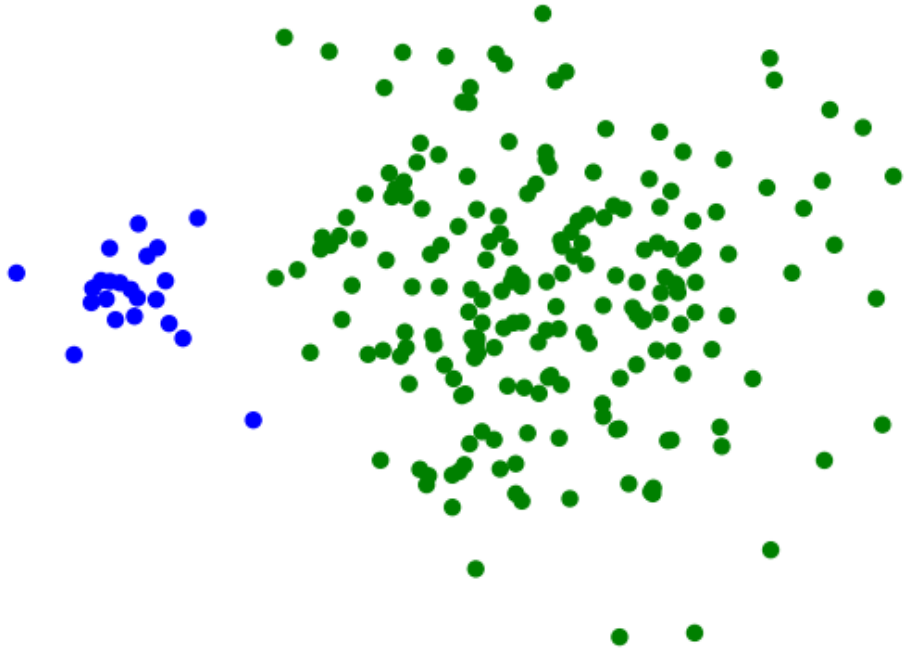
▶ Code



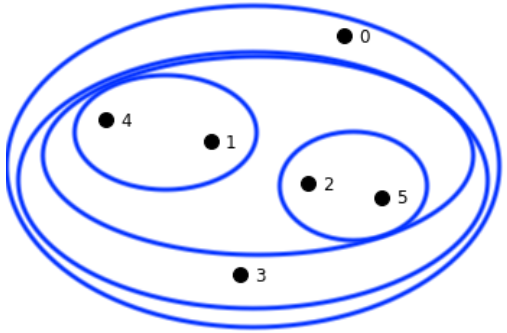
Average-Linkage Clustering Strengths and Limitations, cont.

More resistant to noise than Single-Linkage.

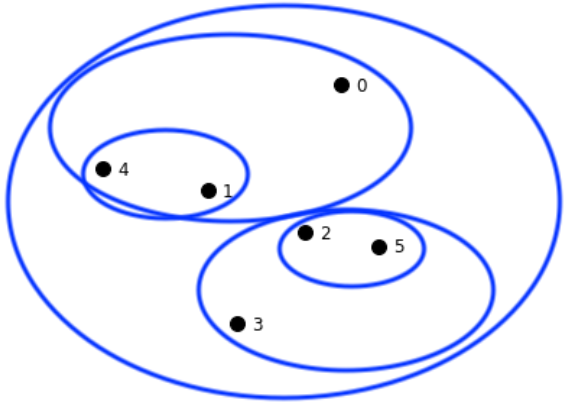
▶ Code



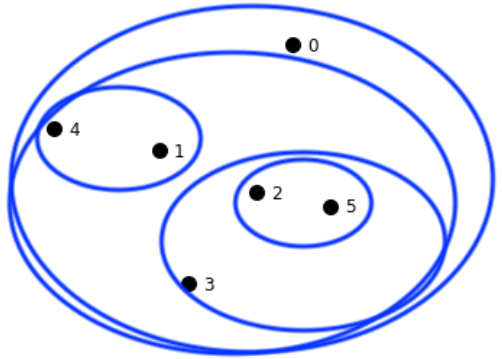
All Three Compared



Single-Linkage



Complete-Linkage



Average-Linkage

Ward's Distance

Finally, we consider one more cluster distance.

Ward's distance asks "What if we combined these two clusters – how would clustering improve?"

