# Hierarchical Agglomerative Clustering

CHIEN CHIN CHEN

# Preface

Different to flat clustering, ***hierarchical clustering*** outputs a ***hierarchy***.

- A structure (**tree**) that is supposed to be more informative than the unstructured set of clusters in flat clustering.
- Some researchers believe that hierarchical clustering produces better clusters than flat clustering But … there is no consensus on this issue.

This chapter focuses on ***hierarchical agglomerative clustering*** (HAC).

- Present five different linkage measures: *single-link*, *complete-link*, *group-average*, *centroid similarity*, and *Ward*.
- Clustering result is **deterministic**!!

# Hierarchical Agglomerative Clustering (1/6)

A **bottom-up** approach.

Treats each document as a singleton cluster at the outset.

**Successively merge** (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster.

An HAC clustering is typically visualized as a ***dendrogram***.

# Hierarchical Agglomerative Clustering (2/6)

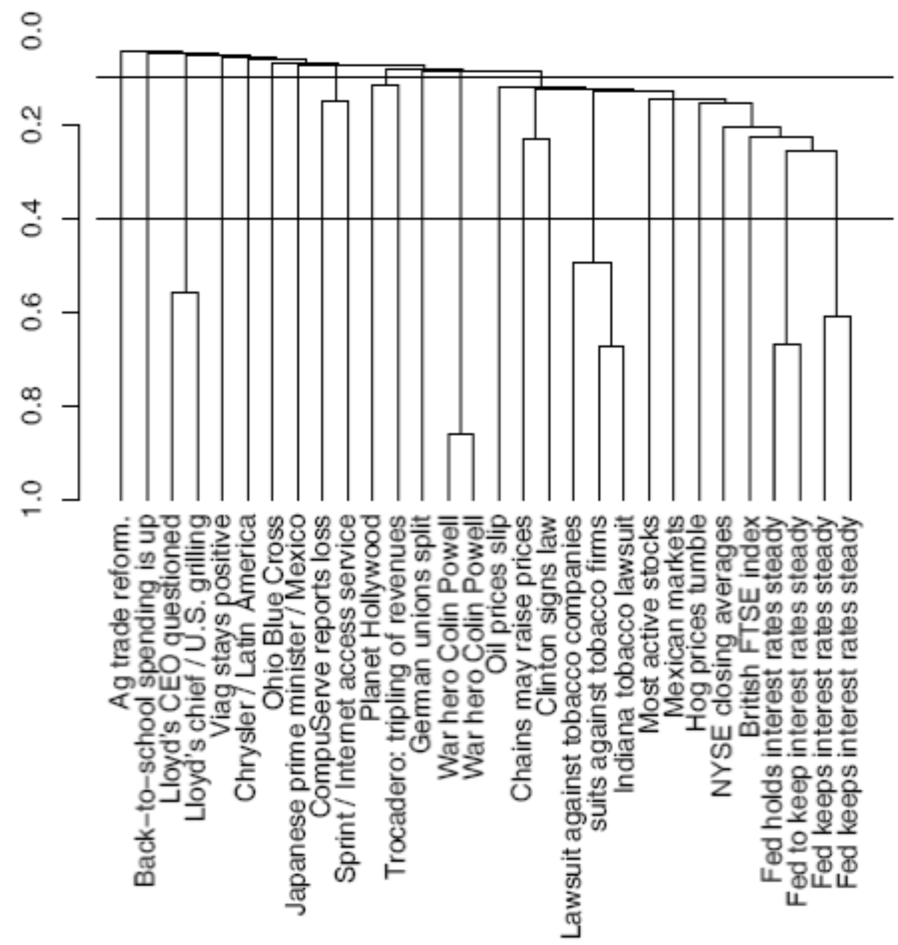
A merge of two clusters is represented as a horizontal line.

The y-axis represents <u>combination similarity</u> (or distance).

- The similarity of the two clusters connected by the horizontal line.

The y-axis at leaf nodes is 1.

- To represent each cluster as a document.



