TEXT CLUSTERING

Prof. Marco Viviani

marco.viviani@unimib.it

INTRO

What is clustering?

- (Document) Clustering: the process of grouping a set of objects (documents) into classes of similar objects (documents).
  - Documents within a cluster should be similar.
  - Documents from different clusters should be dissimilar.

- The commonest form of unsupervised learning.
  - Unsupervised learning = learning from raw data.
    - Opposed to supervised learning where a classification of examples is given.
  - A common and important task that finds many applications Text Mining and NLP tasks.

Clustering VS Classification

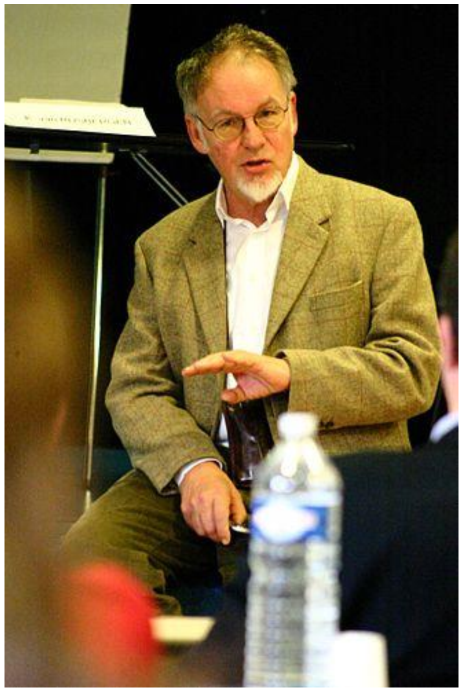
Classification	Clustering
Supervised learning	Unsupervised learning
Classes are human-defined and part of the input to the learning algorithm	Clusters are inferred from the data without human input
Output = membership in class only	Output = membership in class + distance/similarity from/with the centroid ("degree of cluster membership")

The cluster hypothesis

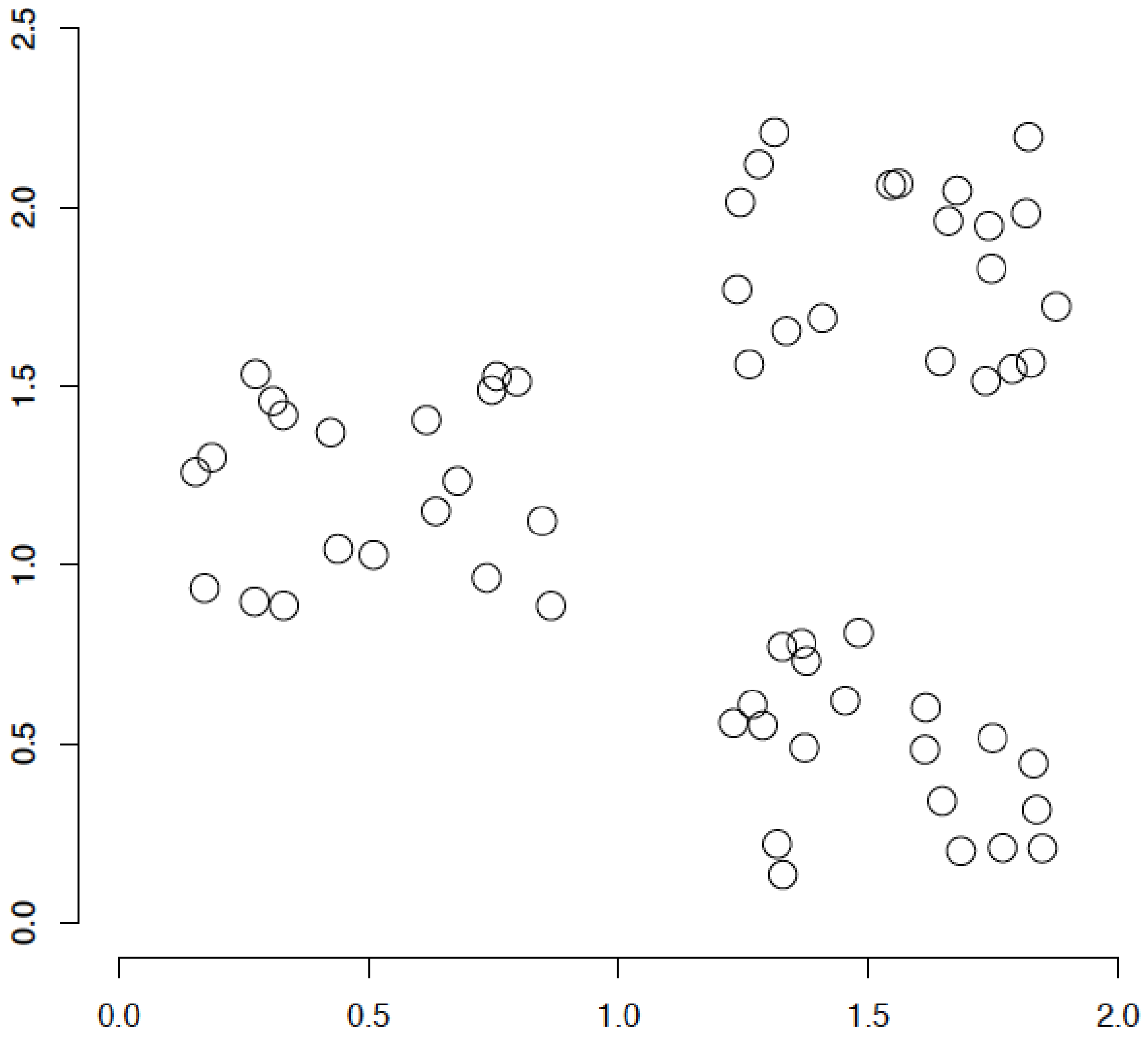
Documents in the same cluster behave similarly with respect to relevance to information needs.

- All applications of clustering in Information Retrieval (IR) are based (directly or indirectly) on the cluster hypothesis.

- C. J. van Rijsbergen (1979): «closely-associated documents tend to be relevant to the same requests».



A data set with clear cluster structure



- How would you design an algorithm for finding the three clusters in this case?

→ DISTANCE

BASIC ISSUES AND NOTIONS

Issues for clustering

- Representation for clustering
  - Document representation.
    - Vector space? Normalization?
  - Need a notion of similarity/distance.

- How many clusters?
  - Fixed a priori?
  - Completely data-driven?
    - Avoid “trivial” clusters - too large or small.

Notion of similarity/distance

- Ideal: semantic similarity.
- Practical: term-statistical similarity.
  - Docs as vectors.
  - For many algorithms, easier to think in terms of a distance (rather than similarity) between docs.
  - We will mostly speak of Euclidean distance.
    - But real implementations use cosine similarity.
- Today: towards semantic similarity.
  - Possibility of using Word Embedding or Contextualized Word Embedding vectors.

Clustering algorithms

- Flat algorithms
  - Flat clustering creates a flat set of clusters without any explicit structure that would relate clusters to each other.
  - Usually start with a random (partial) partitioning.
  - Refine it iteratively:
    - k-means clustering.
    - Model-based clustering.

- Hierarchical algorithms
  - Hierarchical clustering creates a hierarchy of clusters.
  - Bottom-up, agglomerative.
  - Top-down, divisive.

“Hard” VS “soft” clustering

- Hard clustering: Each document belongs to exactly one cluster.
  - More common and easier to do.

- Soft clustering: A document can belong to more than one cluster (in a soft assignment, a document has fractional membership in several clusters).
  - Makes more sense for applications like creating browsable hierarchies.
  - You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes.
    - You can only do that with a soft clustering approach.

FLAT CLUSTERING

Problem statement (1)

We can define the goal in hard flat clustering as follows.

- Given:
  - (i) a set of documents $D = \{d_1, d_2, \dots, d_N\}$,
  - (ii) a desired number of clusters k,
  - (iii) an objective function that evaluates the quality of a clustering.

- We want to compute an assignement

$$\gamma : D \rightarrow \{\omega_1, \omega_2, \dots, \omega_k\}$$

that minimizes (or, in other cases, maximizes) the objective function.

Problem statement (2)

- In most cases, we also demant that γ is surjective, i.e., that none of the k clusters is empty.

- The objective function is often defined in terms of similarity or distance between documents.

- For (textual) documents, the type of similarity we want is usually topic similarity or high values on the same dimensions in the Vector Space Model.
    - E.g., documents about China have high values on dimensions like Chinese, Beijing, and Mao, whereas documents about the UK tend to have high values for London, Britain, and King.

k-means

- k-means is the most important flat clustering algorithm.
- Assumption: documents are represented as length-normalized vectors in a real-valued space in the familiar way.
- Its objective is to minimize the average squared Euclidean distance of documents from their cluster centers.

