Clustering: Part II

CS434

Hard vs. Soft Clustering

- Kmeans performs what we call hard clustering
  - Data point is deterministically assigned to one and only one cluster
- Soft-clustering:
  - Data points are assigned to clusters with certain probabilities
  - Typically assume some probabilistic model to represent the clusters

Soft clustering: Gaussian Mixture Modeling

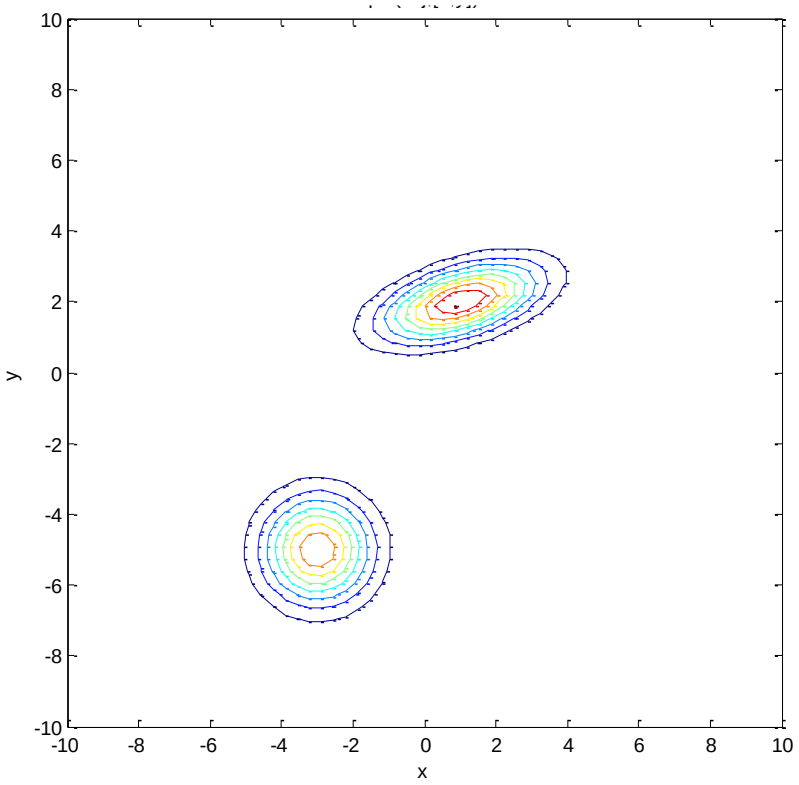
- Given data $\{x^1, \dots, x^m\}$, assume k clusters
- Assume each cluster is generated from a Gaussian
- The proportion of the clusters are decided by $p(y = 1), \dots, p(y = k)$

Example: two Gaussians

$$p(y = 1) = p(y = 2) = 0.5$$

$$\mu_1 = [1,2], \Sigma_1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 5 \end{bmatrix}$$

$$\mu_2 = [-3, -5], \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Soft clustering: Gaussian Mixture Modeling

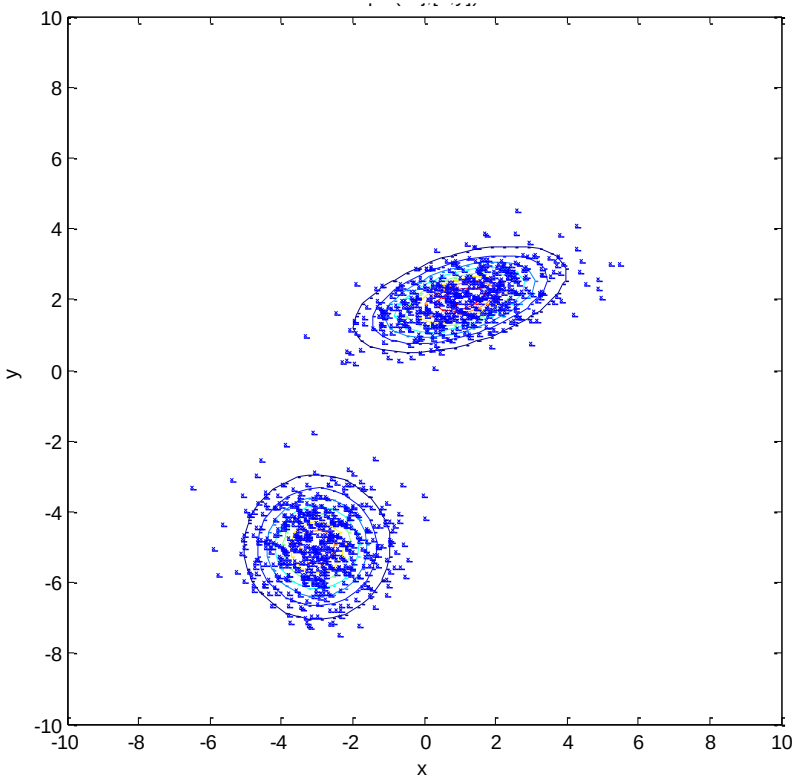
- Given data $\{x^1, \dots, x^m\}$, assume k clusters
- Assume each cluster is generated from a Gaussian
- The proportion of the clusters are decided by $p(y = 1), \dots, p(y = k)$

Example: sample points from the two Gaussians

$$p(y = 1) = p(y = 2) = 0.5$$

$$\mu_1 = [1,2], \Sigma_1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 5 \end{bmatrix}$$

$$\mu_2 = [-3,-5], \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Soft clustering: Gaussian Mixture Modeling

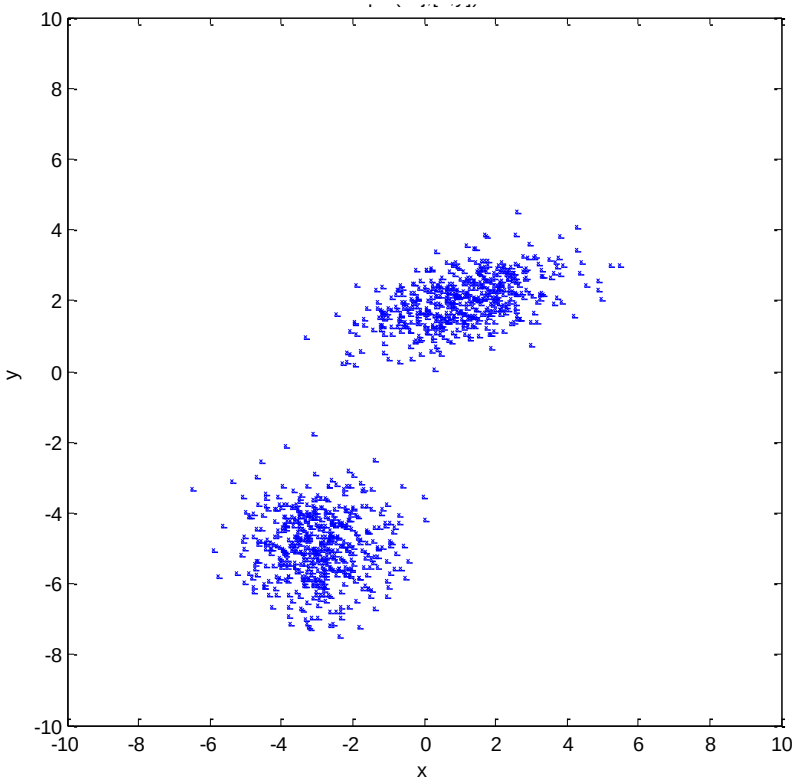
- Given data $\{x^1, \dots, x^m\}$, assume k clusters
- Assume each cluster is generated from a Gaussian
- The proportion of the clusters are decided by $p(y = 1), \dots, p(y = k)$

Example: observe the points, need to estimate the Gaussians

$$p(y = 1) = p(y = 2) = 0.5$$

$$\mu_1 = [1,2], \Sigma_1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 5 \end{bmatrix}$$

$$\mu_2 = [-3,-5], \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Expectation Maximization: a simple case

- A simple case:
  - We have unlabeled data $x^1, \cdots, x^m$
  - We know there are k clusters
  - We know all covariance matrices are the same

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix}$$

Start with an initial guess for $\mu_1, \ldots, \mu_k$, and $p(y=1), \ldots, p(y=k)$, repeat:

1. If we know $\mu_1, \ldots, \mu_k$, we can easily compute probability that a point x^j belongs to class i:

$$p(y=i|x^j) \propto \exp\left(-\frac{1}{2\sigma^2}|x^j - \mu_i|^2\right) p(y=i)$$

Simply evaluate this, then normalize

E-step

2. If we know the probability that each point belongs to each class, we can estimate the $\mu_1, \ldots, \mu_k$ and $p(y=1), \ldots, p(y=k)$

$$p(y=i) = \frac{\sum_{j=1}^{m} p(y=i|x^j)}{m}$$

Counting the points in each cluster

$$\mu_i = \frac{\sum_{j=1}^{m} p(y=i|x^j) x^j}{\sum_{j=1}^{m} p(y=i|x^j)}$$

Computing the center of each cluster

M-step

Similarity/Difference with Kmeans?

- E-step $\approx$ reassignment step
  - E-step assigns a point probabilistically
    - Clusters with closer center will get higher probability
    - How spread out the probability is depends on σ^2
    - In the extreme case, $\sigma^2 \to 0$, this will reduce to kmeans
  - K-Means reassign a point to a cluster deterministically
    - The closest cluster center gets 100%
- M-step $\approx$ recentering
  - M-step computes the center using weighted average

More General Cases

A simple case:

We have unlabeled data $x^1, \cdots, x^m$

We know there are k clusters

We know all covariance matrices are the same

Need to estimate $p_1, \ldots, p_k;\ \mu_1, \mu_2, \ldots, \mu_k$

General Case:

We have unlabeled data $x^1, \cdots, x^m$

We know there are k clusters

Each cluster has its own covariance matrix

Need to estimate $p_1, \ldots, p_k;\ \mu_1, \ldots, \mu_k;\ \Sigma_1, \ldots, \Sigma_k$

Different Covariance Matrix Capture Different Shapes

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix}$$

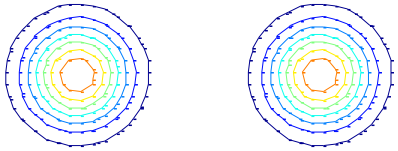
$\sigma^2 I$: a scaled identity matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}$$

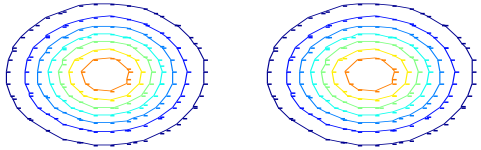
Diagonal matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 \end{bmatrix}$$

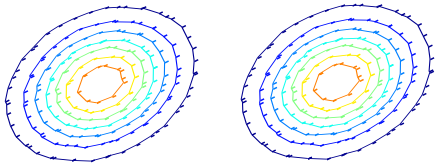
Full covariance matrix – most general



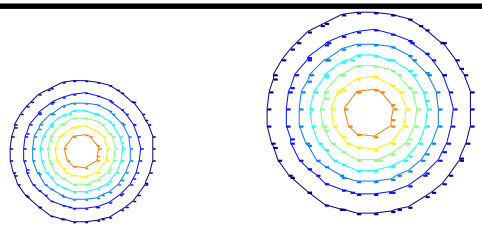
Different clusters with the same σ^2



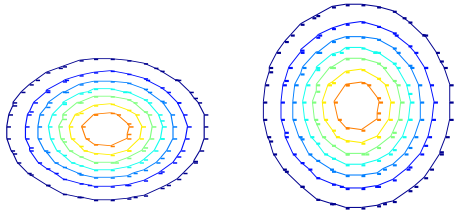
Different clusters with the same Σ



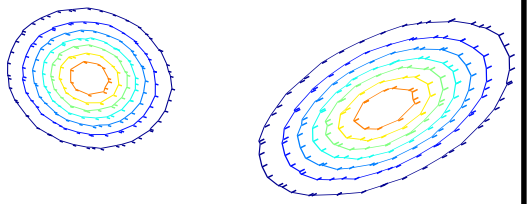
Different clusters with the same Σ



Different clusters with different σ^2 's

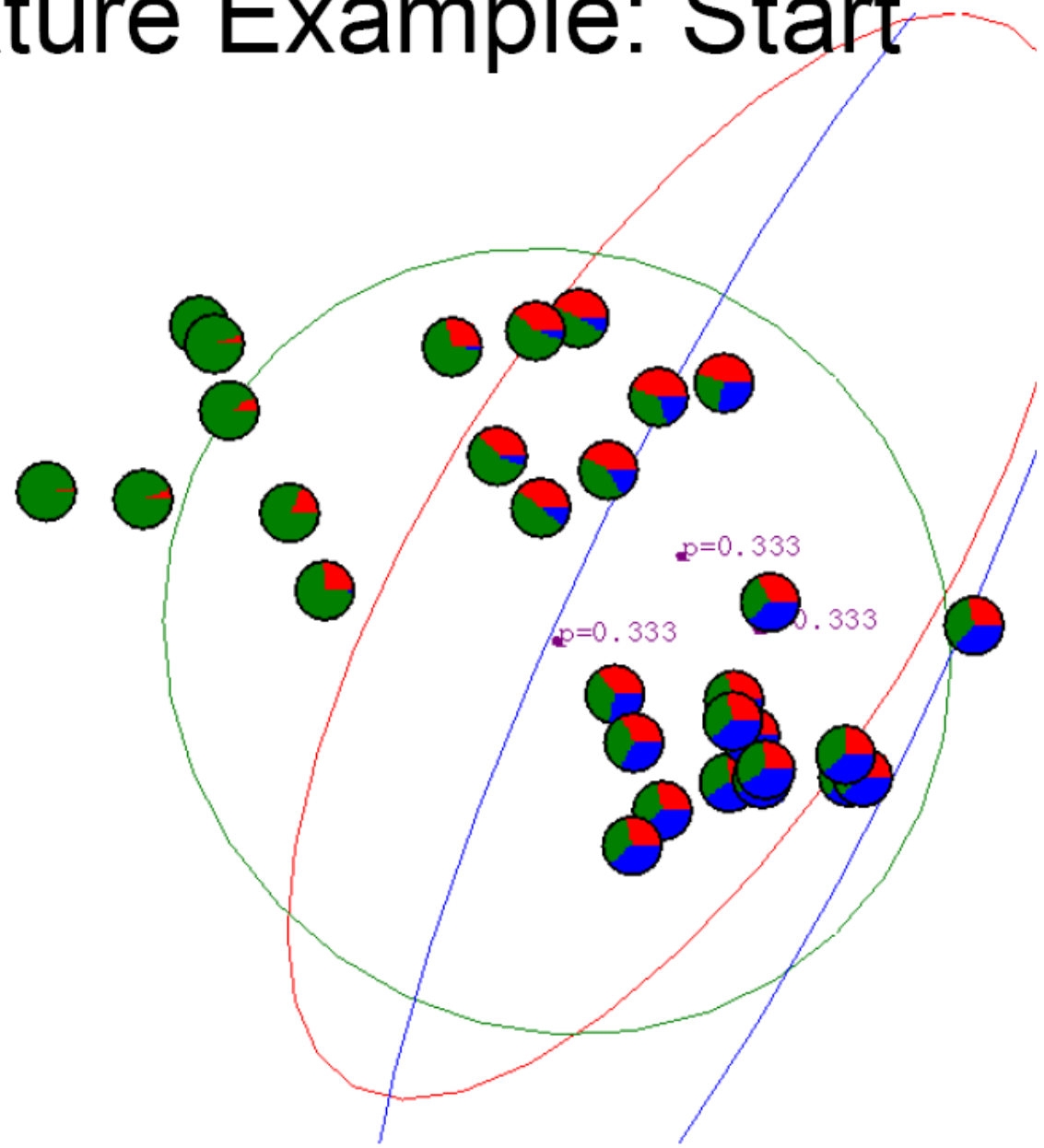


Different clusters with different Σ's

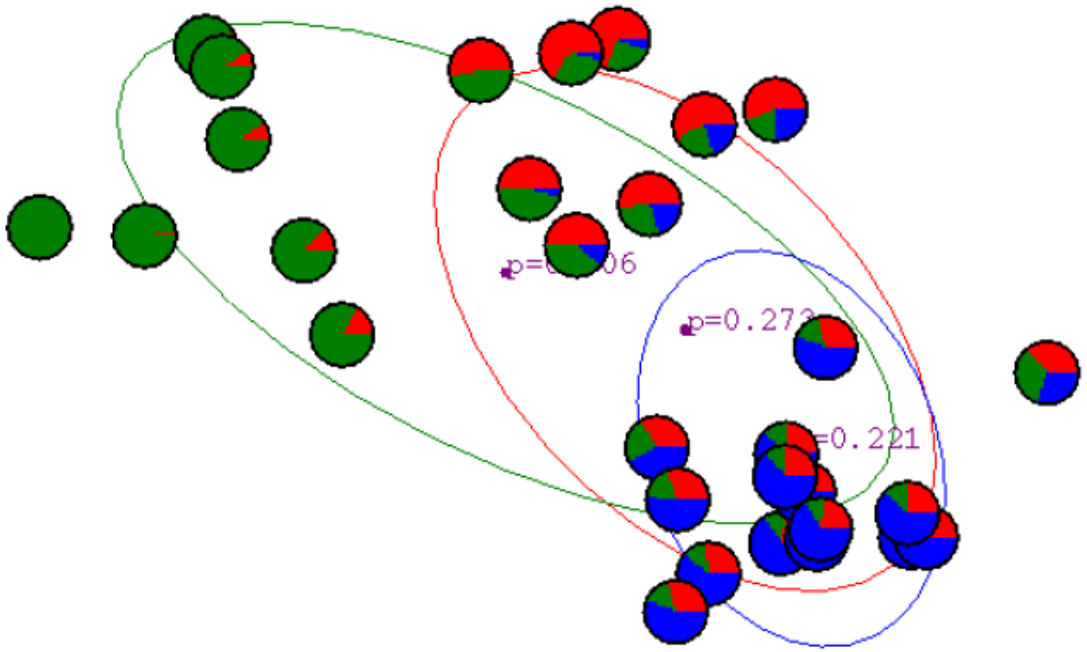


Different clusters with different Σ's

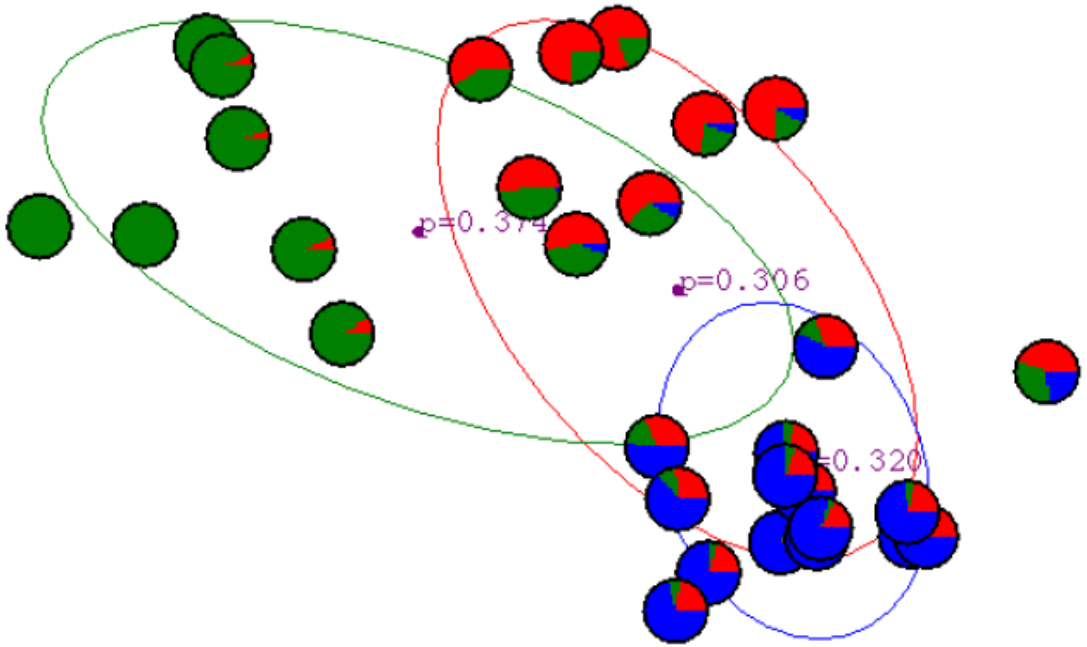
Gaussian Mixture Example: Start



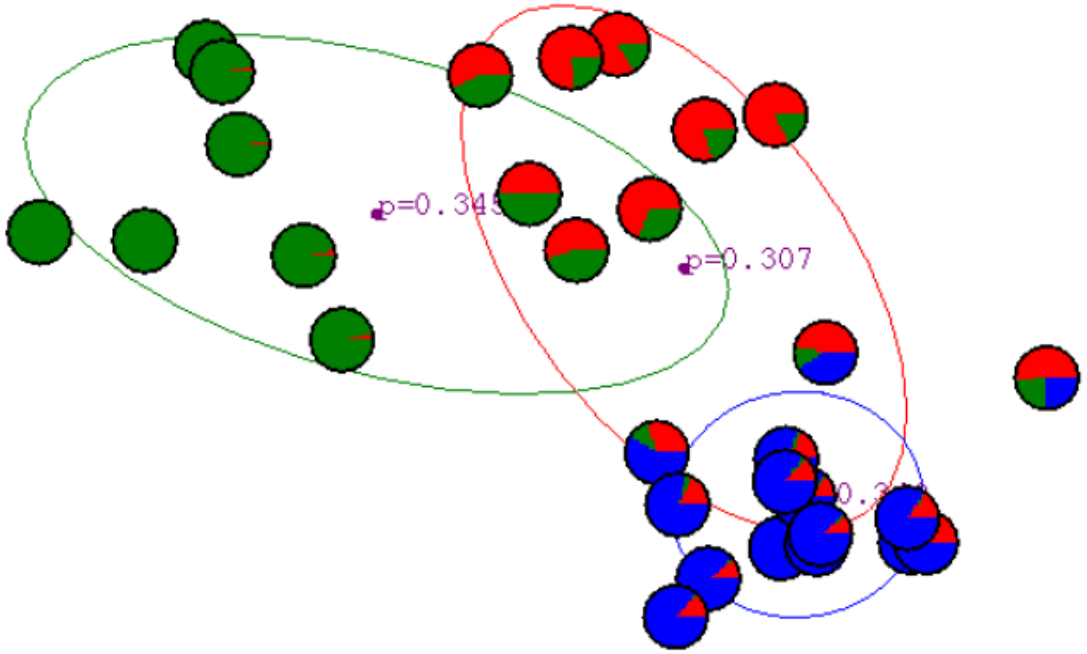
After first iteration



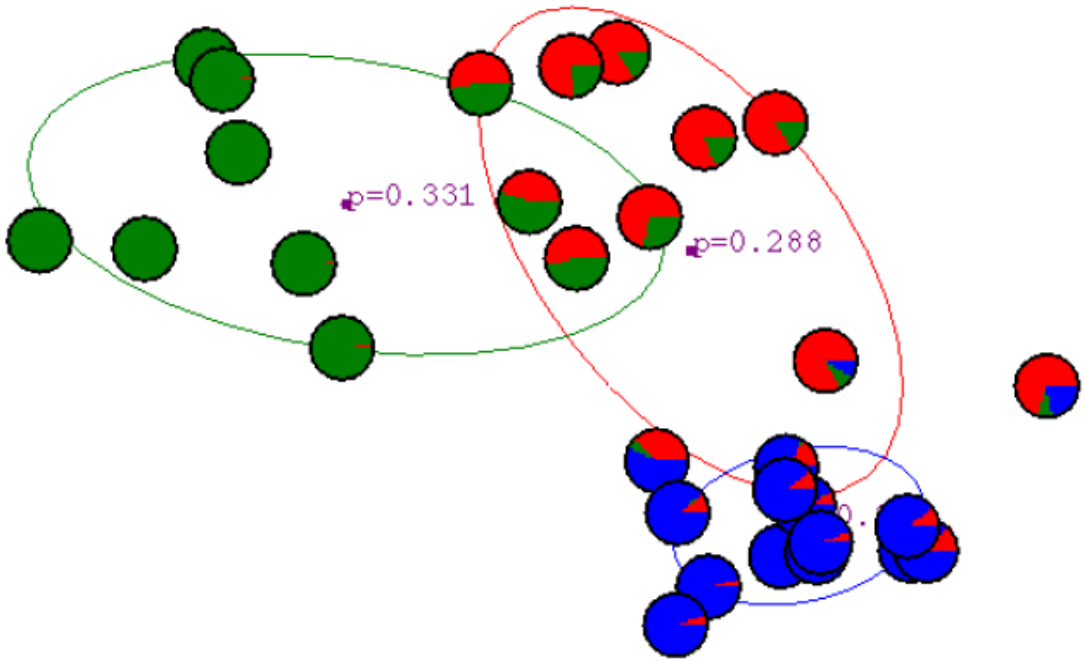
After 2nd iteration



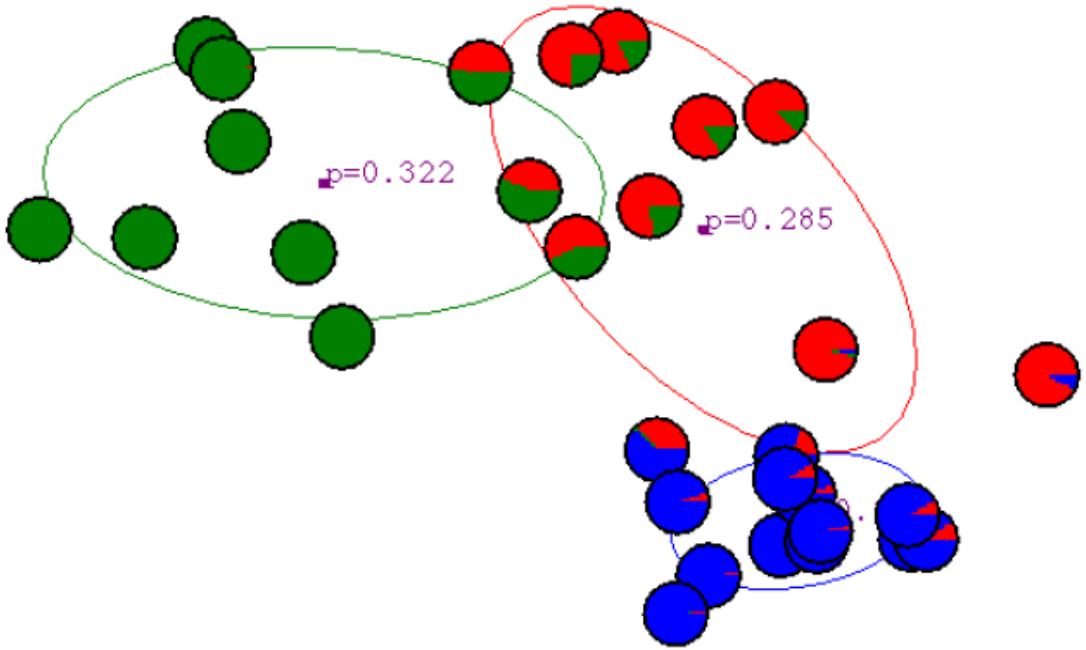
After 3rd iteration



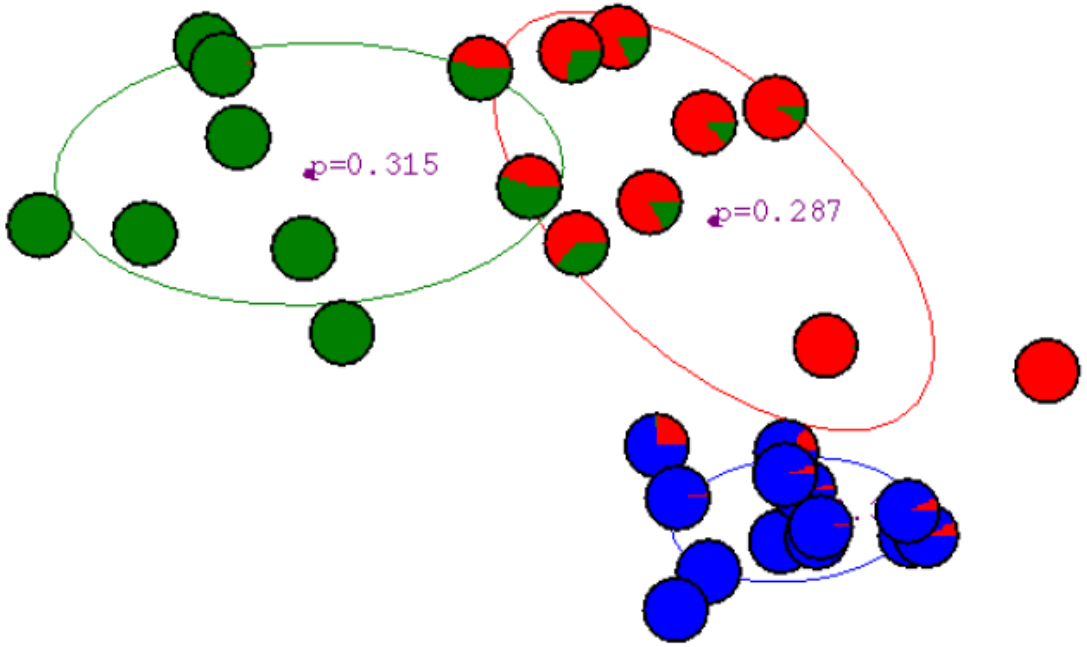
After 4th iteration



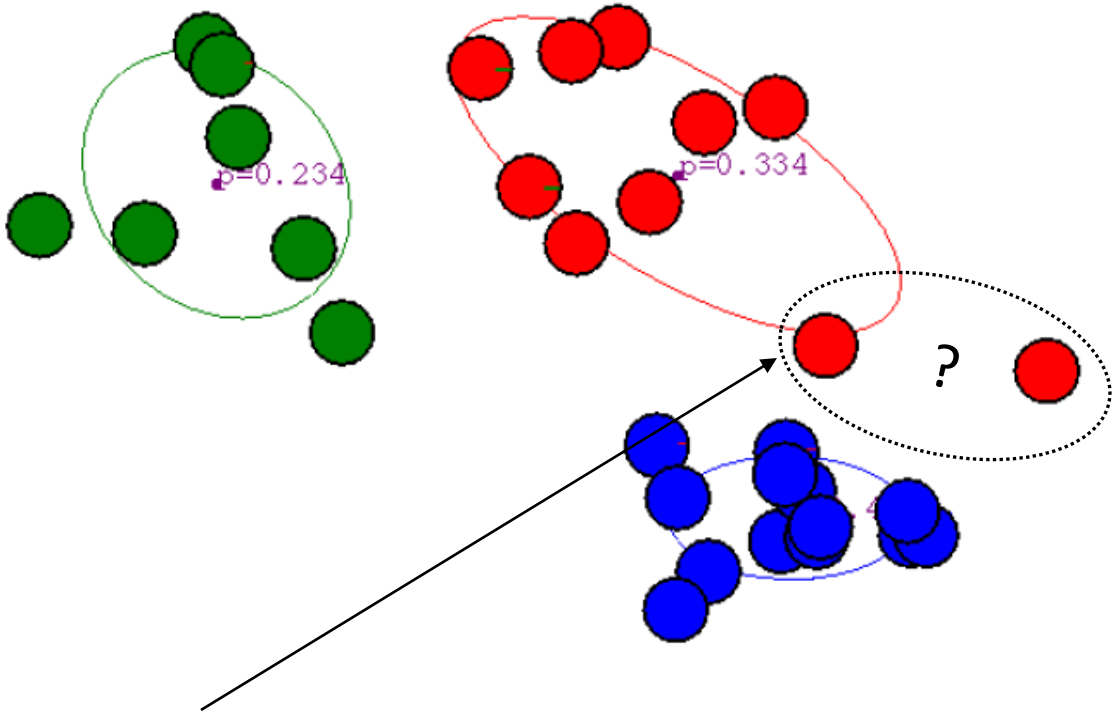
After 5th iteration



After 6th iteration



After 20th iteration



Q: Why are these two points red?

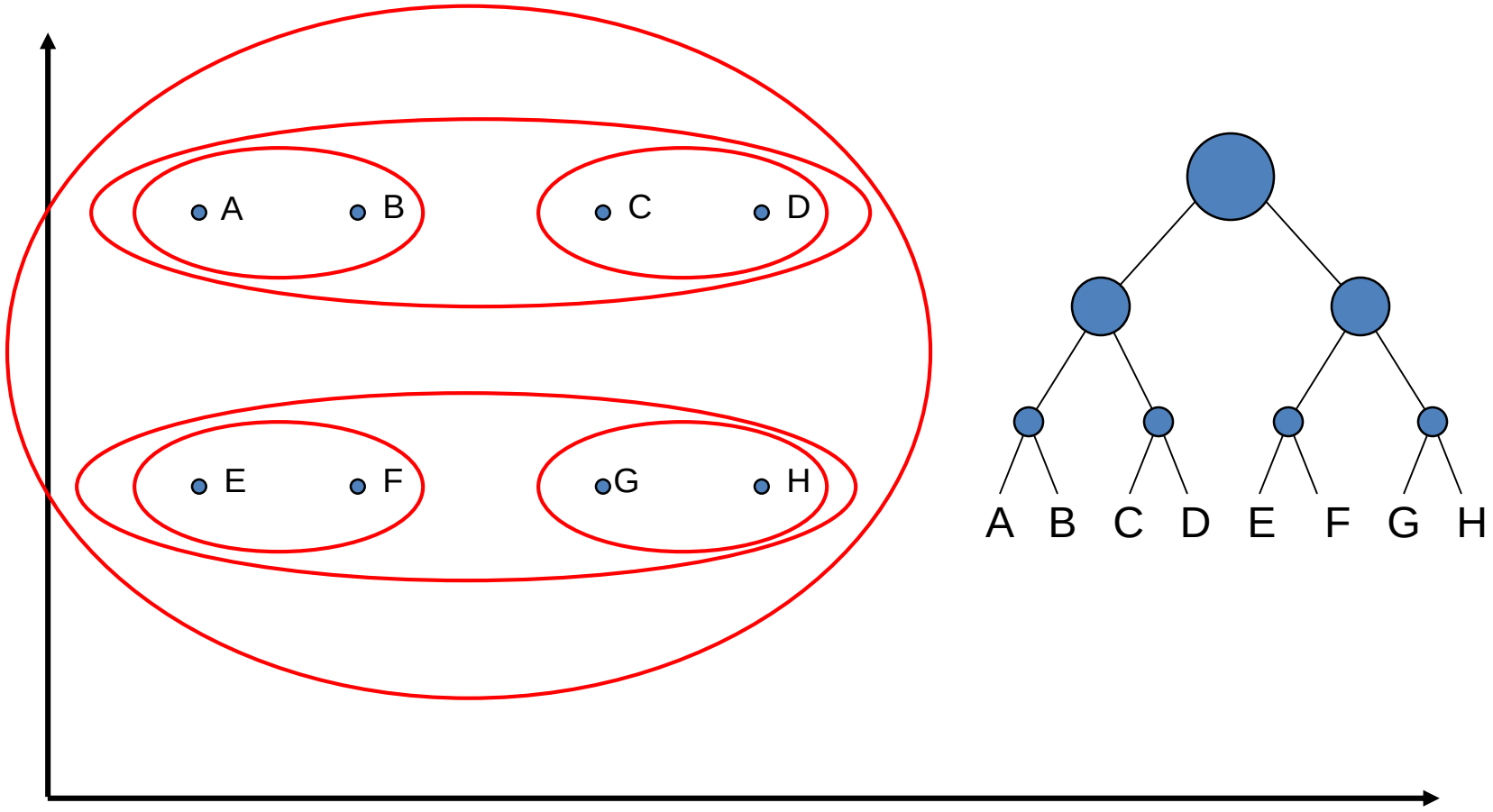
Behavior of EM

- It is guaranteed to converge
  - Convergence proof is based on the fact that $\log P(x|\theta)$ must increase or remain same between iterations (not obvious)
  - In practice it may converge slowly, one can stop early if the change in log-likelihood is smaller than a threshold
- It converges to a local optimum
  - Multiple restart is recommended
  - Choose the one that has the highest $\log P(x|\theta)$

Hierarchical Agglomerative Clustering (HAC)

- Assumes a *distance function* for determining the similarity of two instances.
  - One can also assume a similarity function, and reverse some of the operations (e.g., minimum distance -> maximum similarity) in the algorithm to make it work for similarities
- Starts with each object in a separate cluster and then repeatedly joins the two closest clusters until only one cluster is left

HAC Example



HAC Algorithm

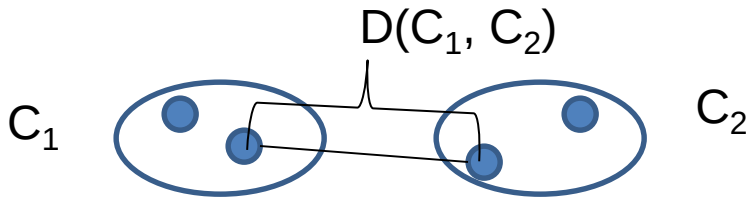
Start with all objects in their own cluster.
Repeat until there is only one cluster:
    Among the current clusters, determine the two clusters, c_i and c_j, that are closest
    Replace c_i and c_j with a single cluster $c_i \cup c_j$

Problem: we assume a distance/similarity function that computes distance/similarity between examples, but here we also need to compute distance between clusters.

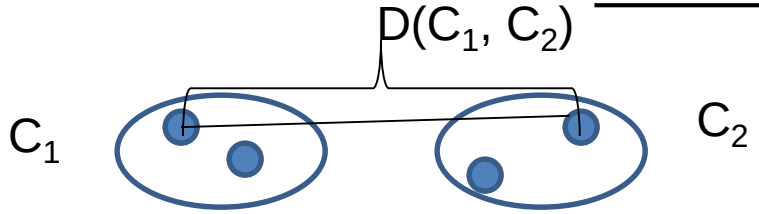
How?

Distance Between Clusters

- Assume a distance function that determines the distance of two objects: $D(x,x')$.
- There are multiple way to define a cluster distance function:
  - Single Link: distance of two closest members of clusters



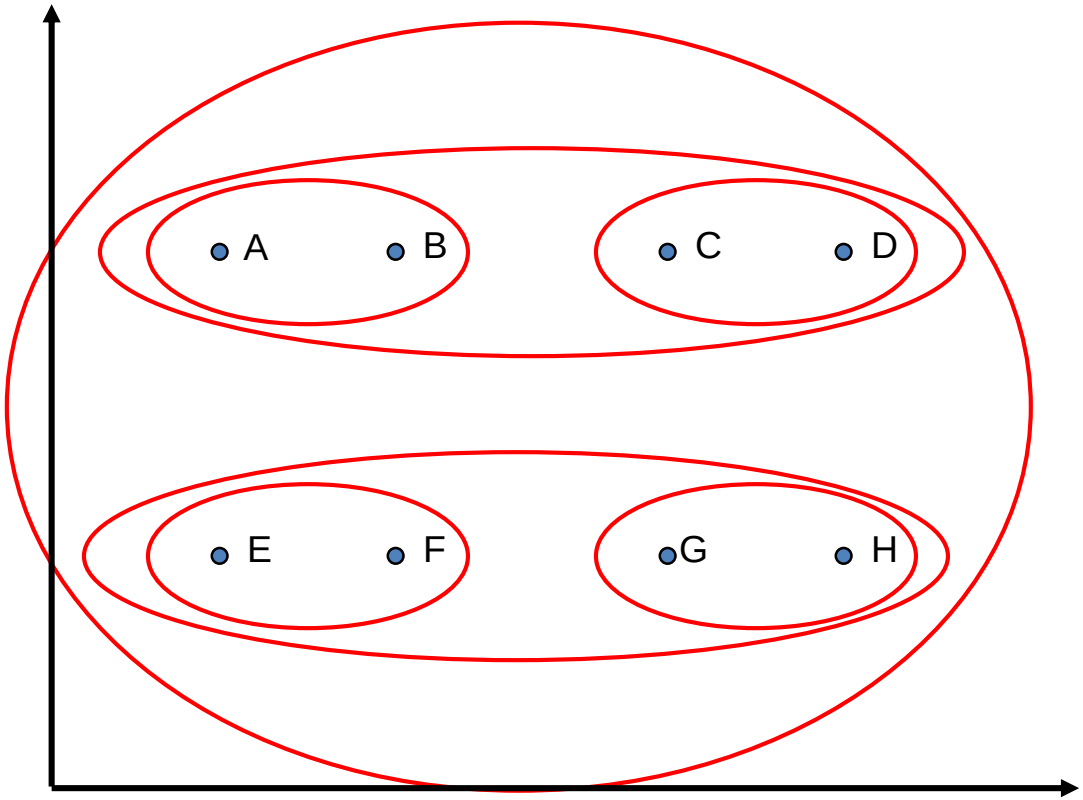
  - Complete Link: distance of two furthest members of clusters

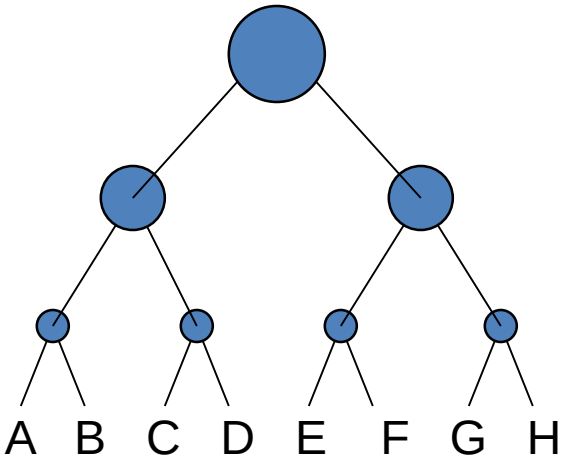


  - Average Link: average distance

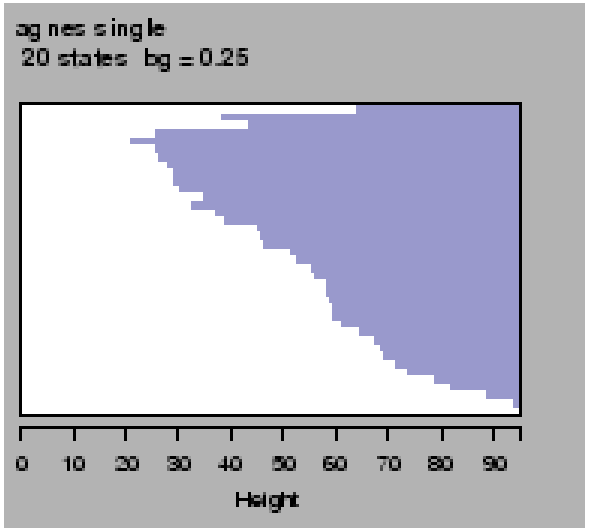
Single Link Agglomerative Clustering

- Use minimum distance of all pairs: $D(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{x}' \in C_j} D(\mathbf{x}, \mathbf{x}')$





Single-link's chaining effect

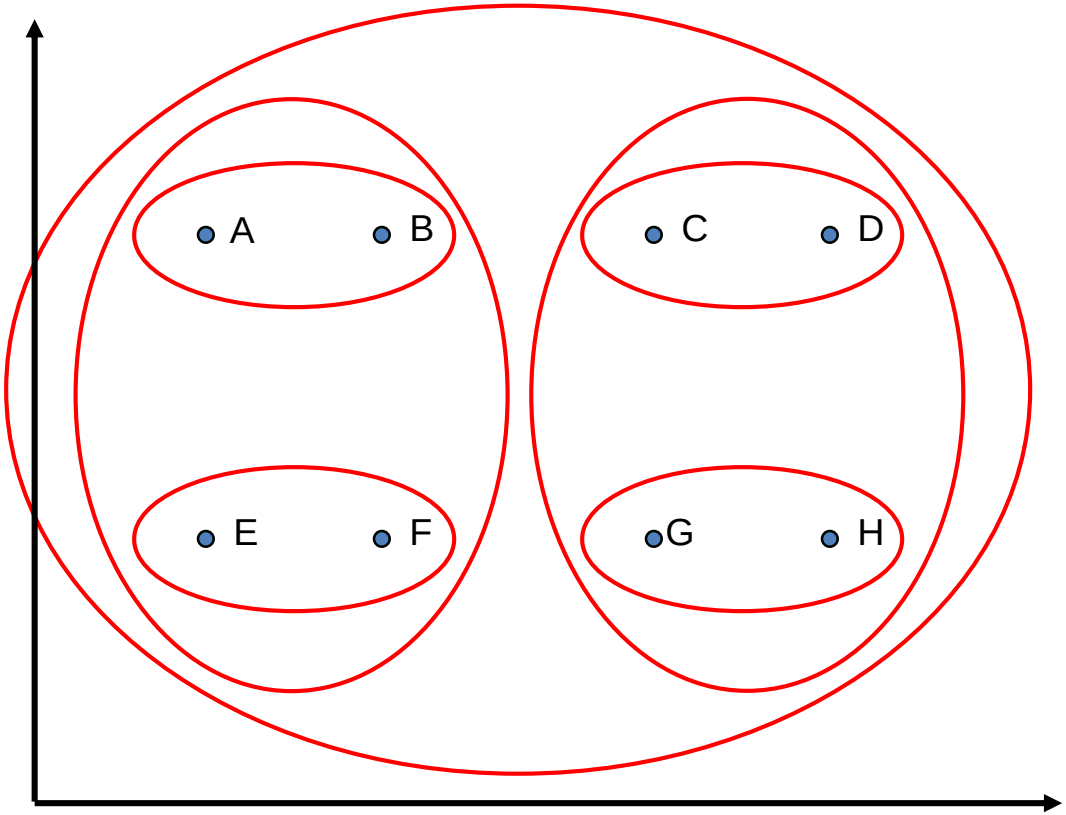


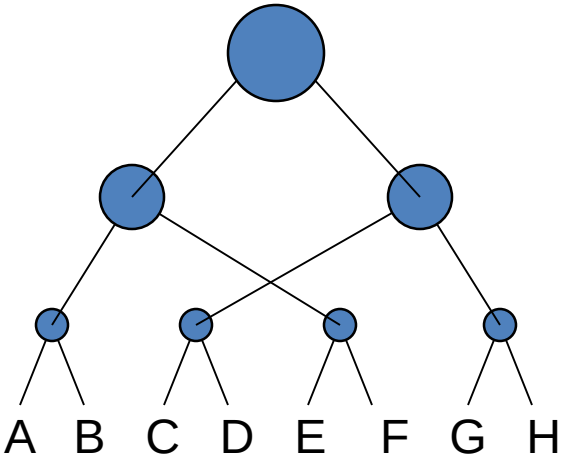
- Single link is famous for its chaining effect
- It can gradually adds more and more examples to the "chain"
- Create long straggling clusters

Complete Link

- Maximum distance of all pairs:

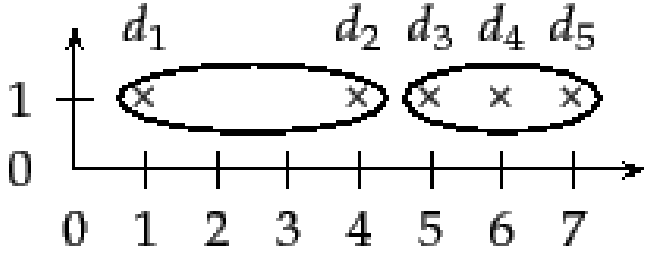
$$D(C_i, C_j) = \max_{x \in c_i, x' \in c_j} D(x, x')$$





- Makes “tight,” spherical clusters

Complete link is outlier sensitive



► **Figure 17.4** Outliers in complete-link clustering. The five documents have the x-coordinates $1+2\epsilon, 4, 5+2\epsilon, 6$ and $7-\epsilon$. Complete-link clustering creates the two clusters shown as ellipses. The most intuitive two-cluster clustering is $\{\{d_1\}, \{d_2, d_3, d_4, d_5\}\}$, but in complete-link clustering, the outlier d_1 splits $\{d_2, d_3, d_4, d_5\}$ as shown.

Updating the Cluster Distances after merging is a piece of

- After merging c_i and c_j, the distance of the resulting cluster to any other cluster, c_k, can be computed by:
  - Single Link:

$$D((c_i \cup c_j), c_k) = \min(D(c_i, c_k), D(c_j, c_k))$$

  - Complete Link:

$$D((c_i \cup c_j), c_k) = \max(D(c_i, c_k), D(c_j, c_k))$$

- This is **constant** time given previous distances

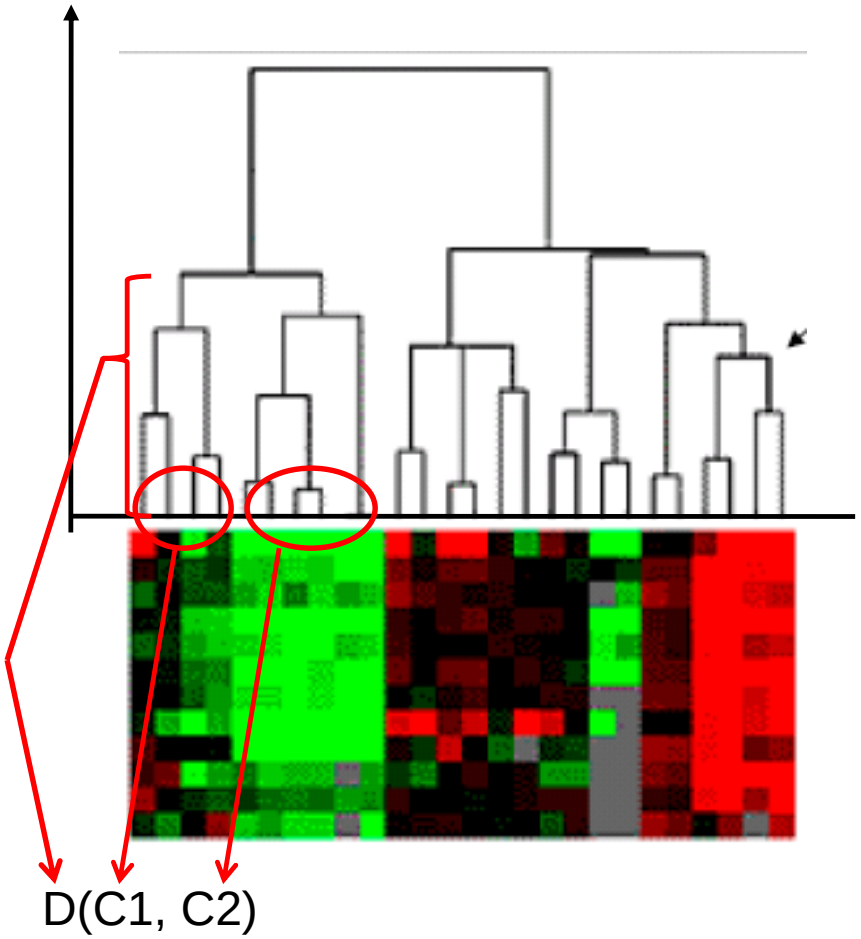
Average Link

- Basic idea: average similarity between members of the two clusters
- Averaged across all ordered pairs in the merged cluster

$$D(c_i, c_j) = \frac{1}{\left|c_i \cup c_j\right|\left(\left|c_i \cup c_j\right| - 1\right)} \sum_{\mathbf{x} \in (c_i \cup c_j)} \sum_{\mathbf{x}' \in (c_i \cup c_j) : \mathbf{x}' \neq \mathbf{x}} D(\mathbf{x}, \mathbf{x}')$$

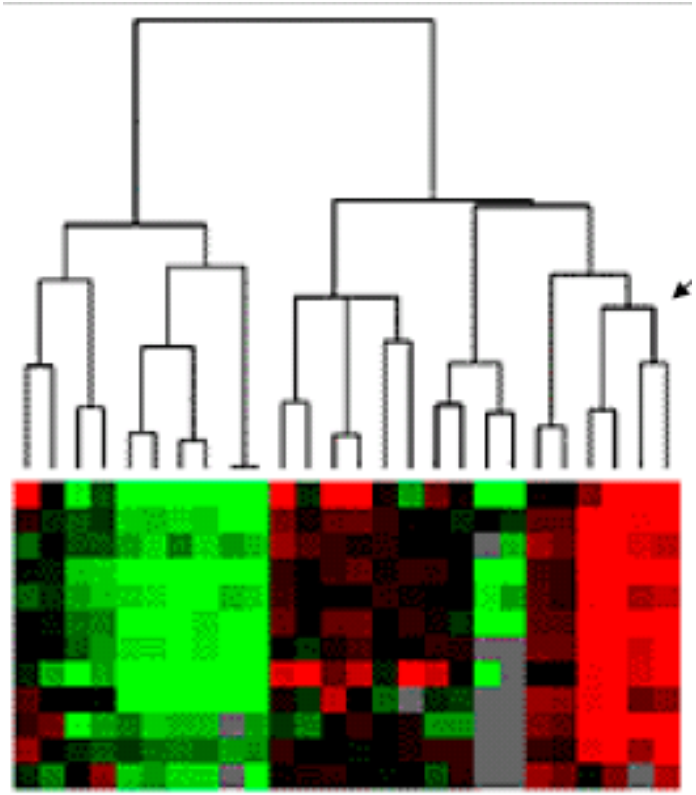
- Compared to single link and complete link:
  - Computationally more expensive – naively it can be $O(n^2)$ to compute the new distance between a pair of clusters
  - If we use cosine similarity, we can compute the new similarity in constant time
  - Achieves a compromise between single and complete link

HAC creates a Dendrogram



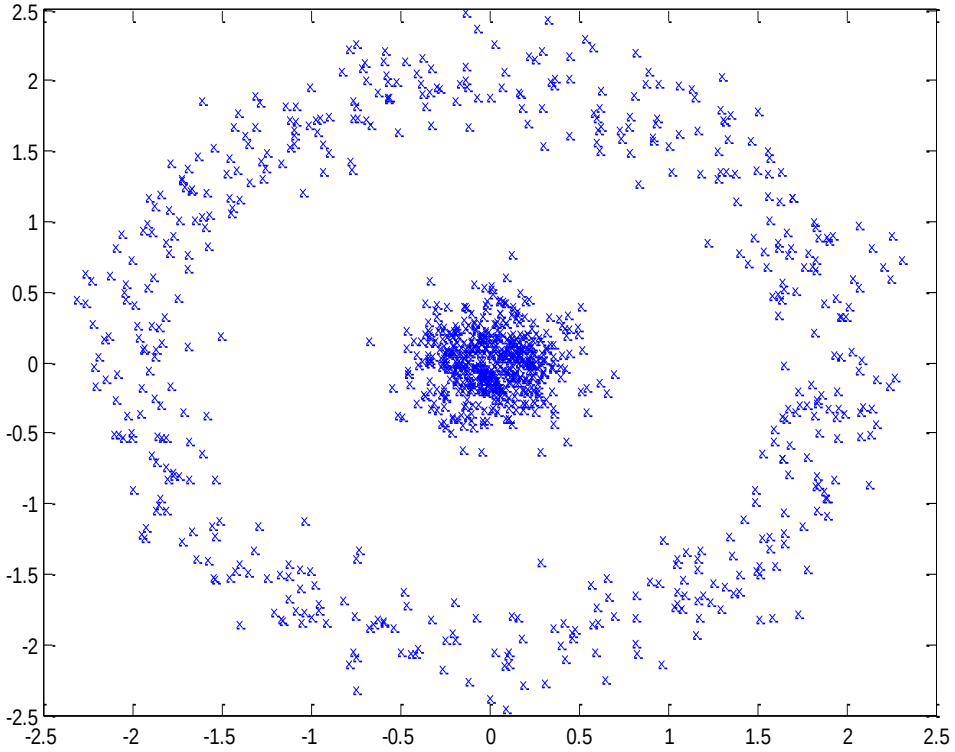
- Dendrogram draws the hierarchy as a tree such that the height of a tree branch = the distance between the two merged clusters at that particular step
- The distances are always monotonically increasing for single, complete and average links

How many clusters

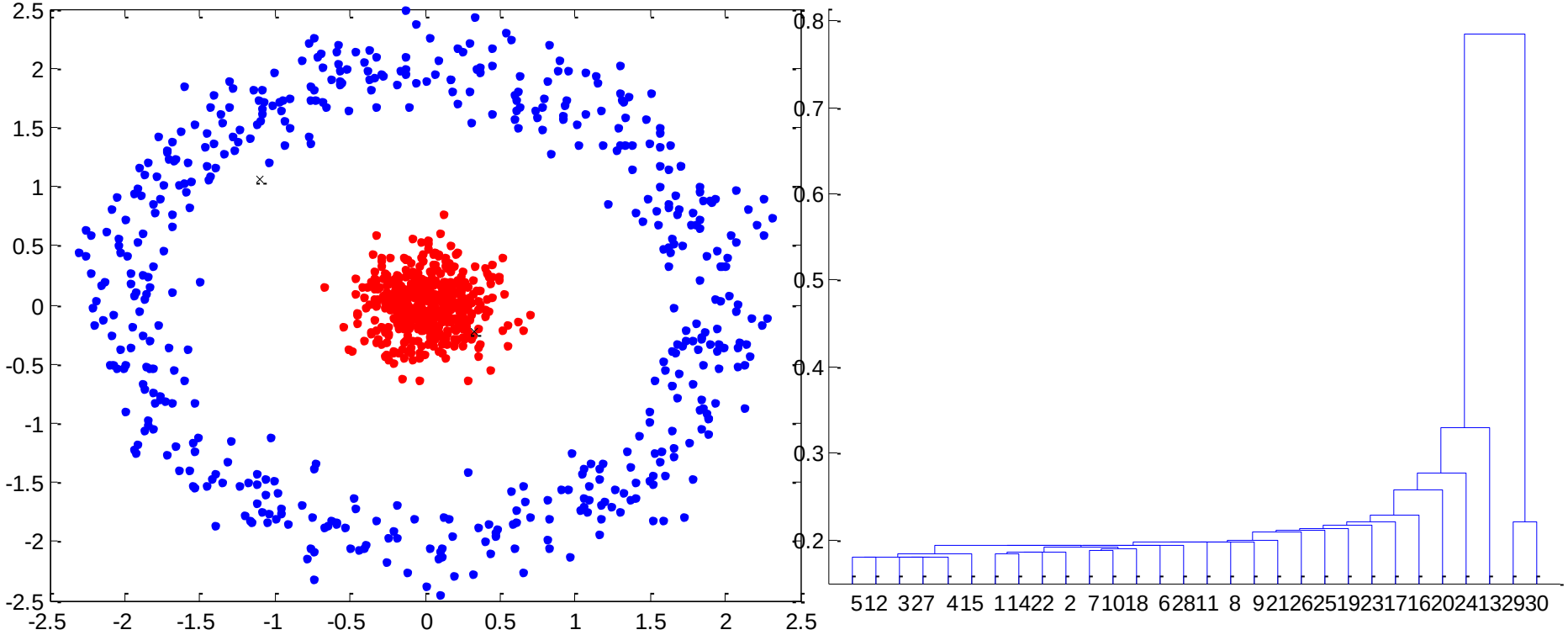


- Cutting the dendrogram at a specific similarity level gives us a simple flat clustering
- Different cutting places lead different numbers of clusters
- One option is to cut the dendrogram where the gap between two successive combination distances is largest

Example

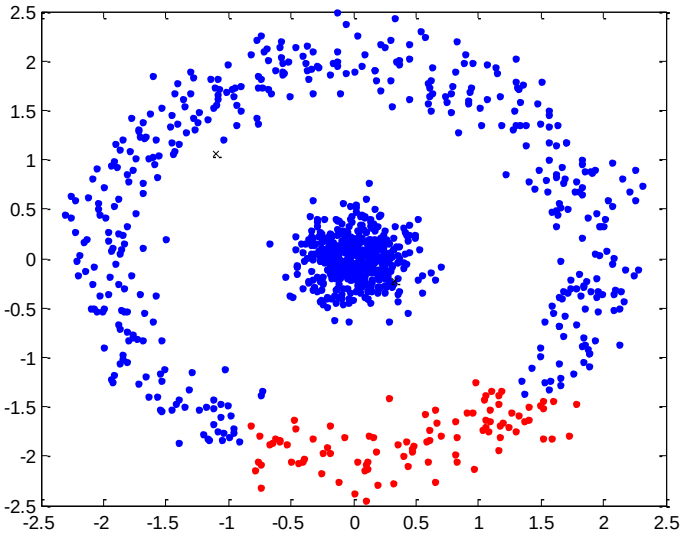


Single Link

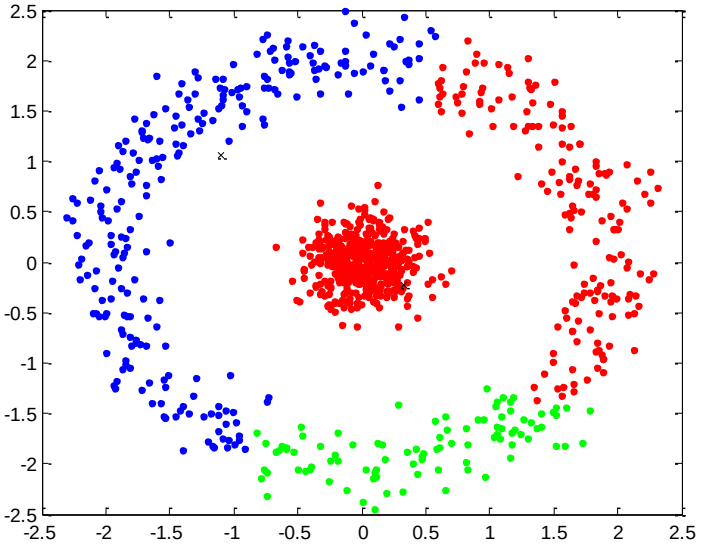

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
0.8
0.7
0.6
0.5
0.4
0.3
0.2
5 12 3 27 4 15 1 14 22 2 7 10 18 6 28 11 8 9 21 26 25 19 23 17 16 20 24 13 29 30

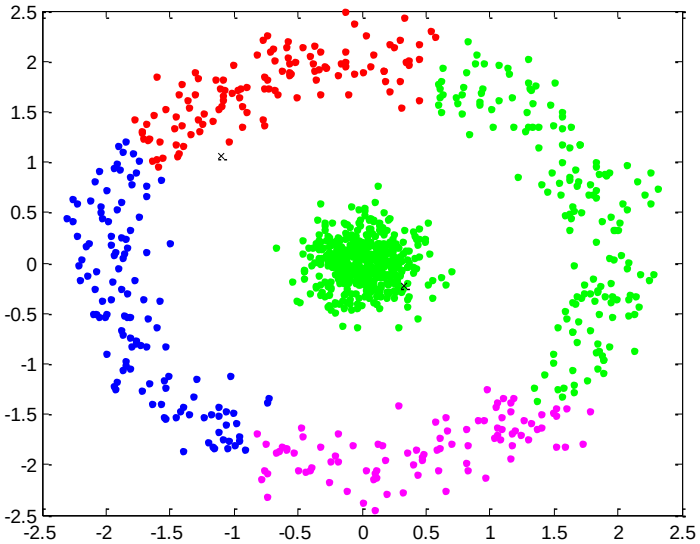
Complete Link

K=2

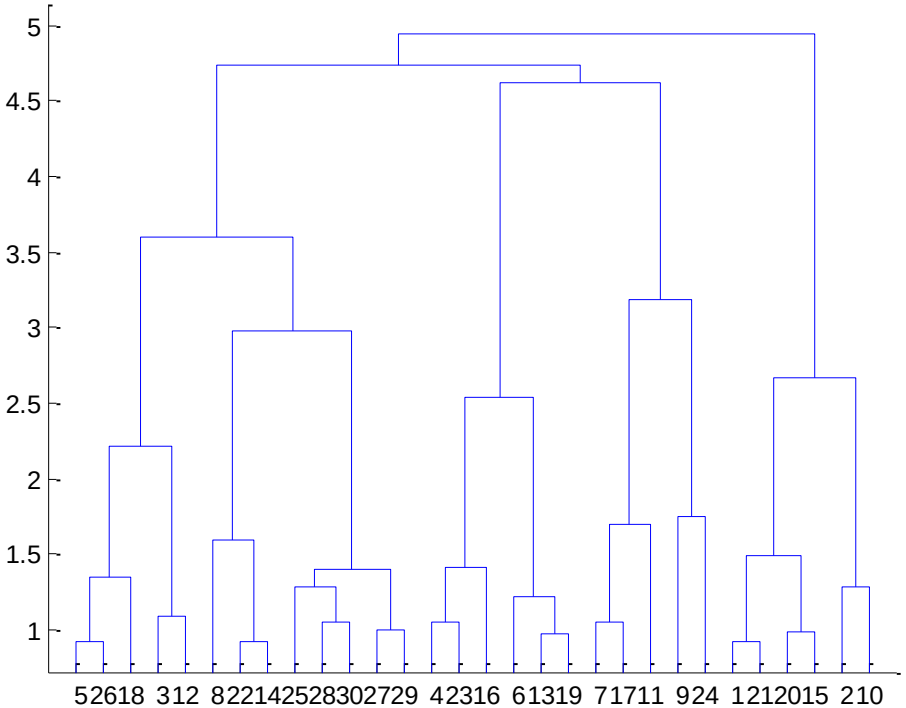


K=3

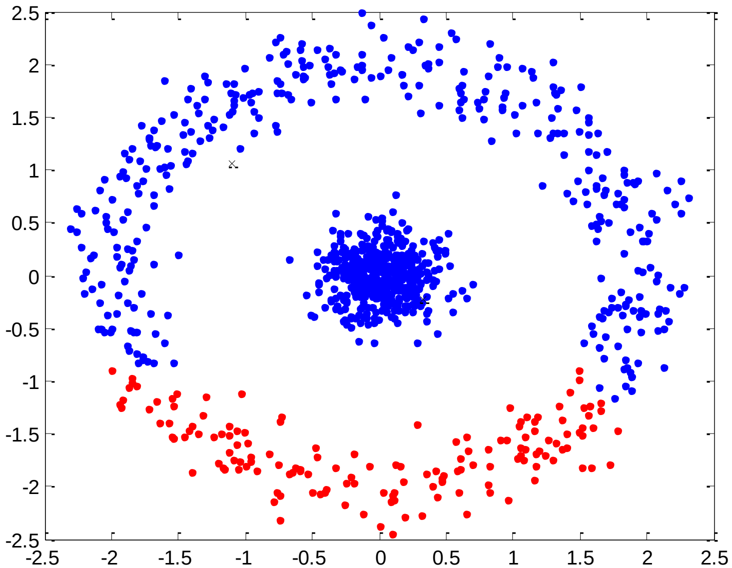


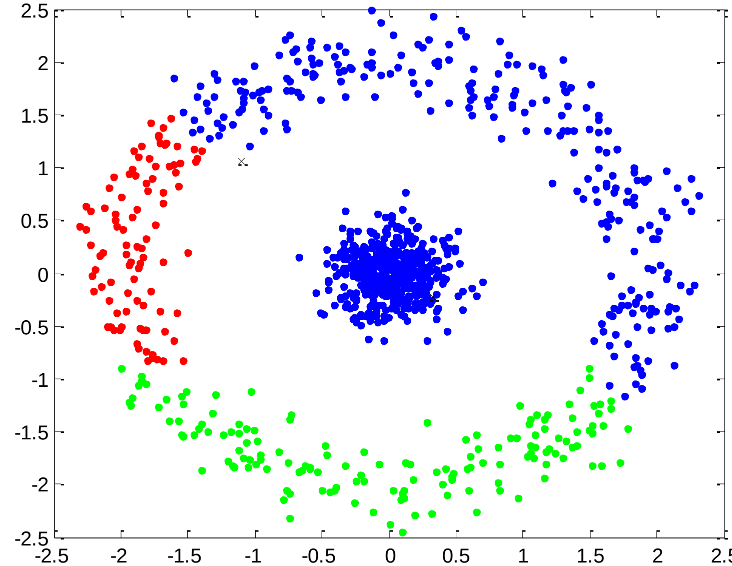


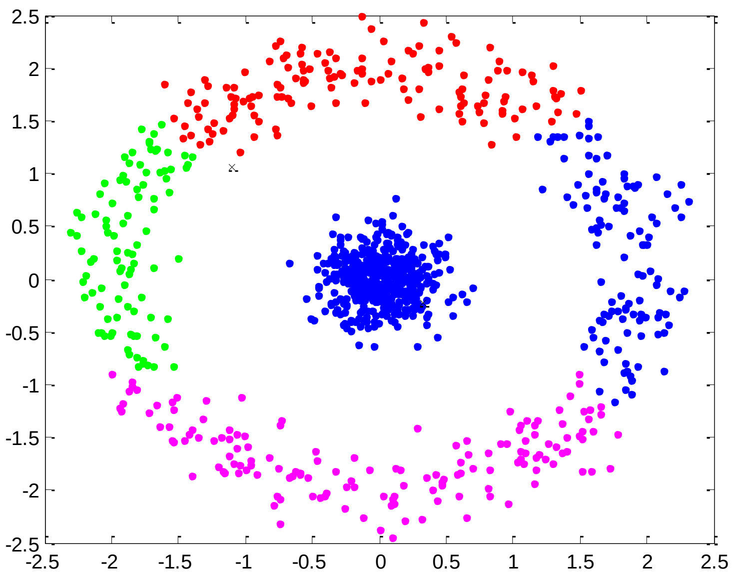
K=4

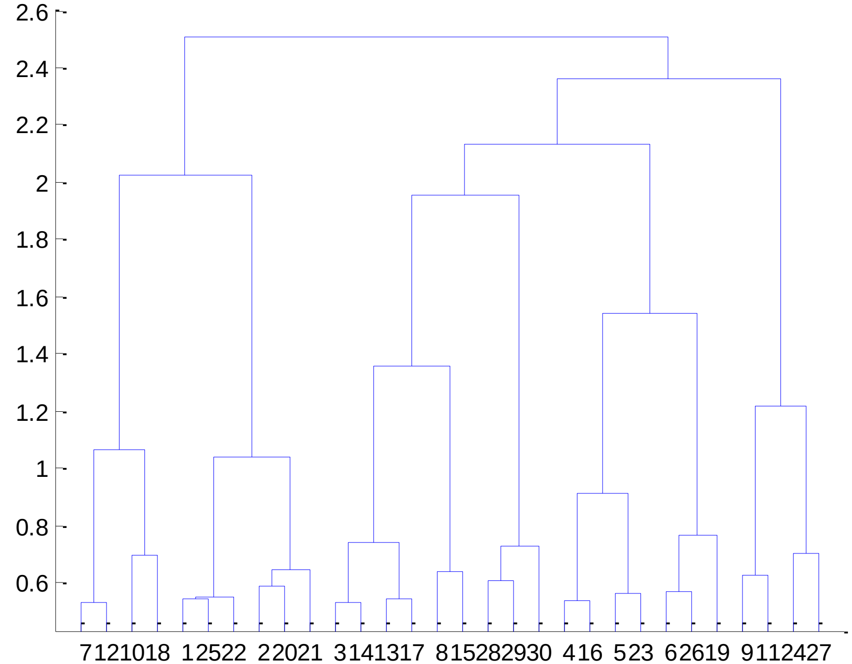


Average Link

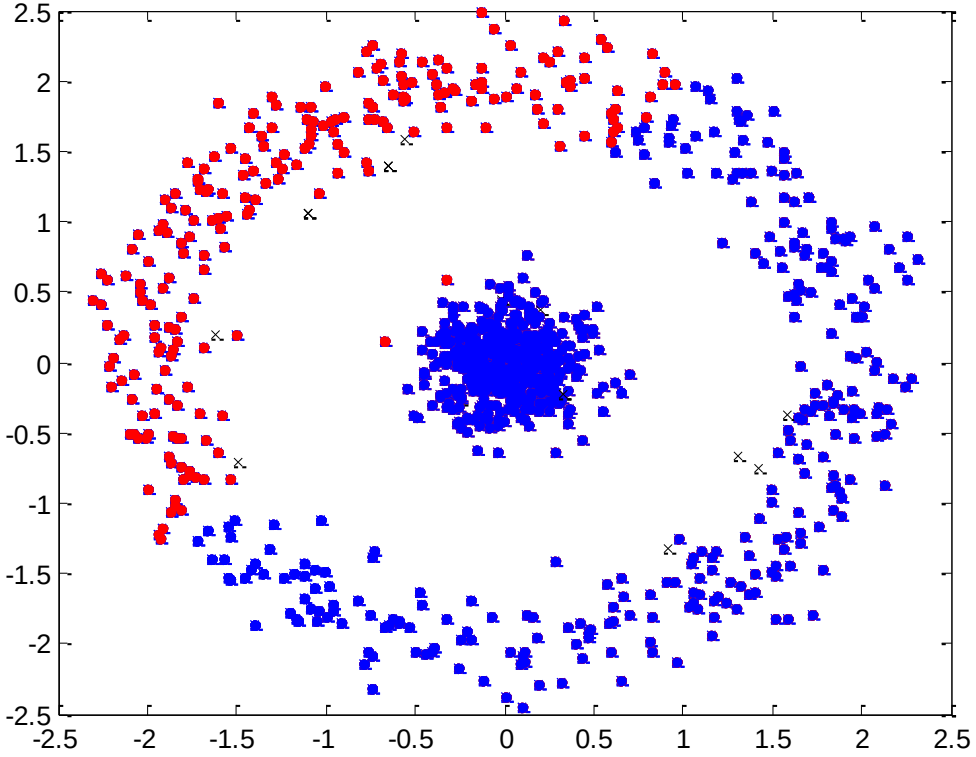








K-means

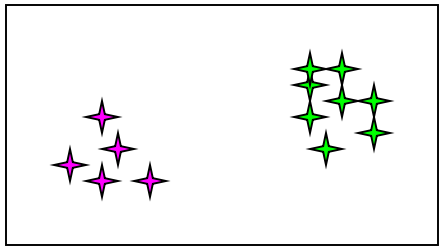


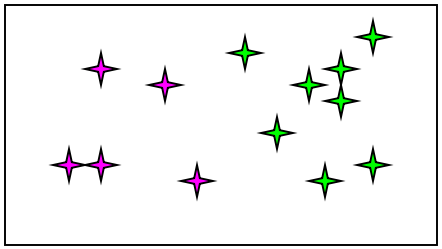
Comments on HAC

- HAC is a convenient tool that often provides interesting views of a dataset
- Primarily HAC can be viewed as an intuitively appealing clustering procedure for data analysis/exploration
- We can create clusterings of different granularity by stopping at different levels of the dendrogram
- HAC often used together with visualization of the dendrogram to decide how many clusters exist in the data
- Different linkage methods (single, complete and average) often lead to different solutions

How to Evaluate Clustering?

- By user interpretation
  - does a document cluster seem to correspond to a specific topic?
- Internal criterion – a good clustering will produce high quality clusters:
  - high within-cluster similarity: $s_w = \sum_{i=1}^{k} \sum_{x,x' \in c_i} sim(x, x')$
  - low between-cluster similarity: $s_b = \sum_{x \in c_i, x' \in c_j, i \neq j} sim(x, x')$
  - E.g., s_w / s_b

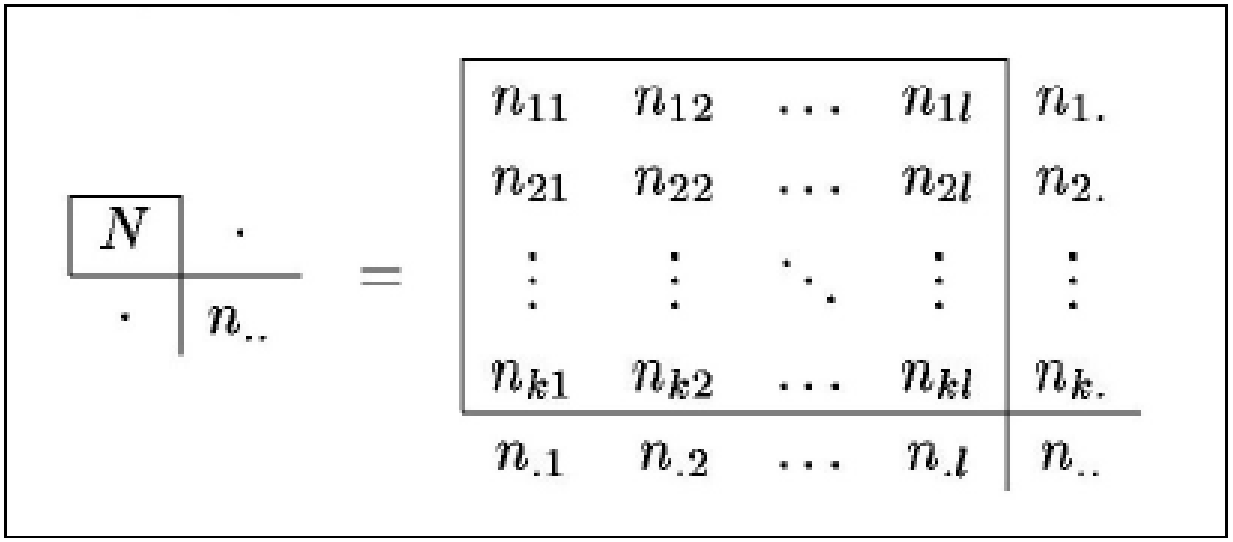




  - The measured quality of a clustering depends on both the object representation and the similarity measure used

External indexes

If true class labels (*ground truth*) are known, the validity of a clustering can be verified by comparing the class labels and clustering labels.



n_{ij} = number of objects in class *i* and cluster *j*

Rand Index and Normalized Rand Index

- Given partition (*P*) and ground truth (*G*), measure the number of vector pairs that are:
  - *a*: in the same class both in *P* and *G*.
  - *b*: in the same class in *P*, but different classes in *G*.
  - *c*: in different classes in *P,* but in the same class in *G*.
  - *d*: in different classes both in *P* and *G*.

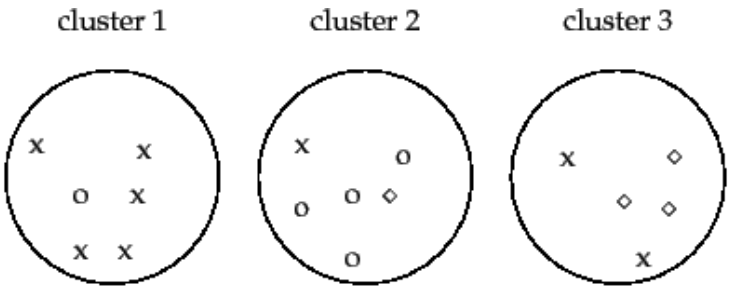
$$R = \frac{a+d}{a+b+c+d}$$

- Adjusted rand index: corrected-for-chance version of rand index
  - Compare to the expectation of the index assuming a random partition of the same cluster sizes

$$ARI = \frac{Index - ExpectedR}{MaxIndex - ExpectedR} = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}$$

Purity and Normalized Mutual Information

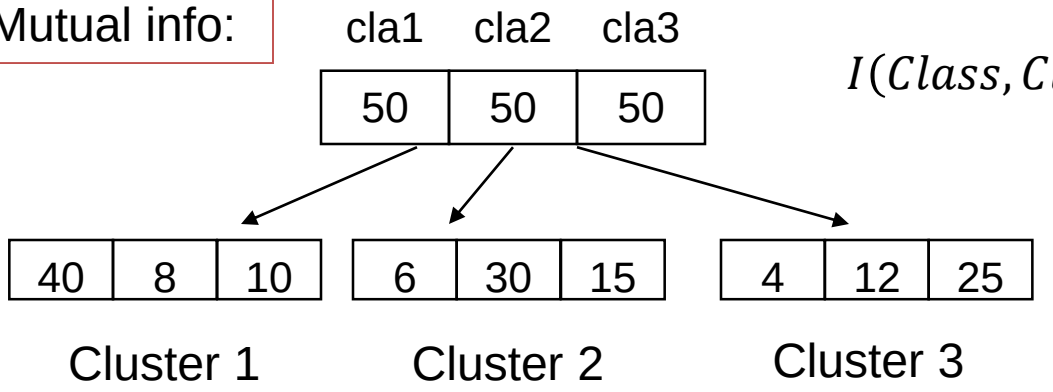
- Purity



► **Figure 16.1** Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ◇, 3 (cluster 3). Purity is $(1/17) \times (5+4+3) \approx 0.71$.

- Normalized Mutual Information

Mutual info:



$$I(Class, Clust) = H(Class) - H(Class|Clust)$$

$$NMI = \frac{2I(Class, Clust)}{H(Clust) + H(Class)}$$