# Hierarchical Agglomerative Clustering (3/6)

```
SimpleHAC (d_1, ..., d_N)

  for n ← 1 to N
  do for i ← 1 to N
     do C[n][i] ← Sim(d_n, d_i)
     I[n] ← 1
  A ← []

  for k ← 1 to N-1
    do <i,m> ← argmax_{<i,m>: i≠m and I[i]=1 and I[m]=1} C[i][m]
       A.Append(<i,m>)
       for j ← 1 to N
       do C[i][j] ← Sim(j,i,m)
          C[j][i] ← Sim(j,i,m)
       I[m] ← 0

  return A
```

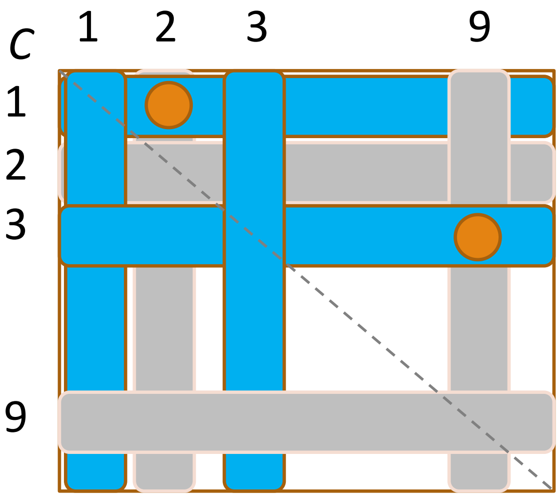
*C*[*i*][*j*]: the similarity between clusters *i* and *j*.

*I*: indicate which clusters are still available to be merged.

*A*: a list of merges

the similarity of cluster *j* with the merge of cluster *i* and *m*.

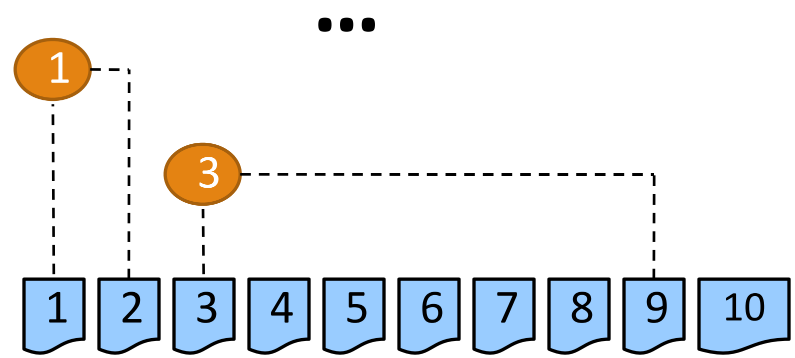
# Hierarchical Agglomerative Clustering (4/6)




*I* =

| *I*[1] | *I*[2] | *I*[3] | *I*[4] | *I*[5] | *I*[6] | *I*[7] | *I*[8] | *I*[9] | *I*[10] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

*A* = { <3,9> <1,2> ••• }




1. Search for the max similarity pair - <*i*,*m*>
2. Record and merge cluster *m* into cluster *i*.
3. Update the pair similarities between the new merged cluster *i*. and other clusters.
4. *I*[*m*] = 0, remove the merged cluster *m*.

So complicated …
No worry, `SKLearn` helps us do it in an easy way!!