Euclidean distance

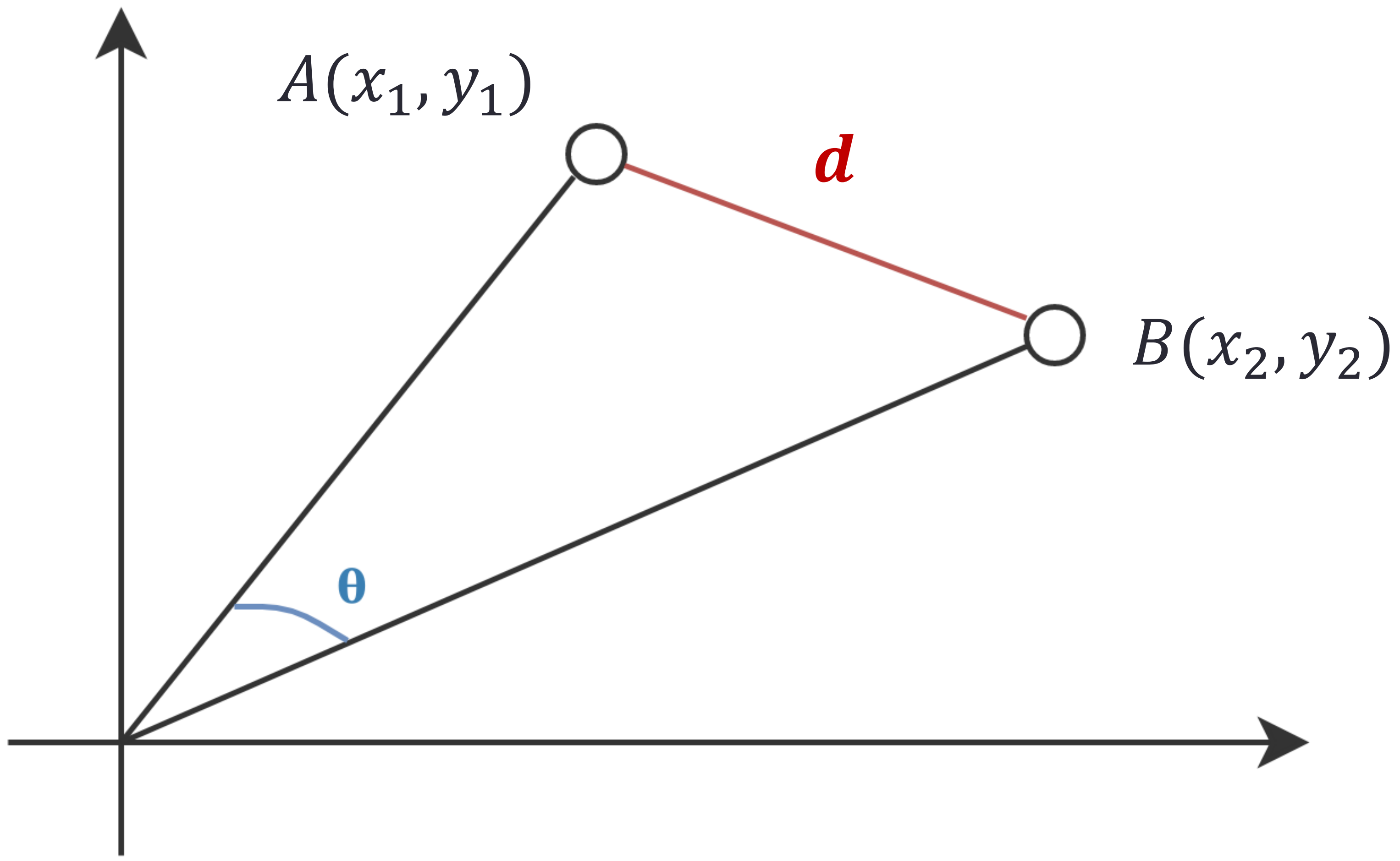
- The Euclidean distance between two vectors $\vec{u}$ and $\vec{v}$ is defined as:

$$d(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\|$$

$$= \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$$

Distance and Similarity



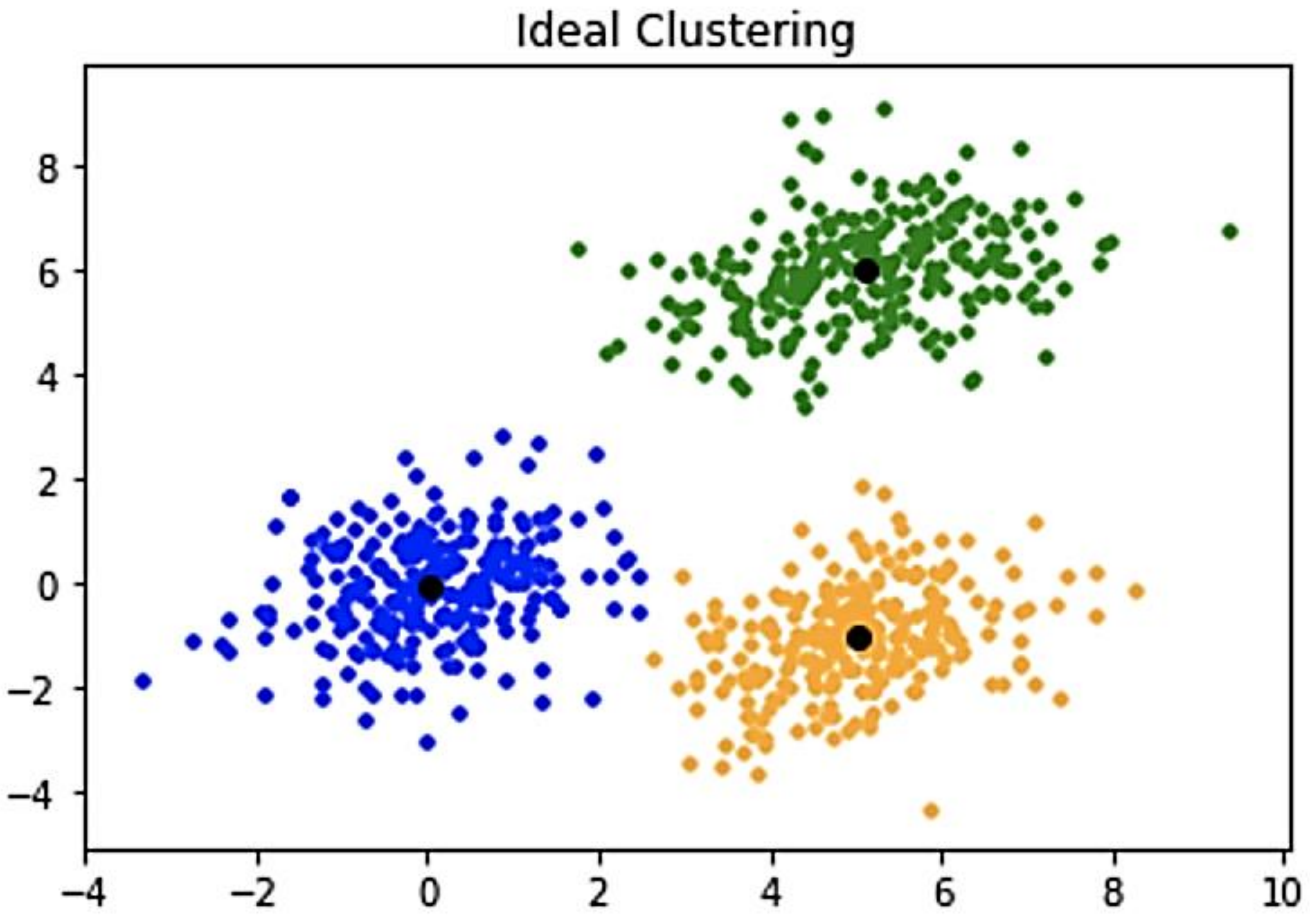
k-means – Centroid

- A cluster center is defined as the centroid (or mean, or center of gravity) $\overrightarrow{\mu}$ of the documents in a cluster ω:

$$\overrightarrow{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities).

Ideal clustering



k-means algorithm (1)

- The first step of k-means is to select as initial cluster centers, k randomly selected documents, i.e., the seeds: $\{s_1, s_2, \dots, s_k\}$.

- For each cluster ω_j, $s_j = \vec{\mu}(\omega_j)$.

