Hierarchical clustering

School of Electrical and Computer Engineering

University of Tehran

Erfan Darzi

erfandarzi@ut.ac.ir

Today's lecture

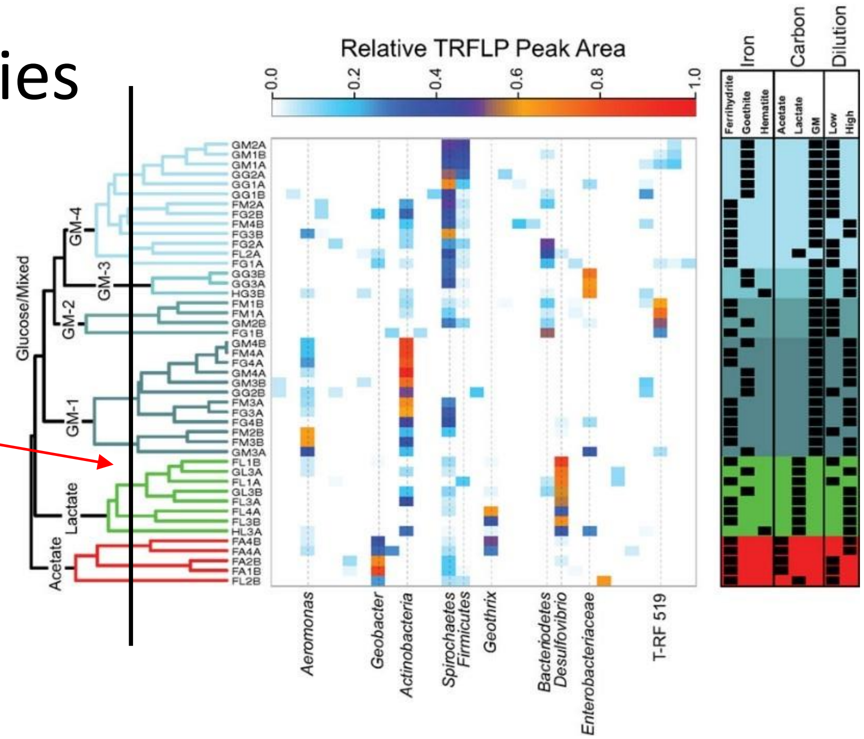
- Hierarchical clustering algorithm
  - Bottom-up: agglomerative
  - Distance between clusters
  - Complexity analysis

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
Supergirl					0	2	3
Invisible Woman						0	2
Elektra							0

Agglomerative hierarchical clustering

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

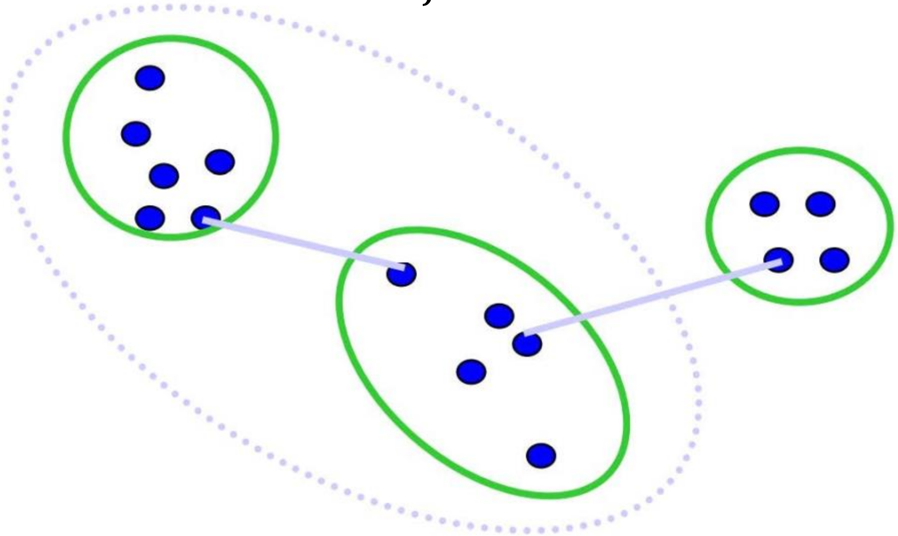
How to compare distance between an ? instance and a cluster of instances?

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

- Single link – Cluster distance = distance of two closest members between the clusters

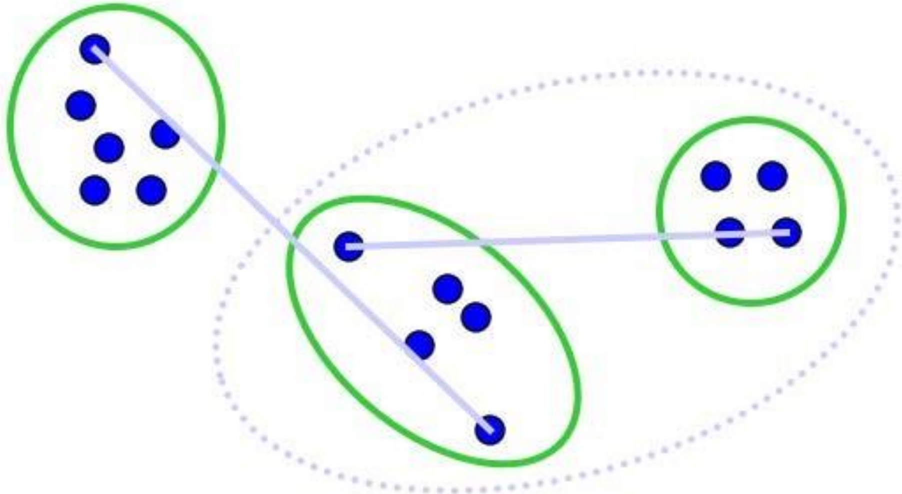
  - $d(c_i, c_j) = \min_{x_n \in c_i, x_m \in c_j} d(x_n, x_m)$

Tend to generate scattered clusters



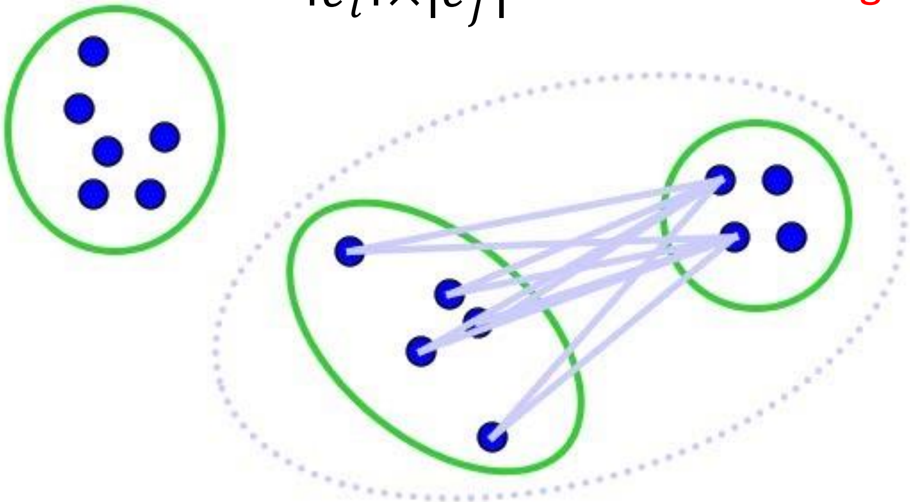
- Complete link
  - Cluster distance = distance of two farthest members between the clusters
  - $d(c_i, c_j) = \max_{x_n \in c_i, x_m \in c_j} d(x_n, x_m)$

Tend to generate tight clusters



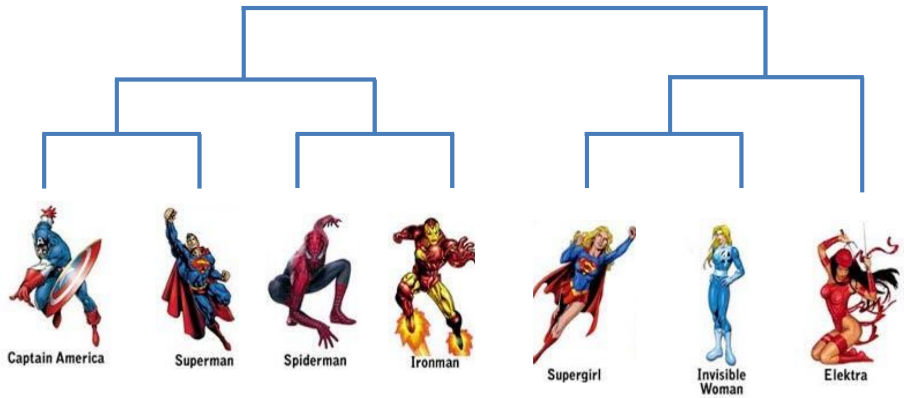
- Average link
  - Cluster distance = average distance of all pairs of members between the clusters
  - $d(c_i, c_j) = \frac{\sum_{x_n \in c_i, x_m \in c_j} d(x_n, x_m)}{|c_i| \times |c_j|}$

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 nn individual instances - $OO(nn^2)$

- In the following $nn-2$ steps
  - It could be $OO(nn^2 \log nn)$ or even $OO(nn^3)$ (naïve implementation)

In k-means, we have $OO(kknnkk)$, a much faster algorithm

Comparisons

- Hierarchical clustering
  - Efficiency: $OO(nn^3)$, slow

- Assumptions – No assumption – Only need distance metric
- Output
  - Dendrogram, a tree
- k-means clustering
  - Efficiency: $OO(kknnkk)$, fast
- Assumptions
  - Strong assumption –

    centroid, latent cluster membership

- Need to specify kk

- Output
  - kk clusters

How to get final clusters?

- If kk is specified, find a cut that generates kk clusters
  - Since every time we only merge 2 clusters, such cut must exist

- If kk 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
-