# Hierarchical Agglomerative Clustering (5/6)

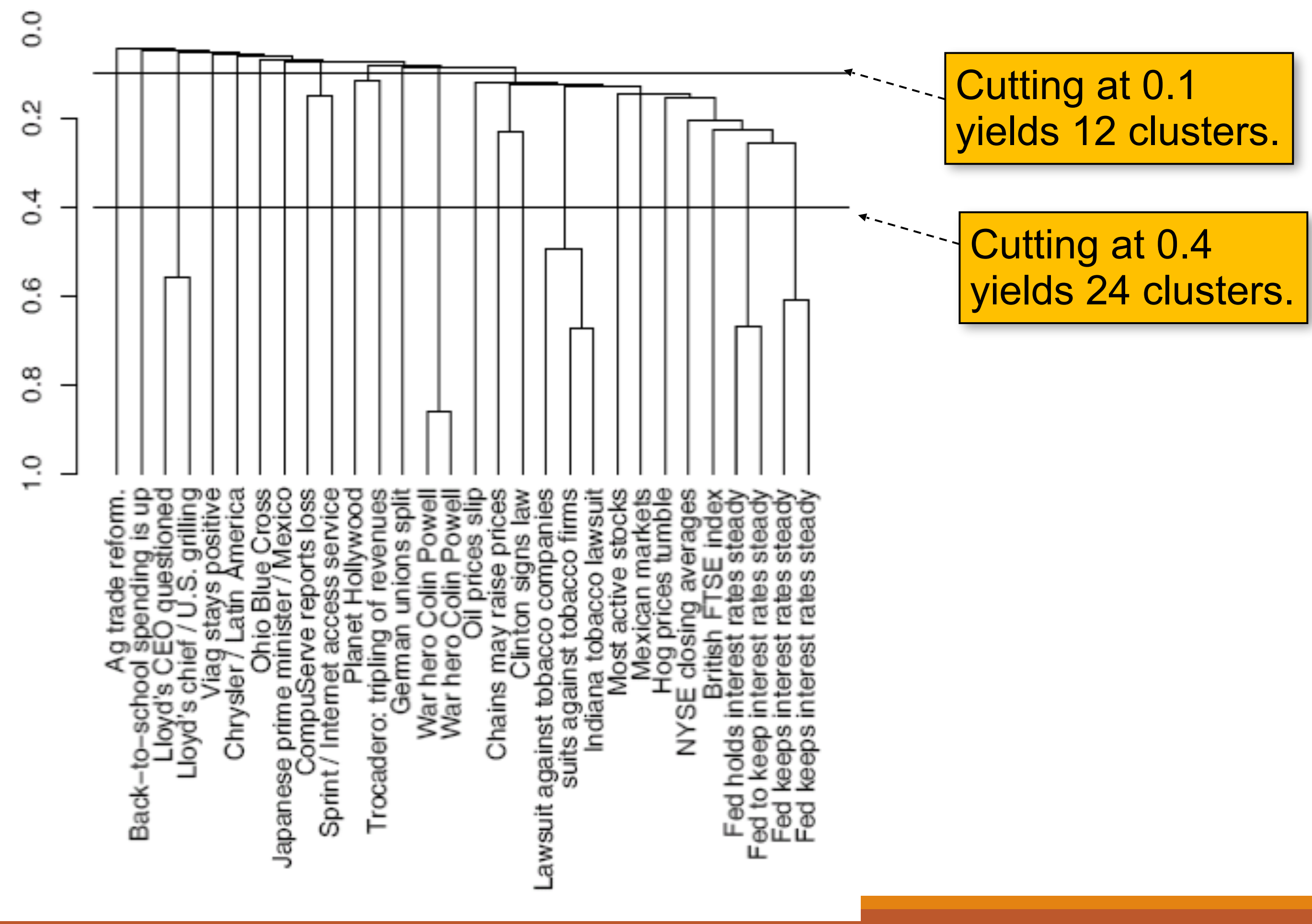
In some cases, we want a partition of disjoint clusters just as in flat clustering.

- The hierarchy needs to be <u>cut at some point</u>!!

**Criteria of cutting**:

- As in flat clustering, we can also pre-specify the number of clusters *K*.
- Cut at a pre-specified level of similarity.
  - We cut the dendrogram at 0.4.
  - We want clusters with a minimum combination similarity of 0.4.
- Cut the dendrogram where the gap between two successive combination similarities is largest.
  - Merging one more cluster decreases the quality of the clustering significantly.

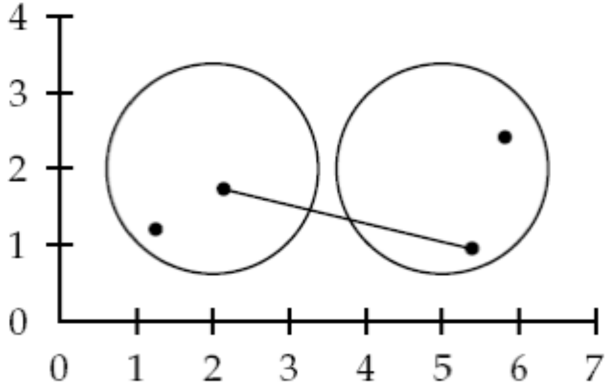
# Hierarchical Agglomerative Clustering (6/6)



# Single-link and Complete-link (1/7)

***Single-link*** :

- The similarity of two clusters is the similarity of **their most similar members**.
- This merge criterion is **local**.
  - We pay attention solely to the area where the two clusters come closest to each other.
  - The cluster overall structure is not taken into account.

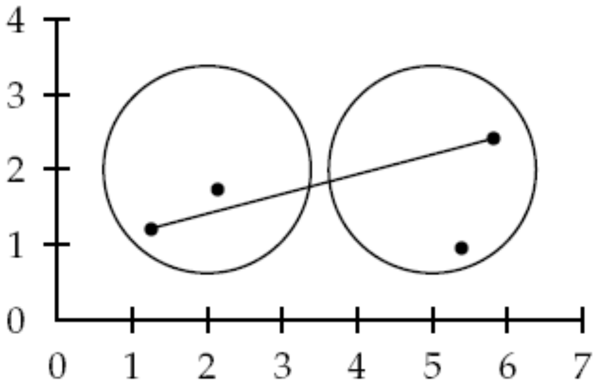



(a) single link: maximum similarity

# Single-link and Complete-link (2/7)

***Complete-link*** :

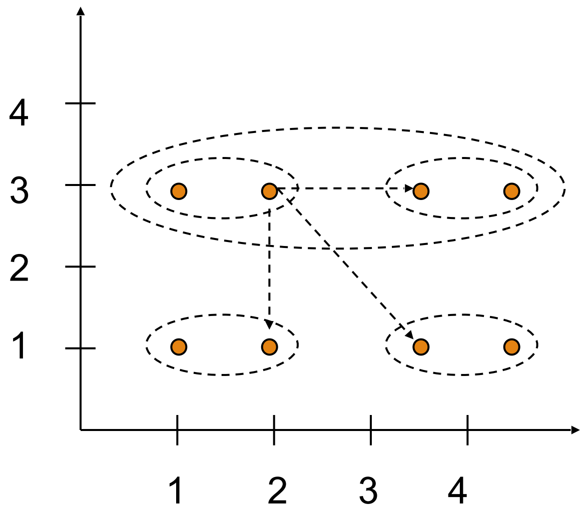
- The similarity of two cluster is the similarity of **their most dissimilar members**.
- This merge criterion is **non-local**.
  - The entire structure of the clustering can influence merge decision.
- Is equivalent to choosing the cluster pair whose merge has the smallest diameter.



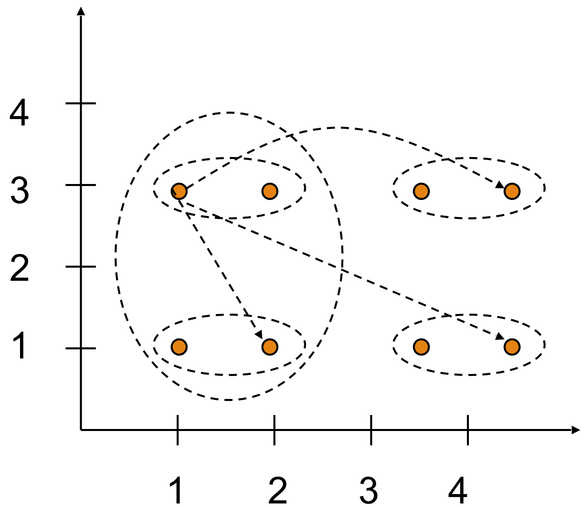
(b) complete link: minimum similarity

# Single-link and Complete-link (3/7)

**Single-link**




**Complete-link**



# Single-link and Complete-link (4/7)

Graph-theoretic interpretations:

- Let $s_k$ to be the combination similarity of the two clusters merged in step $k$.
- $G(s_k)$ the graph that links all data points with a similarity of at least $s_k$.

- Then, the clusters after step $k$ in **single-link** clustering are the ***connected components*** of $G(s_k)$.
  - Points in a clusters are linked via at least one link → **single-link**!!

- And the clusters after step $k$ in **complete-link** clustering are maximal ***cliques*** of $G(s_k)$.
  - A clique is a set of points that are <u>completely linked with each other</u> → **complete-link**!!

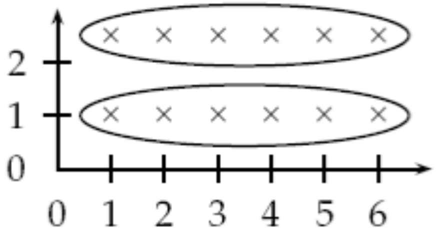
# Single-link and Complete-link (5/7)

Single-link assesses cluster quality via a pair of document.

- The two most similar documents.
- Clustering based on one pair cannot fully reflect the distribution of documents in a cluster.

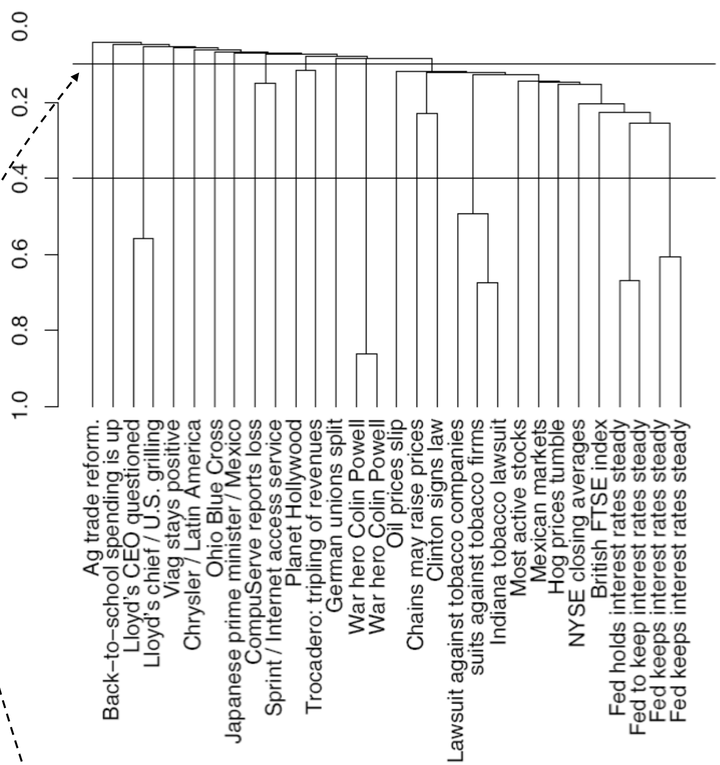
***Chaining***: a problem of single-link.

- A chain of points can be extended for long distances without regard to the overall shape of the cluster.

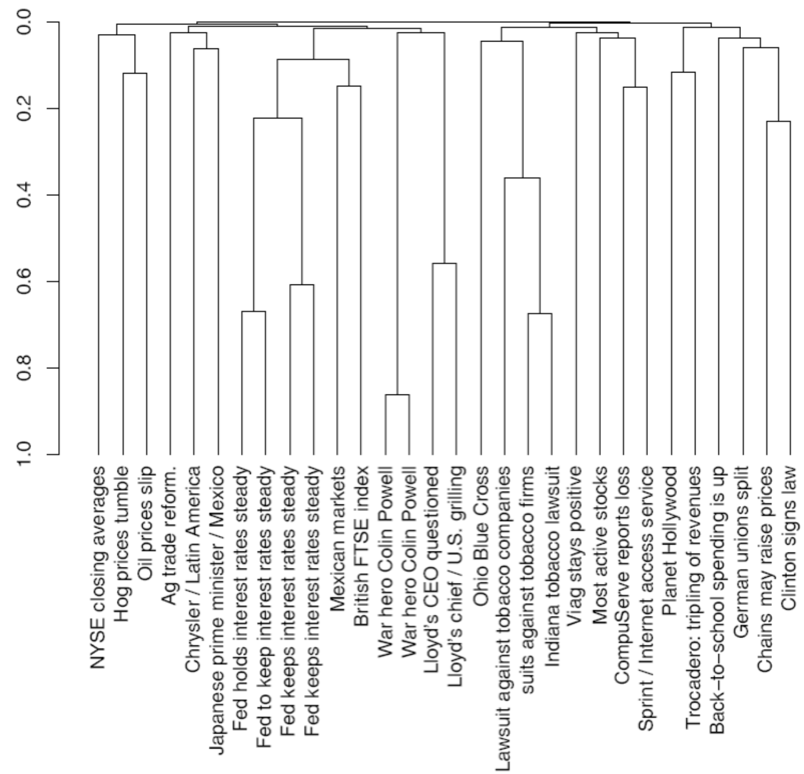



► **Figure 17.6** Chaining in single-link clustering. The local criterion in single-link clustering can cause undesirable elongated clusters.

# Single-link and Complete-link (6/7)




► **Figure 17.1** A dendrogram of a single-link clustering of 30 documents from Reuters-RCV1. The y-axis represents combination similarity, the similarity of the




► **Figure 17.5** A dendrogram of a complete-link clustering of the 30 documents in Figure 17.1.

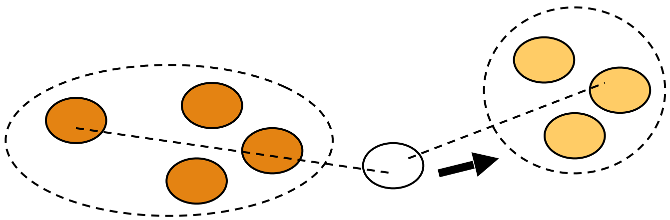
The last 12 merges add on single documents → chaining!!

We **often** prefer compact clusters with small diameters over long, straggly clusters.

# Single-link and Complete-link (7/7)

The problem of complete-link clustering – **very sensitive to outliers**.

- A single document far from the center can increase diameters of candidate merge clusters dramatically.



# Group-average Linkage (1/2)

***Group-average***:

- The similarity of two cluster is the average similarity of all pairs of documents.
  - **Including pairs from the same cluster!!**
  - **But self-similarities are not included in the average!!**

$$sim-ga(\omega_i,\omega_j)=\frac{1}{(N_i+N_j)(N_i+N_j-1)}\sum_{d_m\in\omega_i\cup\omega_j}\ \sum_{d_n\in\omega_i\cup\omega_j,\,d_n\neq d_m}\vec{d}_m\cdot\vec{d}_n$$

clusters *i* and *j*

the number of documents in cluster *i*

the **normalized** document vectors

# Group-average Linkage (2/2)

Why not including self-similarity?

- If we define group-average similarity as including self-similarities:

$$sim-ga'(\omega_i,\omega_j)=\frac{1}{(N_i+N_j)^2}\sum_{d_m\in\omega_i\cup\omega_j}\sum_{d_n\in\omega_i\cup\omega_j}\vec{d}_m\cdot\vec{d}_n$$

- For a cluster of size *i*, the proportion of self-similarities is $i/i^2$ or $1/i$.
  - **This gives an unfair advantage to small clusters**.

- Moreover, for two documents $d_1$, $d_2$ with similarity $s$
  - $sim\text{-}ga'(d_1,d_2) = (2+2s)/4 = (1+s)/2$.
  - $sim\text{-}ga(d_1,d_2) = 2s/2 = s$.
    - is the same as in single-link, complete-link, and centroid clustering.

# Centroid Linkage

***Centroid linkage***:

- The similarity of two cluster is defined as the similarity of their centroids:

$$sim-cent(\omega_i,\omega_j)=\Big(\frac{1}{N_i}\sum_{d_m\in\omega_i}\vec{d}_m\Big)\cdot\Big(\frac{1}{N_j}\sum_{d_n\in\omega_j}\vec{d}_n\Big)$$

$$=\frac{1}{N_iN_j}\sum_{d_m\in\omega_i}\sum_{d_n\in\omega_j}\vec{d}_m\cdot\vec{d}_n$$

  - The similarity is equivalent to average similarity of all pairs of documents from **different** clusters.

# Ward's Linkage (1/2)

***Ward's Linkage***:

- The distance between two clusters is how much the sum of squares will increase when we merge them:

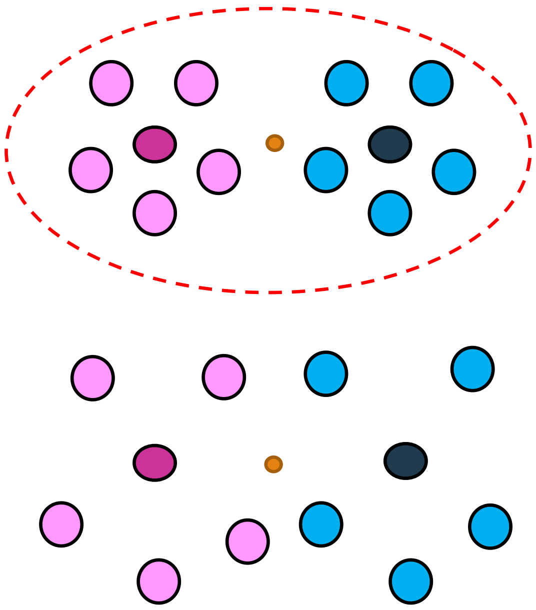
$$Ward(\omega_i, \omega_j) = \sum_{d_m \in \omega_i \cup \omega_j} \left\| \vec{d}_m - centroid_{\omega_i \cup \omega_j} \right\|^2 - \sum_{d_m \in \omega_i} \left\| \vec{d}_m - centroid_{\omega_i} \right\|^2 - \sum_{d_m \in \omega_j} \left\| \vec{d}_m - centroid_{\omega_j} \right\|^2$$

The sum of squares starts out at zero, and grows as we merge clusters

- In the beginning, every document (vector) is in its own cluster.

Ward's linkage keeps this growth as small as possible.

# Ward's Linkage (2/2)



Which two clusters would Ward's linkage group first?

Another interpretation of Ward's linkage is **<u>to minimize the variance of the clusters being merged</u>**.

# Which Similarity Strategy Is The Best ?

[Voorhees 1985] recommends complete-link and centroid clustering over single-link for a retrieval application.

But in news-related applications (e.g., event/topic detection), single-link seems to be a better strategy.

- Allan et al. (1998) apply the single-link clustering to first story detection.

# Practice Time (1/2)

```
In [8]: from sklearn.cluster import AgglomerativeClustering
        hac = AgglomerativeClustering(linkage='ward', n_clusters=2)
        hac.fit(TFIDF_vectors.toarray())

Out[8]: AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                                connectivity=None, distance_threshold=None,
                                linkage='ward', memory=None, n_clusters=2)

In [9]: hac.labels_

Out[9]: array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0])
```

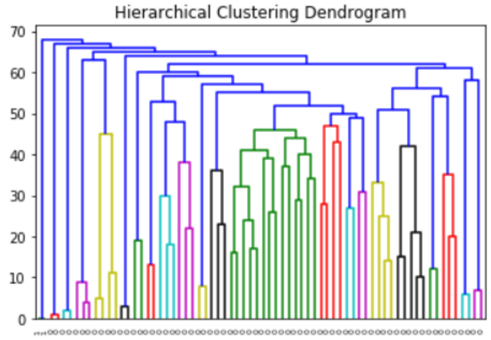
Parameters:

- `linkage`: `ward`, `single`, `complete`, `average`
  - default – `ward`
- `n_clusters`: the number of clusters to find.
- `distance_threshold`: the distance threshold to stop merging

# Practice Time (2/2)

How to show the cluster hierarchy?

```
In [12]: plt.title('Hierarchical Clustering Dendrogram')
         plot_dendrogram(hac, labels=hac.labels_)
         plt.show()
```




```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram


def plot_dendrogram(model, **kwargs):
    # https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto
    # Children of hierarchical clustering
    children = model.children_

    # Distances between each pair of children
    # Since we don't have this information, we can use a uniform one for plotting
    distance = np.arange(children.shape[0])

    # The number of observations contained in each cluster level
    no_of_observations = np.arange(2, children.shape[0]+2)

    # Create linkage matrix and then plot the dendrogram
    linkage_matrix = np.column_stack([children, distance, no_of_observations]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```