- For each doc d_i:
  - Assign d_i to the cluster ω_j such that $dist(d_i, s_j)$ is minimal.

k-means algorithm (2)

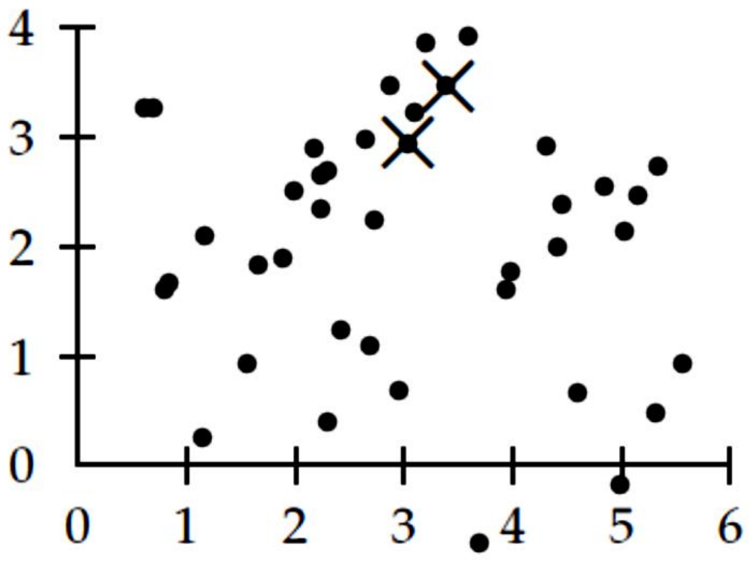
- The algorithm then moves the cluster centers around in space in order to minimize distance.

- This is done iteratively by repeating two steps until a stopping criterion is met:
  1. Reassigning documents to the cluster with the closest centroid.
  2. Recomputing each centroid based on the current members of its cluster.

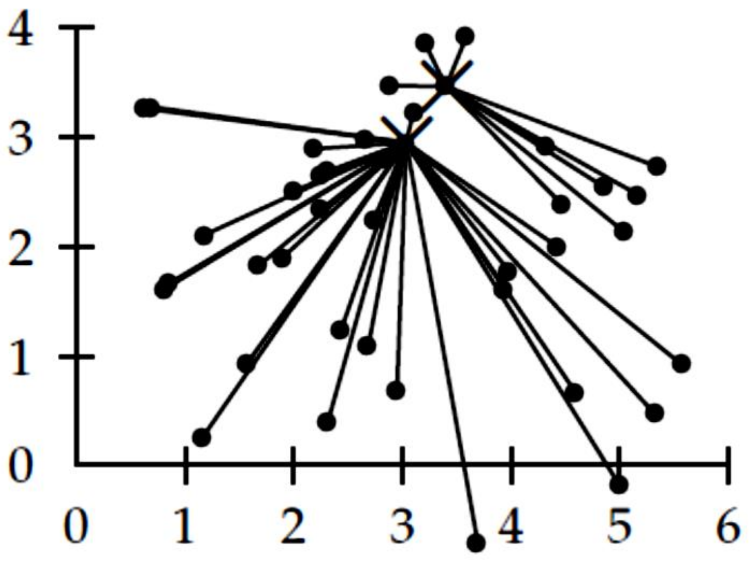
k-means algorithm (3)

```
K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
for $k \leftarrow 1$ to $K$
do $\vec{\mu}_k \leftarrow \vec{s}_k$
while stopping criterion has not been met
do for $k \leftarrow 1$ to $K$
      do $\omega_k \leftarrow \{\}$
      for $n \leftarrow 1$ to $N$
      do $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
         $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ (reassignment of vectors)
      for $k \leftarrow 1$ to $K$
      do $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ (recomputation of centroids)
return $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$
```

k-means example (1)

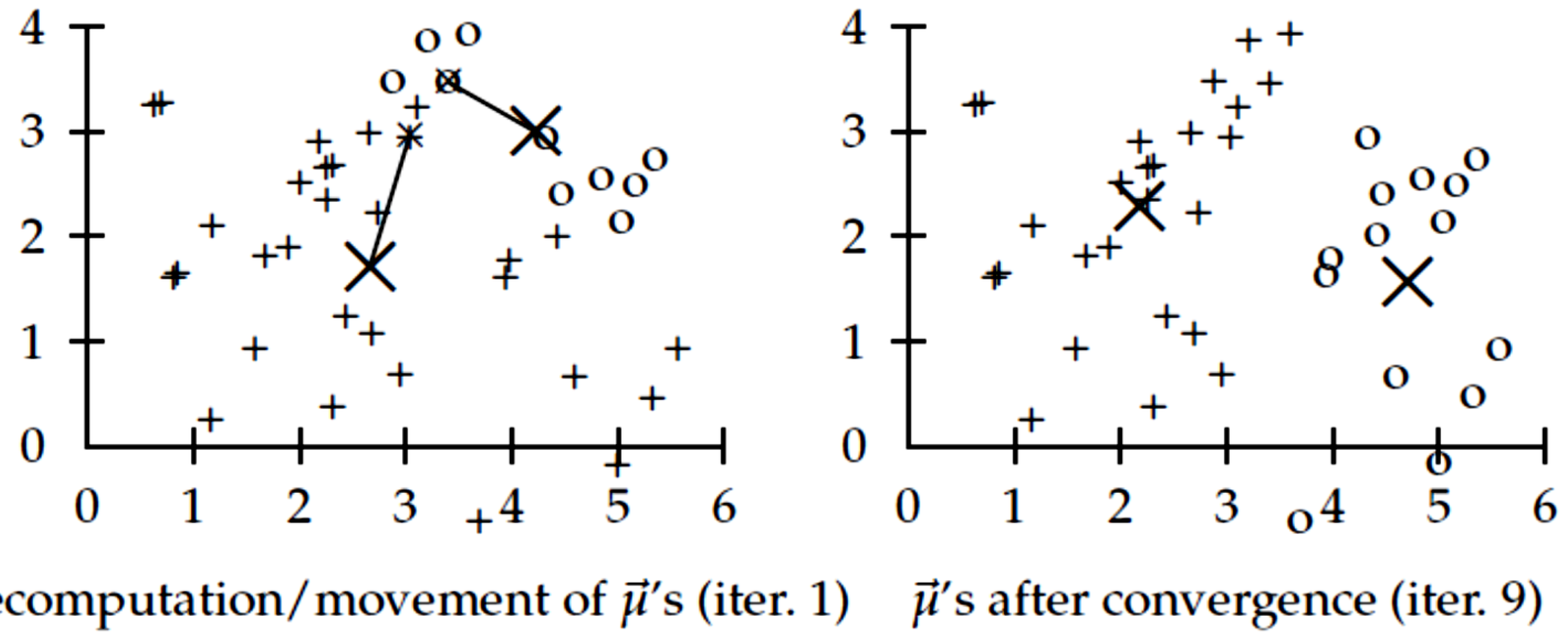


selection of seeds

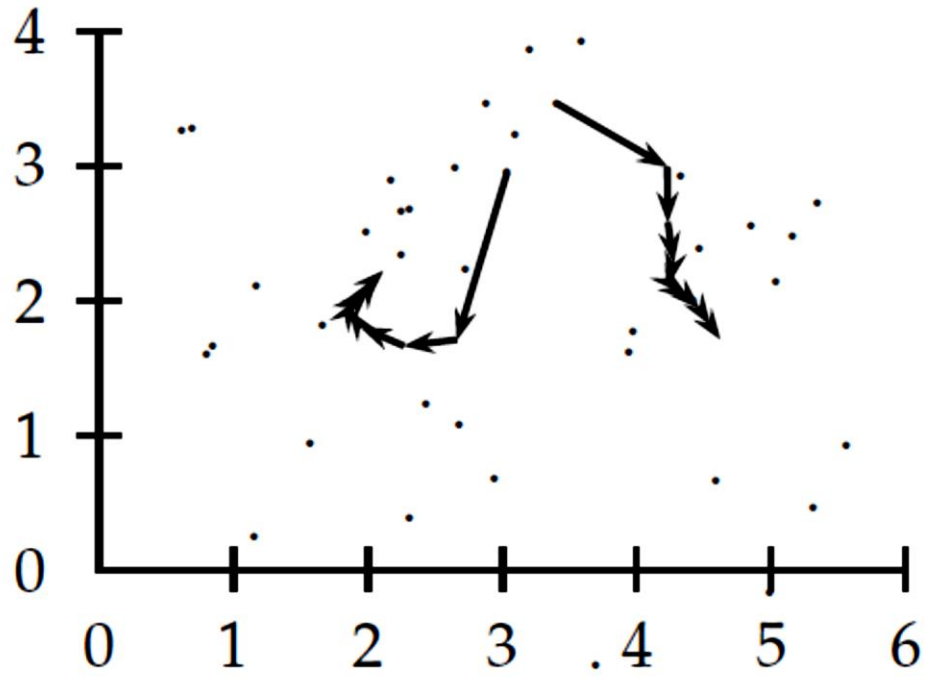


assignment of documents (iter. 1)

k-means example (2)



k-means example (3)



movement of $\vec{\mu}$'s in 9 iterations

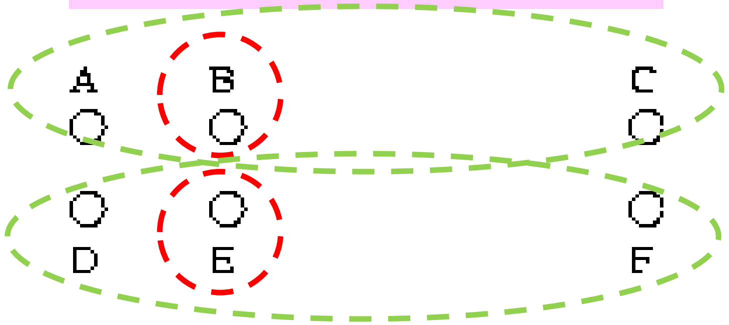
Termination conditions

- Several possibilities, e.g.,
  - A fixed number of iterations → insufficient number of iteration (poor quality).
  - Doc partition unchanged → good clustering, runtime could be too long.
  - Centroid positions do not change.
  - The distance between documents and centroids falls below a certain threshold.

Seed Choice

- Results can vary based on random seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  - Select good seeds using a heuristic (e.g., doc least similar to any existing mean).
  - Try out multiple starting points.
  - Initialize with the results of another method.

Example showing sensitivity to seeds

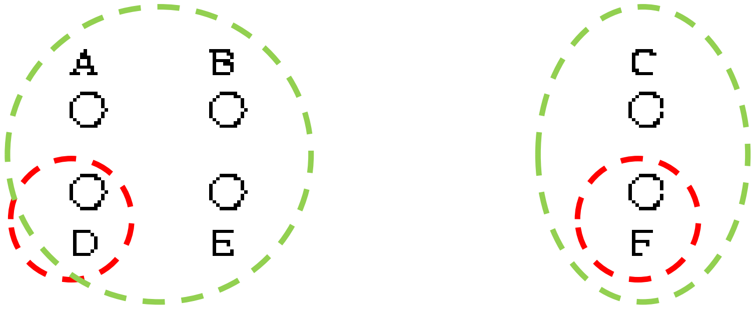


- In the above, if we start with B and E as centroids we converge to {A,B,C} and {D,E,F}

Seed Choice

- Results can vary based on random seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  - Select good seeds using a heuristic (e.g., doc least similar to any existing mean).
  - Try out multiple starting points.
  - Initialize with the results of another method.

Example showing sensitivity to seeds



- If we start with D and F you converge to {A,B,D,E} {C,F}

How many clusters?

- Number of clusters k is given.
  - Partition n docs into predetermined number of clusters.

- k not specified in advance.
  - Finding the "right" number of clusters is part of the problem.
    - Given docs, partition into an "appropriate" number of subsets.
    - Trade-off between having more clusters (better focus within each cluster) and having too many clusters.
    - E.g., for query results - ideal value of k not known up front - though UI may impose limits.

k-means and Python: A simple example

1. Fetch some textual documents;

2. Represent each textual document as a vector;

3. Perform k-means clustering;

4. Evaluate qualitatively the result of the clustering.

Phase 1: Fetch some textual documents

- The simplest solution:

```
documents = ["This little kitty came to play when I was
             eating at a restaurant.", "Merley has the
             best squooshy kitten belly.", "Google
             Translate app is incredible.", "If you
             open 100 tab in google you get a smiley
             face.", "Best cat photo I've ever
             taken.", "Climbing ninja cat.",
             "Impressed with google map feedback.",
            "Key promoter extension for Google
             Chrome."]
```

Phase 2: Represent each doc as a vector

```
from sklearn.feature_extraction.text import
TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words={'english'})

X = vectorizer.fit_transform(documents)
```

Phase 3: Perform *k*-means clustering

```
from sklearn.cluster import Kmeans

k = 4  #or any other number

Labels = model.labels_

model = KMeans(n_clusters = k, init = 'k-means++’,
        max_iter = 100, n_init = 1)

model.fit(X)
```

Phase 4: Qualitative analysis

```
import pandas as pd

clusters = pd.DataFrame(list(zip(documents,labels)),
columns = ['document','cluster’])

print(documents.sort_values(by = ['cluster']))
```

```
                                        document  cluster
0  This little kitty came to play when I was eati...        0
3  If you open 100 tab in google you get a smiley...        0
4                  Best cat photo I've ever taken.        1
5                              Climbing ninja cat.        1
2                 Google Translate app is incredible.        2
6                 Impressed with google map feedback.        2
7            Key promoter extension for Google Chrome.        2
1            Merley has the best squooshy kitten belly.        3
```

Flat clustering with embeddings?

- Flat clustering does not care where the vectors come from → It just needs numeric representations of your data points in some **embedding space**.

Task	Typical Embedding	Description
Word clustering	Word2Vec, GloVe, fastText	Each word → 1 vector (static)
Document/sentence clustering	BERT, Sentence-BERT, etc.	Each document, sentence, (or paragraph) → 1 vector (contextualized)

- Flat clustering with semantic vectors is not only possible, but also often **better** than clustering TF-IDF vectors, because:
  - Semantic embeddings capture meaning and synonymy;
  - They are dense and continuous, unlike sparse TF-IDF vectors;
  - They enable context-aware grouping.

HIERARCHICAL CLUSTERING

Hierarchical Clustering (HC)

- Hierarchical outputs a hierarchy, a structure that is more informative than the unstructured set of clusters returned by flat clustering.

- Hierarchical clustering does not require to prespecify the number of clusters.

- Advantages of hierarchical clustering come at the cost of lower efficiency.
  - The most common hierarchical clustering algorithms have a complexity that is at least quadratic in the number of documents compared to the linear complexity of k-means.

Bottom-up and top-down HC

- Hierarchical clustering algorithms are either bottom-up or top-down.

- Bottom-up clustering treat each document as a singleton cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all documents.
  - It is therefore called Hierarchical Agglomerative Clustering or HAC.

- Top-down clustering requires a method for splitting a cluster. It proceeds by splitting clusters recursively until individual documents are reached.
  - It is therefore called Hierarchical Divisive Clustering or HDC.

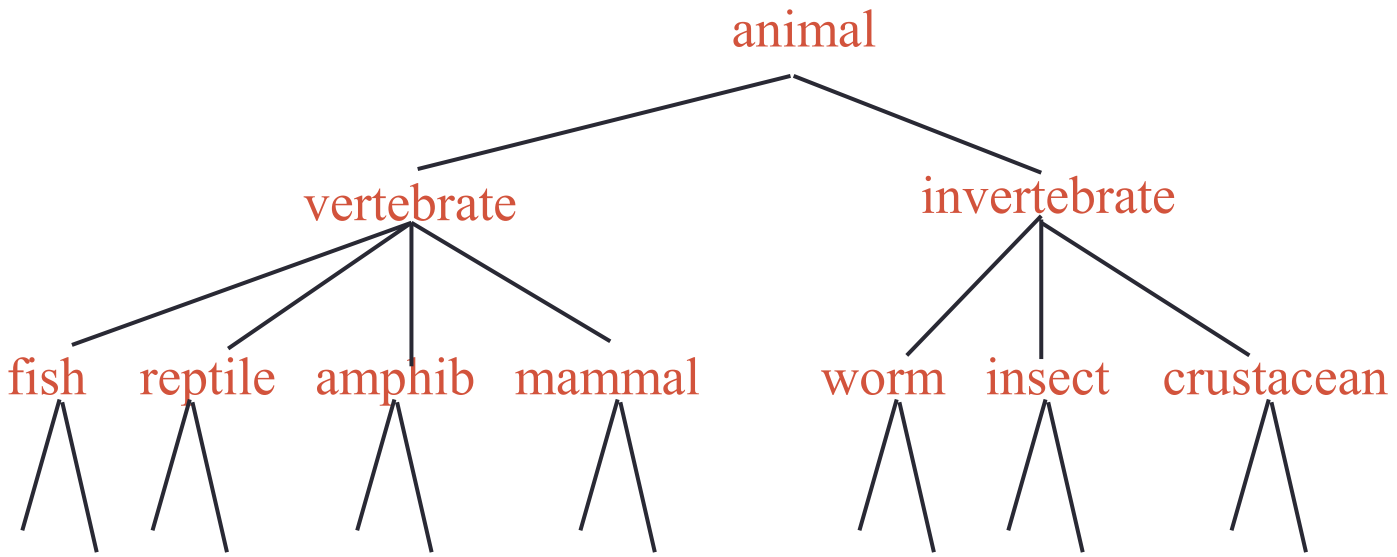
Hierarchical Agglomerative Clustering (HAC)

- Starts with each doc in a separate cluster.
  - Then repeatedly joins the closest pair of clusters, until there is only one cluster.
  - Hierarchical clustering employs a measure of distance/similarity to create new clusters.

- The history of merging forms a binary tree or hierarchy.

Dendrogram (1)

- An HAC clustering is typically visualized as a dendrogram.
    - It builds a tree-based hierarchical taxonomy (dendrogram) from a set of documents.

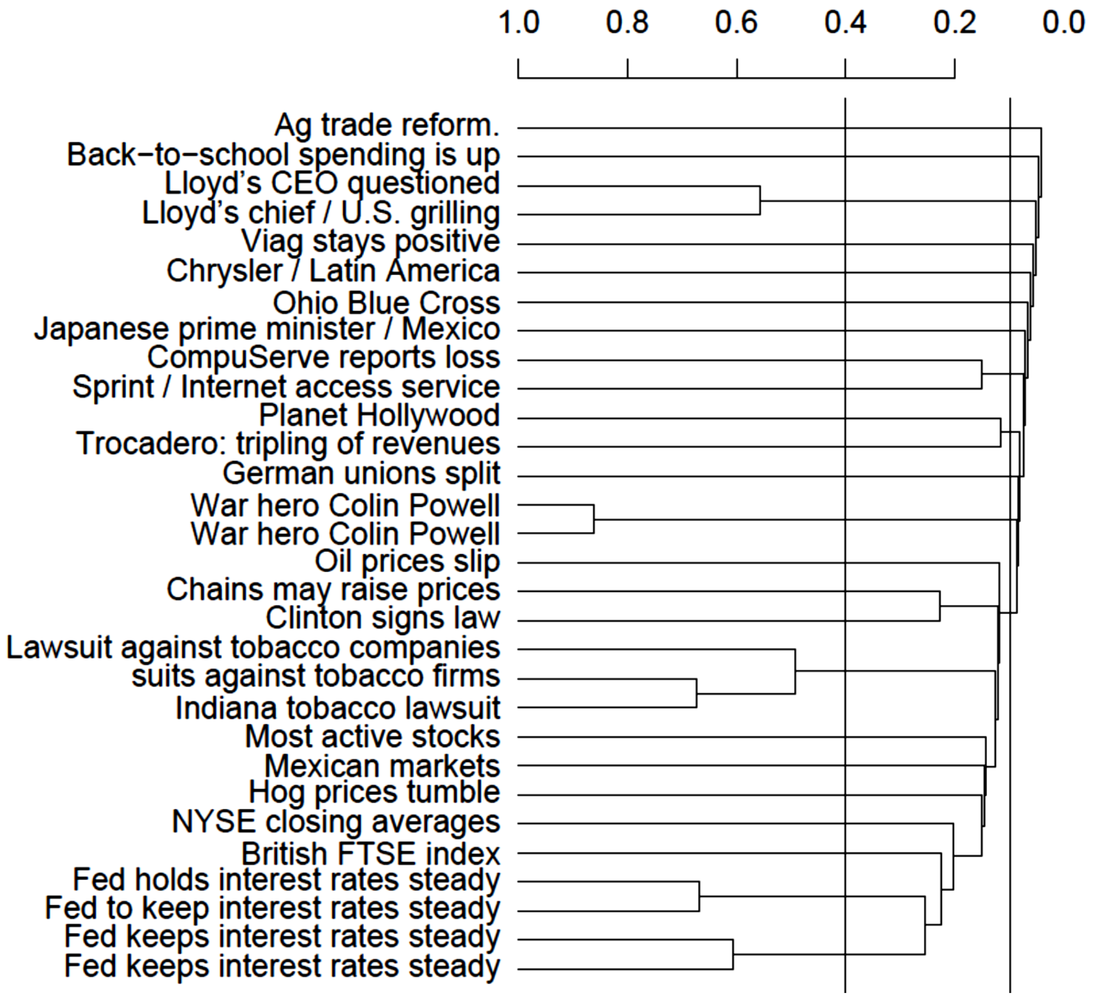


Dendrogram (2)

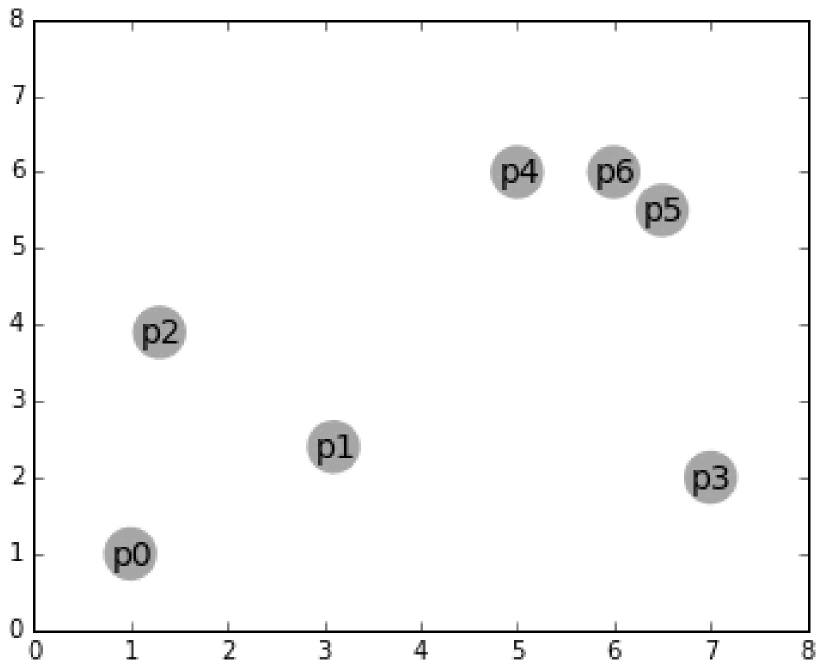
Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.

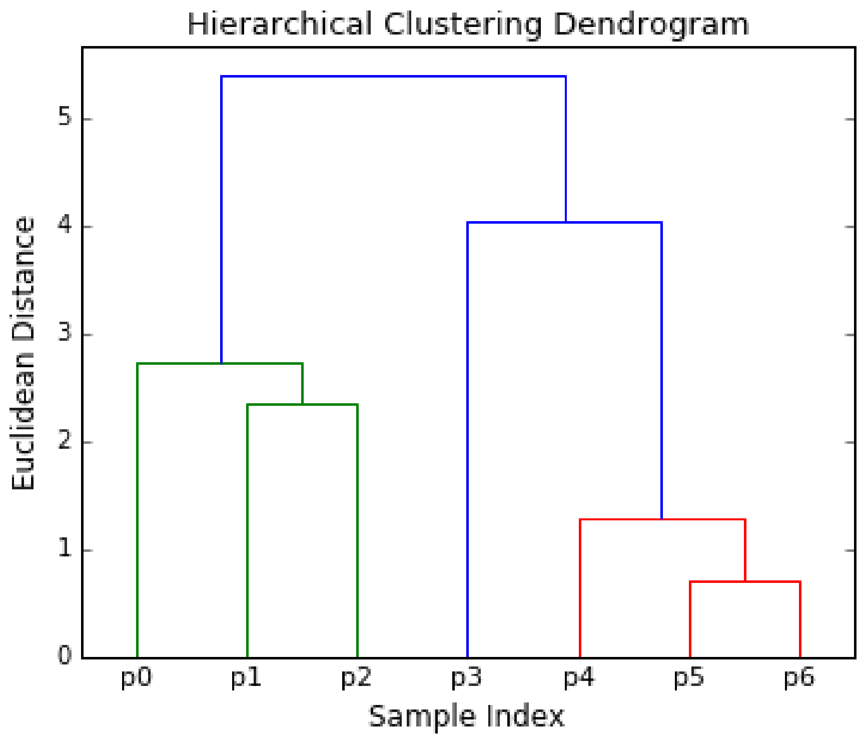
Two possible cuts of the dendogram are shown:

- at 0.4 into 24 clusters,
- at 0.1 into 12 clusters.



