Hierarchical clustering

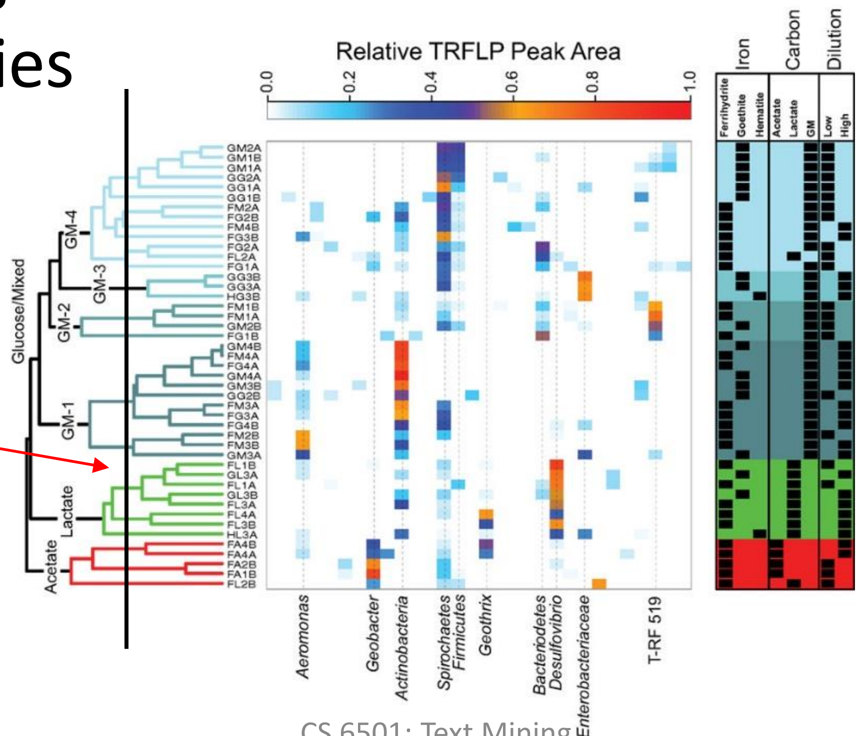
Hongning Wang
CS@UVa

Today’s lecture

- Hierarchical clustering algorithm
  - Bottom-up: agglomerative
  - Distance between clusters
  - Complexity analysis

Hierarchical clustering

- Build a tree-based hierarchical taxonomy from a set of instances
  - Dendrogram – a useful tool to summarize similarities



After cutting, each connected component will be a cluster

Agglomerative hierarchical clustering

- Pairwise distance metric between instances

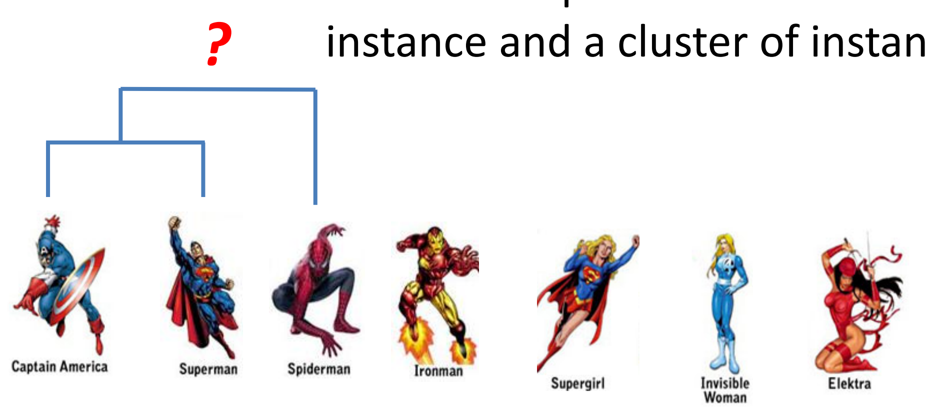
	Captain America	Superman	Spiderman	Ironman	Supergirl	Invisible Woman	Elektra
Captain America	0	1	2	2	3	3	4
Superman		0	2	2	3	3	4
Spiderman			0	2	3	3	4
Ironman				0	3	4	4
Supergi					0	2	3
nvisible Woman						0	2
Elektra							0

Agglomerative hierarchical clustering

1. Every instance is in its own cluster when initialized
2. Repeat until one cluster left
   1. Find the best pair of clusters to merge and break the tie arbitrarily

Enumerate all the possibilities!

How to compare distance between an instance and a cluster of instances?