To define "how would clustering improve?" we appeal to the k-means criterion.

So:

Ward's Distance between clusters C_i and C_j is the difference between the total within cluster sum of squares for the two clusters separately, **compared to** the *within cluster sum of squares* resulting from merging the two clusters into a new cluster C_{i+j}:

$$D_{\text{Ward}}(i,j) = \sum_{x \in C_i} (x - c_i)^2 + \sum_{x \in C_j} (x - c_j)^2 - \sum_{x \in C_{i+j}} (x - c_{i+j})^2$$

where c_i, c_j, c_{i+j} are the corresponding cluster centroids.

Ward's Distance continued

In a sense, this cluster distance results in a hierarchical analog of k-means.

As a result, it has properties similar to k-means:

- Less susceptible to noise and outliers
- Biased toward elliptical clusters

Hence it tends to behave more like average-linkage hierarchical clustering.

Hierarchical Clustering in Practice

In Practice

Now we'll look at doing hierarchical clustering in practice.

We'll use 1/10th of the synthetic data set from the beginning of this lecture.

scipy has a comprehensive set of clustering tools.

▶ Code

In Practice, cont.

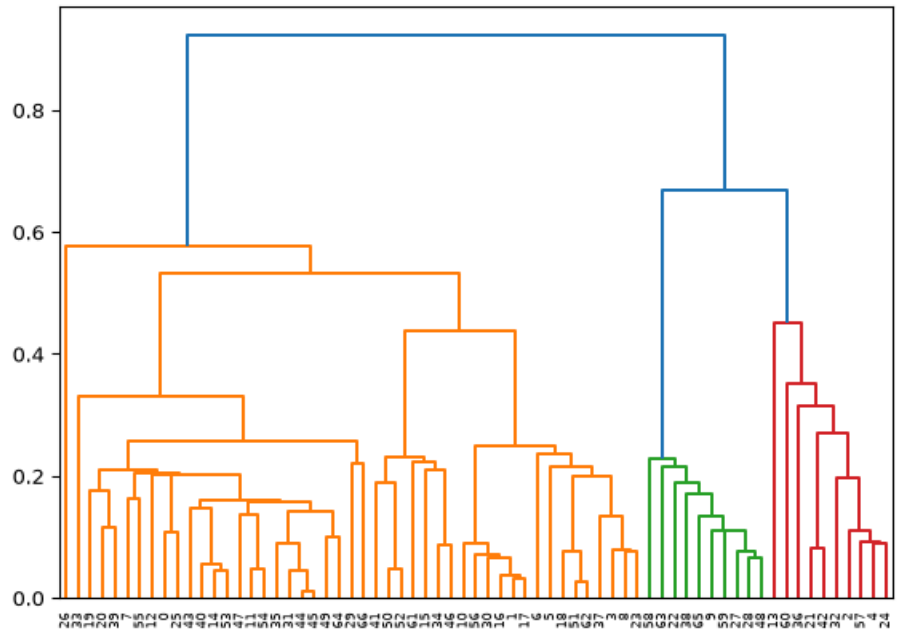
Let's run hierarchical clustering on our synthetic dataset.

And draw the dendrogram.

▶ Code

> Try the other linkage methods and see how the clustering and dendrogram changes.

```
# linkages = ['single','complete','average','weight
Z = hierarchy.linkage(X, method = 'single')
```



Hierarchical Clustering – Newsgroups

Once again we'll use the "20 Newsgroup" data provided as example data in sklearn. Experiment with different metrics.

Load three of the newsgroups.