Dendrogram (3)



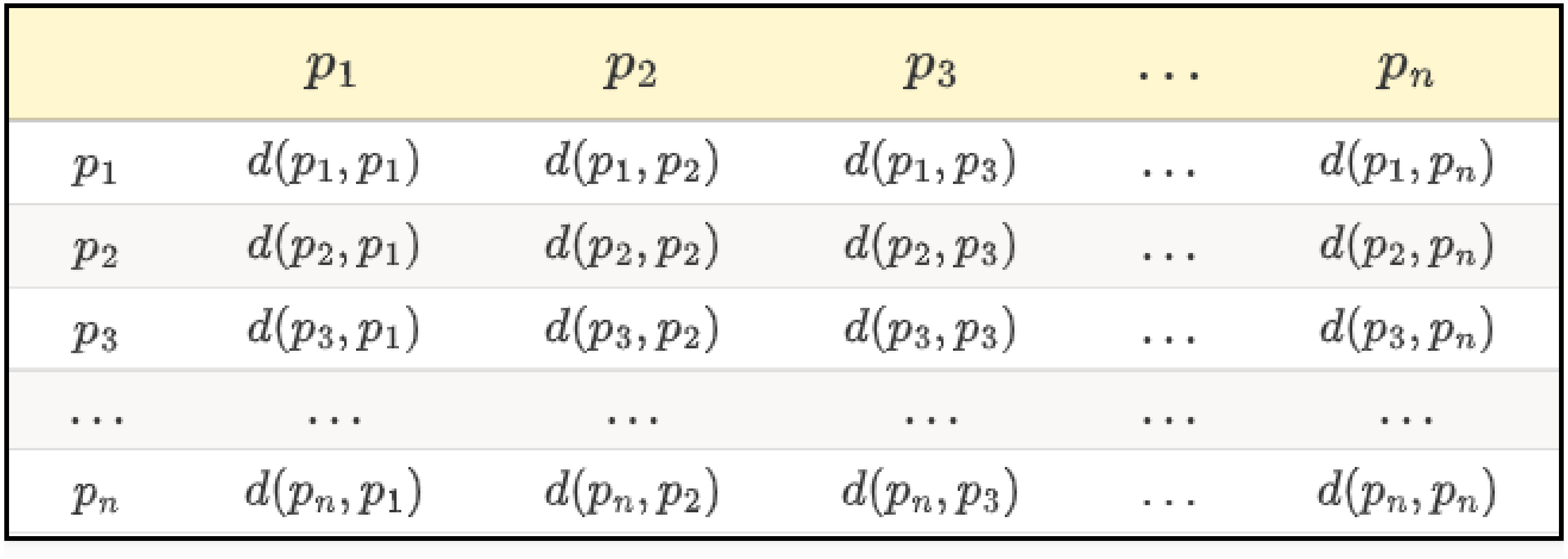


Source: https://www.kdnuggets.com/2018/06/5-clustering-algorithms-data-scientists-need-know.html

HAC (typical) steps

- Step 1:
  - Each data point is assigned to a cluster.
- Step 2:
  - Compute the proximity matrix using a particular distance (or similarity) metric.
- Step 3:
  - Merge the clusters based on a metric for the distance (or similarity) between clusters.
- Step 4:
  - Update the distance matrix.
- Step 5:
  - Repeat Step 3 and Step 4 until only a single cluster remains.

Steps 1, 2: Proximity matrix (data points)



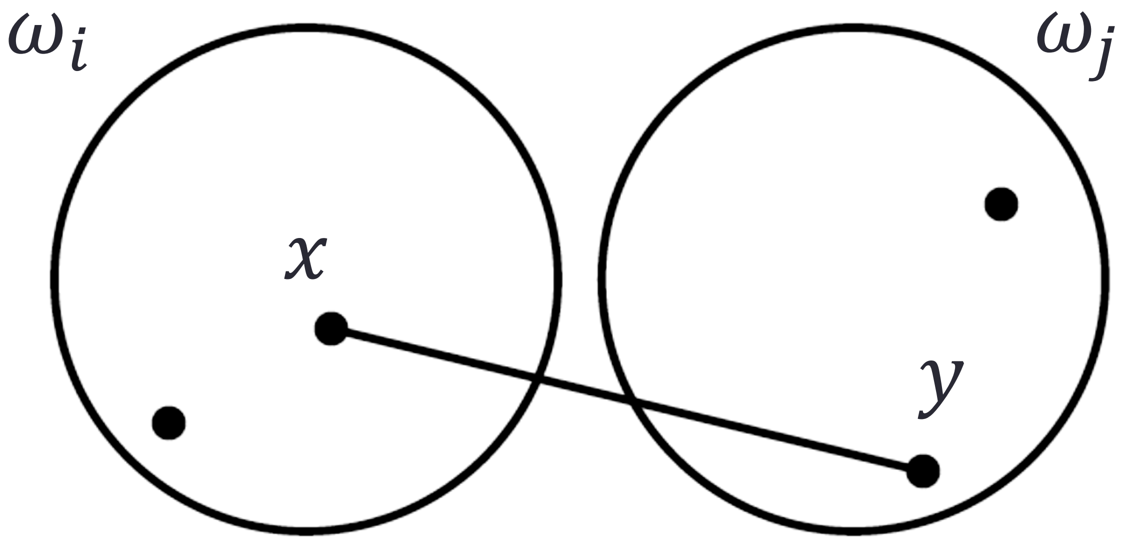
	p_1	p_2	p_3	$\dots$	p_n
p_1	$d(p_1, p_1)$	$d(p_1, p_2)$	$d(p_1, p_3)$	$\dots$	$d(p_1, p_n)$
p_2	$d(p_2, p_1)$	$d(p_2, p_2)$	$d(p_2, p_3)$	$\dots$	$d(p_2, p_n)$
p_3	$d(p_3, p_1)$	$d(p_3, p_2)$	$d(p_3, p_3)$	$\dots$	$d(p_3, p_n)$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
p_n	$d(p_n, p_1)$	$d(p_n, p_2)$	$d(p_n, p_3)$	$\dots$	$d(p_n, p_n)$

- At this stage, clustering can be, for example, performed by using k-means.

Steps 3, 4, 5: Closest pair of clusters

- The main question in hierarchical clustering is how to calculate the distance/similarity between clusters and update the proximity matrix.
  - Many variants to defining closest pair of clusters (merging criteria).
- Single-link
  - Distance of the least distant.
  - Similarity of the most cosine-similar.
- Complete-link
  - Distance of the furthest points.
  - The least cosine-similar.
- Group-average
  - Average distance/cosine similarity between pairs of elements.
- Centroid
  - Clusters whose centroids (centers of gravity) are the least distant/most similar.

Single-link: Example



(a) Single-link: Minimum distance (or maximum similarity)

Single-link HAC

- In single-link clustering, the distance (similarity) of two clusters is the distance of their least distant (similarity of their most similar) members (the merge criterion is local).

- Use minimum distance (maximum similarity) of pairs:

$$d(\omega_i, \omega_j) = \min_{x \in \omega_i, y \in \omega_j} d(x, y)$$

- After merging ω_i and ω_j, the similarity of the resulting cluster to another cluster, ω_k, is:

$$d((\omega_i \cup \omega_j), \omega_k) = \min\big(d(\omega_i, \omega_k), d(\omega_j, \omega_k)\big)$$

Single-link: Example with distance (1)

- $d(\omega_i, \omega_j) = \min_{x \in \omega_i, y \in \omega_j} d(x, y)$
- $d((\omega_i \cup \omega_j), \omega_k) = \min(d(\omega_i, \omega_k), d(\omega_j, \omega_k))$

	ω_1	ω_2	ω_3	ω_4	ω_5
ω_1	0	17	21	31	23
ω_2	17	0	30	34	21
ω_3	21	30	0	28	39
ω_4	31	34	28	0	43
ω_5	23	21	39	43	0

Single-link: Example with distance (2)

- $d(\omega_i, \omega_j) = \min_{x \in \omega_i, y \in \omega_j} d(x, y)$

	ω_1	ω_2	ω_3	ω_4	ω_5
ω_1	0	17	21	31	23
ω_2	17	0	30	34	21
ω_3	21	30	0	28	39
ω_4	31	34	28	0	43
ω_5	23	21	39	43	0

Single-link: Example with distance (3)

- $d((\omega_i \cup \omega_j), \omega_k) = \min(d(\omega_i, \omega_k), d(\omega_j, \omega_k))$

	ω_1	ω_2	ω_3	ω_4	ω_5
ω_1	0	17	21	31	23
ω_2	17	0	30	34	21
ω_3	21	30	0	28	39
ω_4	31	34	28	0	43
ω_5	23	21	39	43	0

- $d((\omega_1 \cup \omega_2), \omega_3) = \min(d(\omega_1, \omega_3), d(\omega_2, \omega_3)) = \min(21{,}30) = 21$
- $d((\omega_1 \cup \omega_2), \omega_4) = \min(d(\omega_1, \omega_4), d(\omega_2, \omega_4)) = \min(31{,}34) = 31$
- $d((\omega_1 \cup \omega_2), \omega_5) = \min(d(\omega_1, \omega_5), d(\omega_2, \omega_5)) = \min(23{,}21) = 21$

Single-link: Example with distance (4)

	$\omega_1 \cup \omega_2$	ω_3	ω_4	ω_5
$\omega_1 \cup \omega_2$	0	21	31	21
ω_3	21	0	28	39
ω_4	31	28	0	43
ω_5	21	39	43	0

- Since $d\big((\omega_1 \cup \omega_2), \omega_3\big) = d\big((\omega_1 \cup \omega_2), \omega_5\big) = 21$, we can join cluster $(\omega_1 \cup \omega_2)$ with ω_3 and ω_5

- Hence, this means that later, we have to compute
$$d\big((\omega_1 \cup \omega_2) \cup \omega_3, \omega_5), \omega_4\big)$$

Single-link: Example with distance (5)

- $d\big((\omega_1 \cup \omega_2) \cup \omega_3, \omega_5), \omega_4\big) =$

$$= \min\Big(d\big((\omega_1 \cup \omega_2), \omega_4\big), d(\omega_3, \omega_4), d(\omega_5, \omega_4)\Big) = 28$$

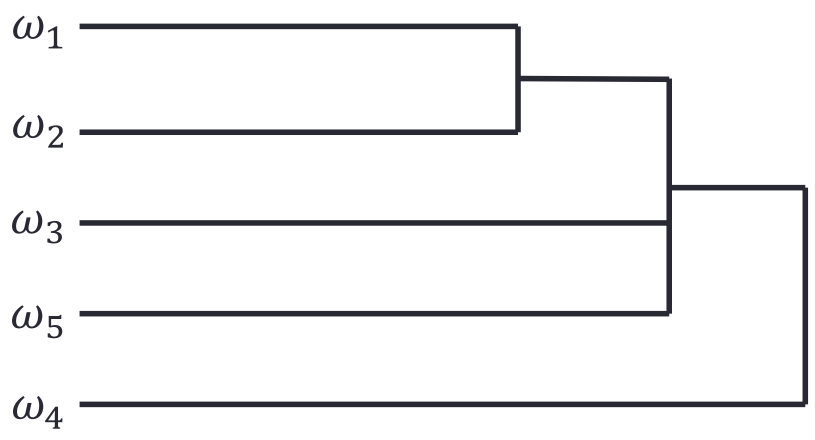
	$\omega_1 \cup \omega_2$	ω_3	ω_4	ω_5
$\omega_1 \cup \omega_2$	0	21	31	21
ω_3	21	0	28	39
ω_4	31	28	0	43
ω_5	21	39	43	0

- $d\big((\omega_1 \cup \omega_2), \omega_4\big) = 31$
- $d(\omega_3, \omega_4) = 28$
- $d(\omega_5, \omega_4) = 43$

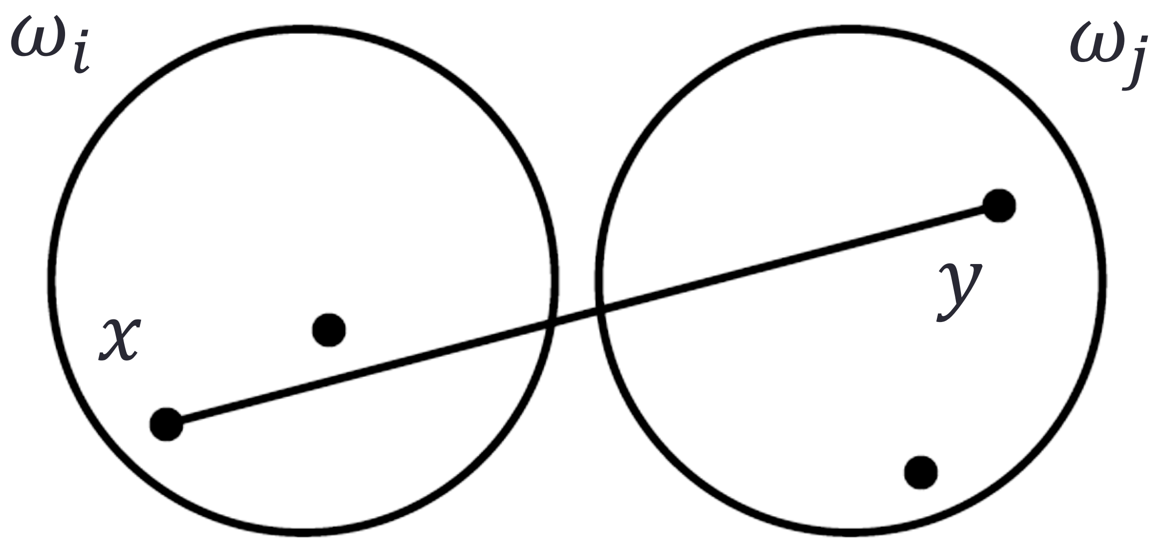
$\min(31, 28, 43) = 28$

Single-link: Example with distance (6)

	$(\omega_1 \cup \omega_2) \cup \omega_3, \omega_5$	ω_4
$(\omega_1 \cup \omega_2) \cup \omega_3, \omega_5$	0	28
ω_4	28	0



Complete-link: Example



(b) Complete-link: Maximum distance (or minimum similarity)

Complete-link HAC

- In complete-link clustering or complete-linkage clustering, the distance (similarity) of two clusters is the distance of their furthest (similarity of their most dissimilar) members (the merge criterion is non-local).

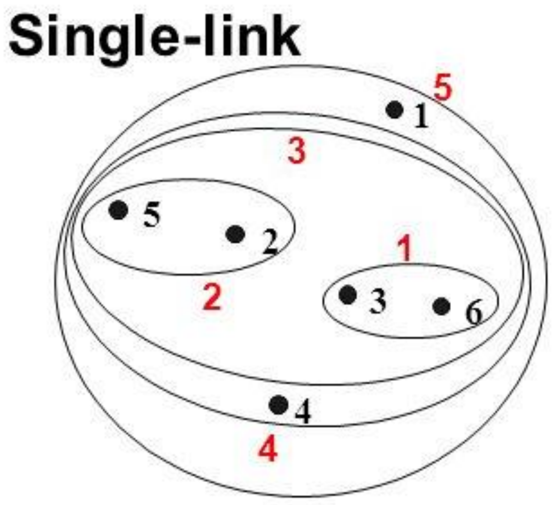
- Use maximum distance (minimum similarity) of pairs:

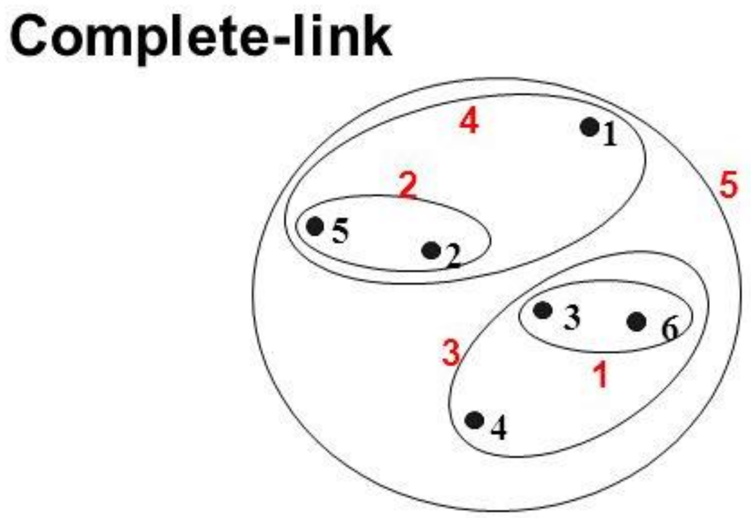
$$d(\omega_i, \omega_j) = \max_{x \in \omega_i, y \in \omega_j} d(x, y)$$

- After merging ω_i and ω_j, the similarity of the resulting cluster to another cluster, ω_k, is:

$$d\left((\omega_i \cup \omega_j), \omega_k\right) = \max\left(d(\omega_i, \omega_k), d(\omega_j, \omega_k)\right)$$

HAC: Comparison





- **SL: sensitive to noise and outliers** → One stray point can link two otherwise separate clusters.
- **CL: slightly slower** than SL.

Hierarchical Divisive Clustering (HDC) (1)

- We start at the top with all documents in one cluster.
- The cluster is split using a flat clustering algorithm.
- This procedure is applied recursively until each document is in its own singleton cluster.

Hierarchical Divisive Clustering (HDC) (2)

- Divisive hierarchical clustering with k-means is one of the efficient clustering methods among all the clustering methods.

- In this method, a cluster is split into k-smaller clusters under continuous iteration using k-means clustering until every element has its own cluster.

- It has the advantage of being more efficient than HAC if we do not generate a complete hierarchy all the way down to individual document leaves.

EVALUATION

What is a good clustering?

- When evaluating clustering results, we can use both internal evaluation and external evaluation criteria.

- Internal evaluation
  - Measures the quality of clustering based on the data and the clustering results, without using any external information or ground truth.
    - The evaluation is performed using metrics that assess the structure and characteristics of the clusters formed by the algorithm.

- External evaluation
  - Involves comparing the clustering results to some external, independent criterion or ground truth.
    - In this case, we have access to information about the true cluster assignments of the data.

Internal evaluation criteria

- A good clustering will produce high quality clusters in which:
  - The intra-class (that is, intra-cluster) similarity is high.
  - The inter-class similarity is low.