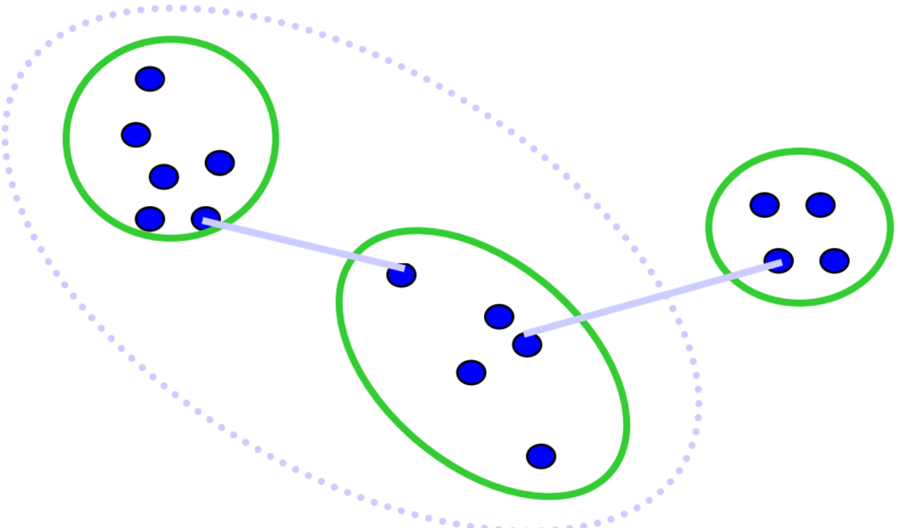


0	1	2	2	3	3	4
	0	2	2	3	3	4
		0	2	3	3	4
			0	3	4	4
				0	2	3
					0	2
						0

Distance measure between clusters

- Single link
  - Cluster distance = distance of two closest members between the clusters
  - $d(c_i, c_j) = \min_{x_n \in c_i, x_m \in c_j} d(x_n, x_m)$

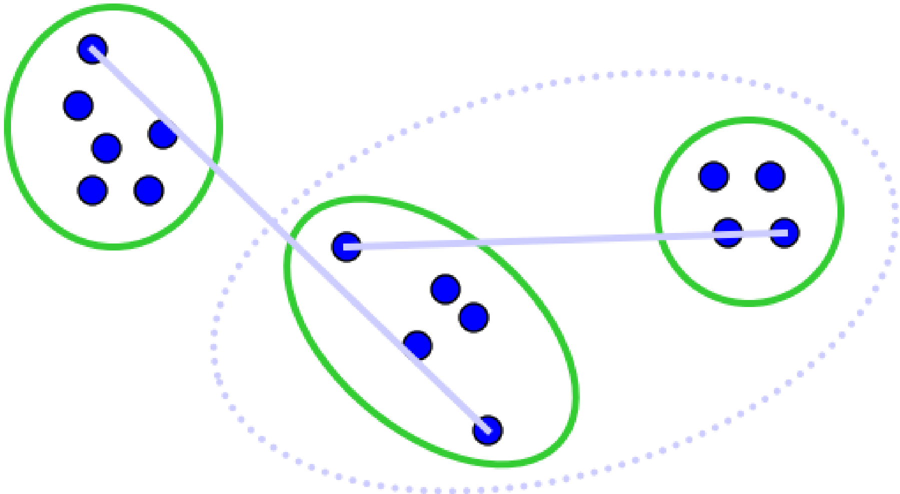
Tend to generate scattered clusters



Distance measure between clusters

- Complete link
  - Cluster distance = distance of two farthest members between the clusters
  - $d(c_i, c_j) = \max_{x_n \in c_i, x_m \in c_j} d(x_n, x_m)$

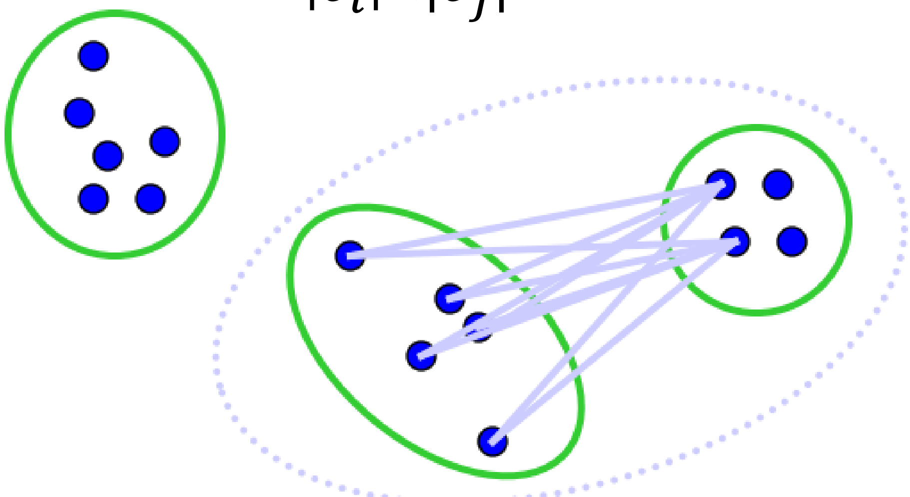
Tend to generate tight clusters



Distance measure between clusters

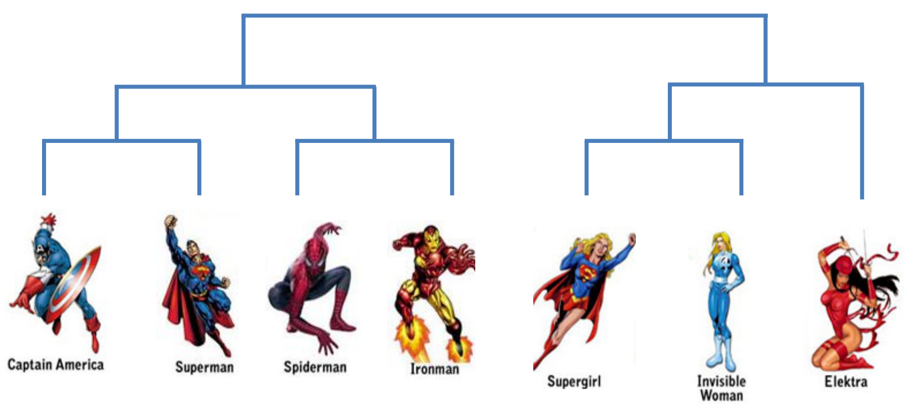
- Average link
  - Cluster distance = average distance of all pairs of members between the clusters
  - $d(c_i, c_j) = \frac{\sum_{x_n \in c_i, x_m \in c_j} d(x_n, x_m)}{|c_i| \times |c_j|}$

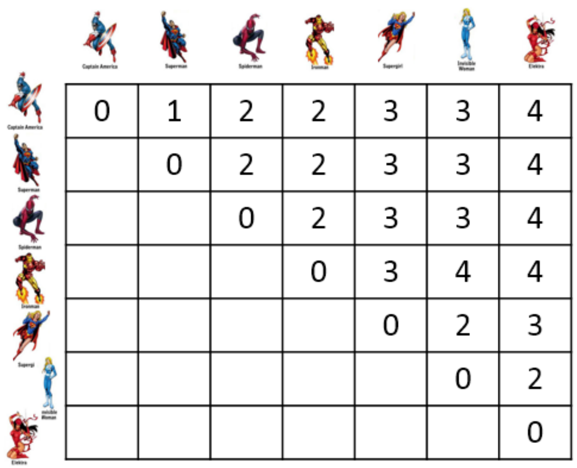
Mostly popularly used measure, robust against noise



Agglomerative hierarchical clustering

1. Every instance is in its own cluster when initialized
2. Repeat until one cluster left
    1. Find the best pair of clusters to merge and break the tie arbitrarily





0	1	2	2	3	3	4
	0	2	2	3	3	4
		0	2	3	3	4
			0	3	4	4
				0	2	3
					0	2
						0

Complexity analysis

- In step one, compute similarity between all pairs of n individual instances - $O(n^2)$
- In the following $n-2$ steps
  – It could be $O(n^2 \log n)$ or even $O(n^3)$ (naïve implementation)

In k-means, we have $O(knl)$,
a much faster algorithm

Comparisons

- Hierarchical clustering
  - Efficiency: $O(n^3)$, slow
- Assumptions
  - No assumption
  - Only need distance metric
- Output
  - Dendrogram, a tree

- k-means clustering
  - Efficiency: $O(knl)$, fast
- Assumptions
  - Strong assumption – centroid, latent cluster membership
  - Need to specify k
- Output
  - k clusters

How to get final clusters?

- If k is specified, find a cut that generates k clusters
  - Since every time we only merge 2 clusters, such cut must exist
- If k is not specified, use the same strategy as in k-means
  - Cross validation with internal or external validation

What you should know

- Agglomerative hierarchical clustering
  - Three types of linkage function
    - Single link, complete link and average link
  - Comparison with *k*-means

Today’s reading

- Introduction to Information Retrieval
  - Chapter 17: Hierarchical clustering
    - 17.1 Hierarchical agglomerative clustering
    - 17.2 Single-link and complete-link clustering
    - 17.3 Group-average agglomerative clustering
    - 17.5 Optimality of HAC