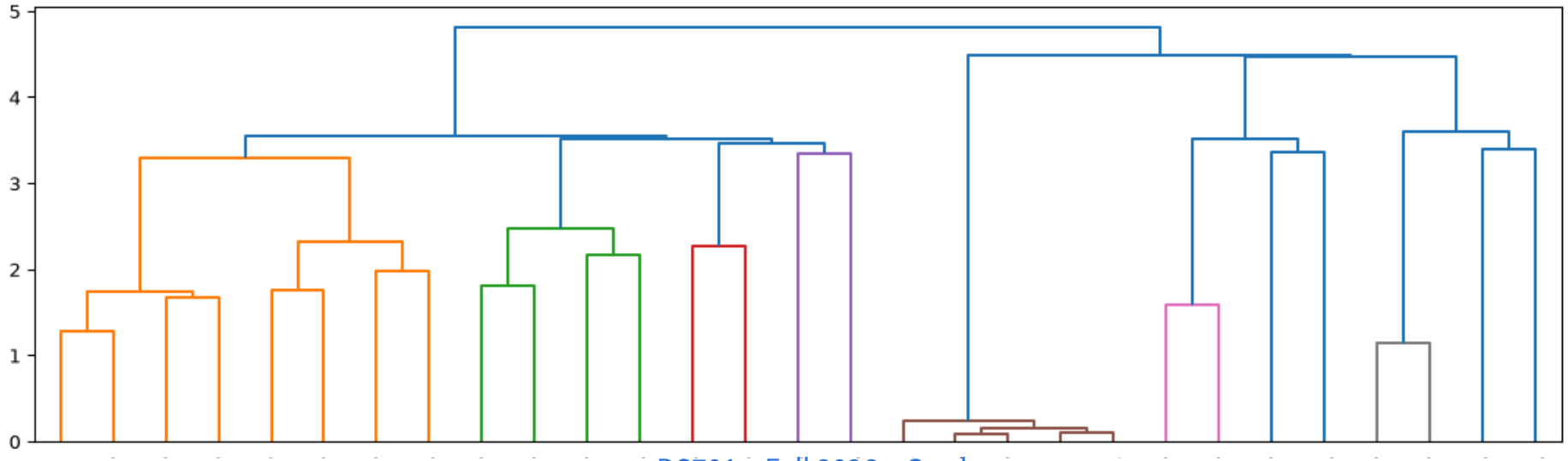
▶ Code

▶ Code

Vectorize the data.

▶ Code

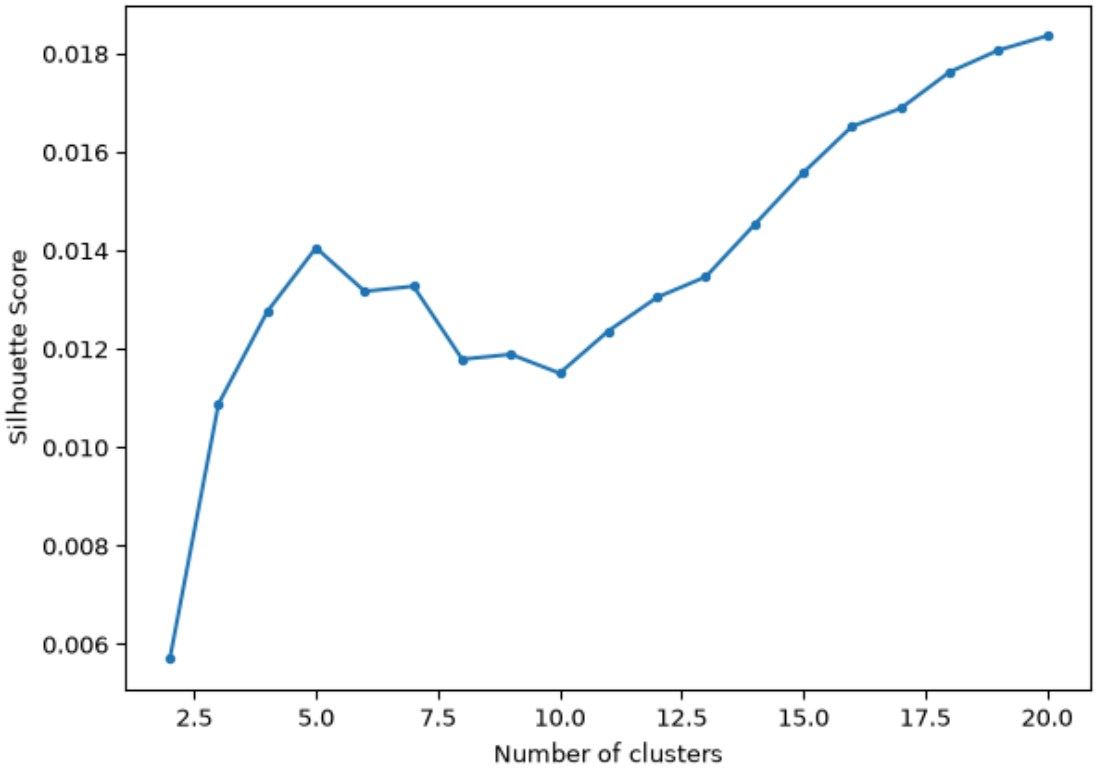
```
(1781, 9409)
```



Selecting the Number of Clusters

Let’s flatten the hierarchy to different numbers clusters and calculate the *Silhouette Score*.