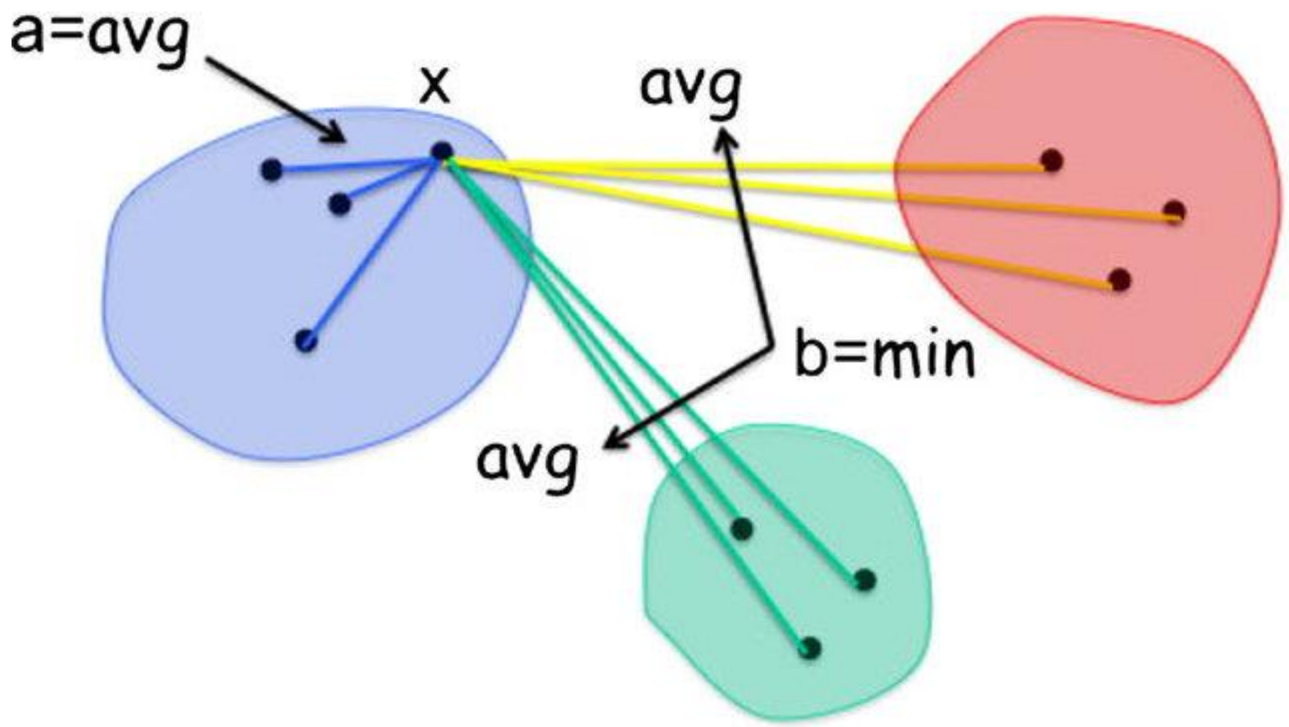
- The measured quality of a clustering depends on both the document representation and the similarity measure used.
  - The representation does not capture relevant information in the text.
    - The algorithm may struggle to find meaningful patterns, and clusters may not reflect the actual structure of the data.
  - The similarity measure does not align with the nature of the data.
    - The algorithm may group documents incorrectly, leading to suboptimal clustering results.

Internal evaluation: Silhouette (1)

- The Silhouette analysis measures how well an observation is clustered and it estimates the average distance between clusters.
- The Silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters.
- The Silhouette Coefficient (S) is calculated using the mean intra-cluster distance $a(i)$ and the mean nearest-cluster distance $b(i)$ for each sample i.

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Internal evaluation: Silhouette (2)



Internal evaluation: Silhouette (3)

- $S(i)$ will lies between $[-1,1]$.
- If the Silhouette value is close to 1, sample is well-clustered and already assigned to a very appropriate cluster.
- If the Silhouette value is about to 0, sample could be assigned to another cluster closest to it and the sample lies equally far away from both the clusters. That means it indicates overlapping clusters.
- If the Silhouette value is close to –1, sample is misclassified and is merely placed somewhere in between the clusters.

External evaluation criteria

- Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data.
- Assesses a clustering with respect to ground truth… requires labeled data.
- Assume documents with C gold standard classes, while our clustering algorithms produce k clusters, $\omega_1, \omega_2, \ldots, \omega_k$ with n_i members.

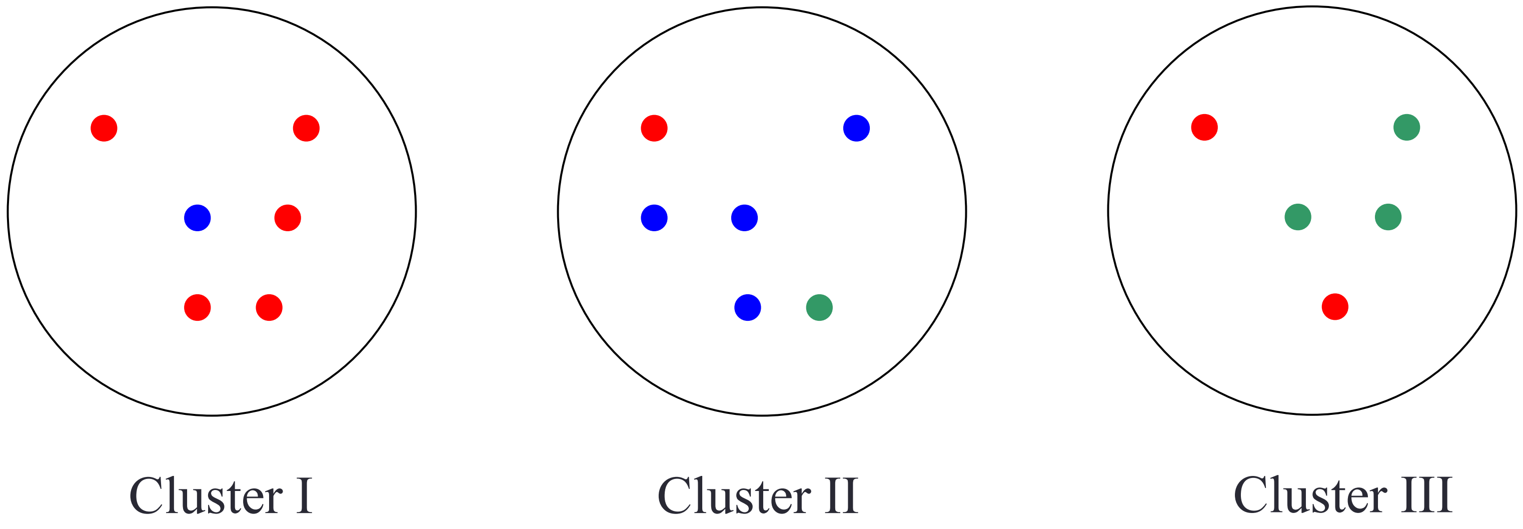
External evaluation: Purity (1)

- Simple measure: purity, the ratio between the dominant class in the cluster ω_i and the size of cluster ω_i.

$$Purity(\omega_i) = \frac{1}{n_i} \max_j(n_{ij}) \quad j \in C$$

- Bad clustering have purity values close to **0**, a perfect clustering has a purity of **1**.

External evaluation: Purity (2)



- Cluster I: $Purity(I) = 1/6(\max(5,1,0)) = 5/6$
- Cluster II: $Purity(II) = 1/6(\max(1,4,1)) = 4/6$
- Cluster III: $Purity(III) = 1/5(\max(2,0,3)) = 3/5$

External evaluation: The Rand Index (1)

- It measures the percentage of decisions that are correct.
  - A true positive (TP) decision assigns two similar documents to the same cluster.
  - A true negative (TN) decision assigns two dissimilar documents to different clusters.
  - A false positive (FP) decision assigns two dissimilar documents to the same cluster.
  - A false negative (FN) decision assigns two similar documents to different clusters.

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

External evaluation: The Rand Index (2)

Number of points	Same Cluster in clustering	Different Clusters in clustering
Same class in ground truth	20	24
Different classes in ground truth	20	72

$$RI = \frac{20 + 72}{20 + 20 + 24 + 72} \approx 0.68$$

Precision, Recall and F-measure

- The Rand Index gives equal weight to false positives and false negatives.
- Separating similar documents is sometimes worse than putting pairs of dissimilar documents in the same cluster.
- We can use the F-measure to penalize false negatives more strongly than false positives by selecting a value $\beta > 1$, thus giving more weight to recall.

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \qquad F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$$

Example

- $P = \frac{TP}{TP+FP} = \frac{20}{20+20} = 0.5$

- $R = \frac{TP}{TP+FN} = \frac{20}{20+24} \approx 0.455$

- $RI = \frac{TP+TN}{TP+FP+FN+TN} \approx 0.68$

- $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$ $\quad F_1 \approx 0.48 \quad F_5 \approx 0.456$

Clustering in R and Python

- Introduction to text clustering in R:
  - https://recast.ai/blog/text-clustering-with-r-an-introduction-for-data-scientists/

- Introduction to text clustering in Python:
  - http://brandonrose.org/clustering
  - https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html