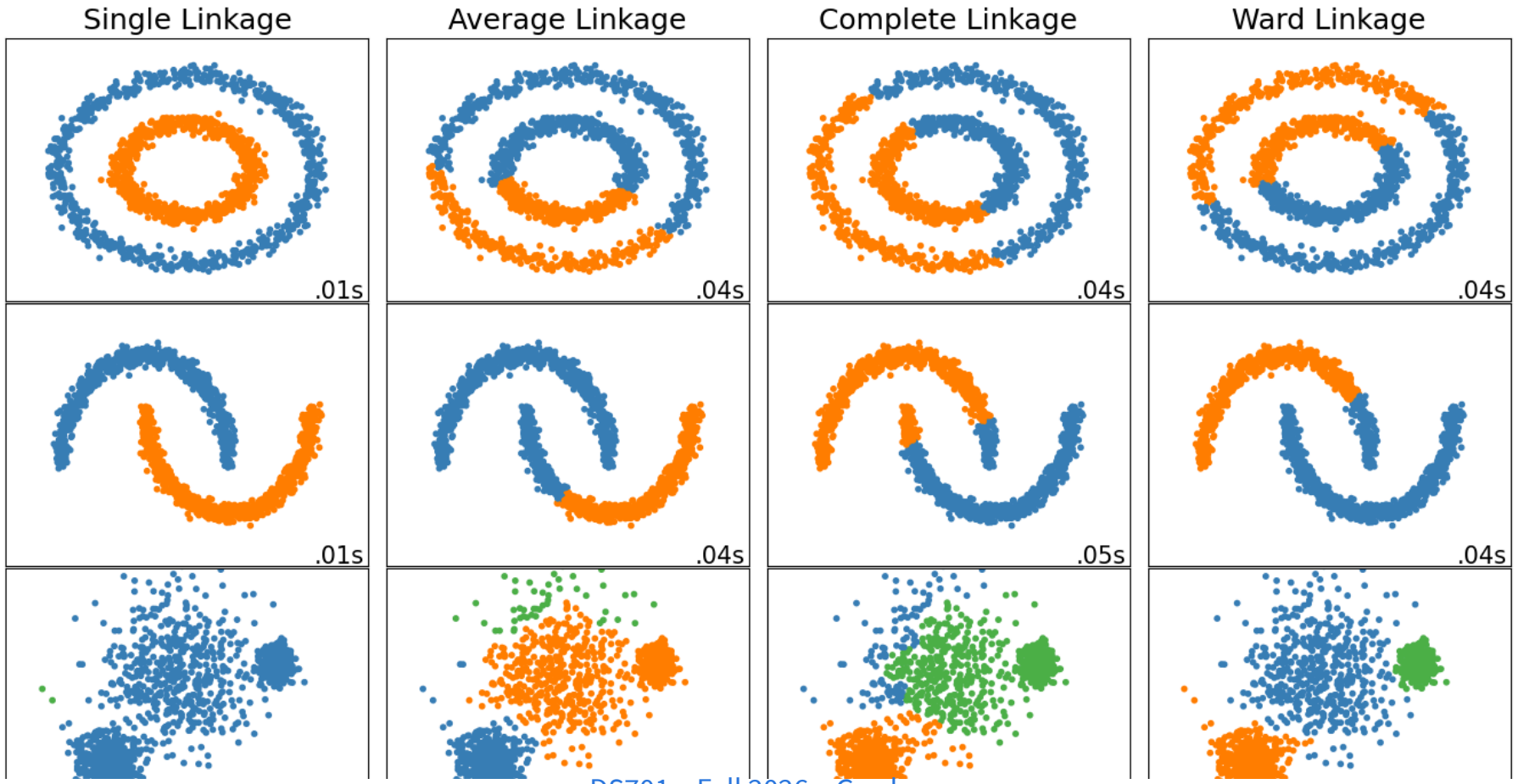
▶ Code



We see a first peak at 5

Comparison of Linkages

Scikit-Learn has a very nice notebook and plot, copied here, that shows the different clusters resulting from different linkage methods.



Recap

Clustering Recap

This wraps up our *partitional* Cluster topics. We covered: