NPFL103: Information Retrieval (10)

Document clustering

Pavel Pecina

pecina@ufal.mff.cuni.cz

Lecturer

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University

Based on slides by Hinrich Schütze, University of Stuttgart.

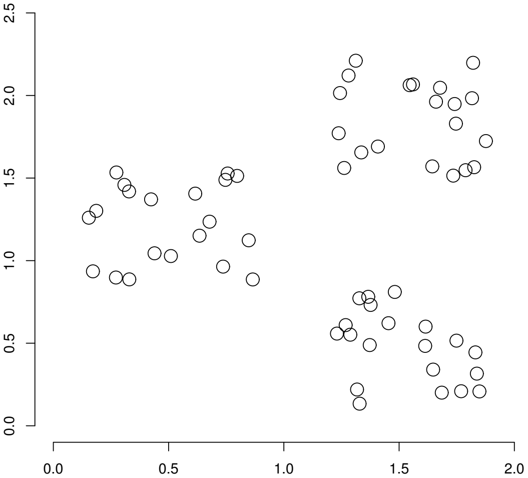
Contents

Introduction

Clustering: Definition

- (Document) clustering is the process of grouping a set of documents into clusters of similar documents.
- Documents within a cluster should be similar.
- Documents from different clusters should be dissimilar.
- Clustering is the most common form of unsupervised learning.
- Unsupervised = there are no labeled or annotated data.

Exercise: Data set with clear cluster structure



Classification vs. Clustering

- Classification: supervised learning
- Clustering: unsupervised learning
- Classification: Classes are human-defined and part of the input to the learning algorithm.
- Clustering: Clusters are inferred from the data without human input.
  - However, there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, ...

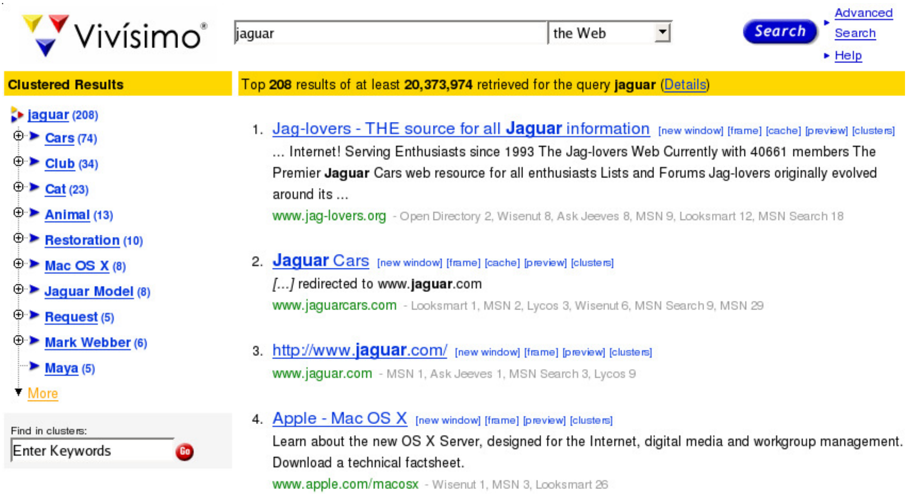
The cluster hypothesis

- Cluster hypothesis: Documents in the same cluster behave similarly with respect to relevance to information needs.
- All applications of clustering in IR are based (directly or indirectly) on the cluster hypothesis.
- Van Rijsbergen's original wording (1979): "closely associated documents tend to be relevant to the same requests".

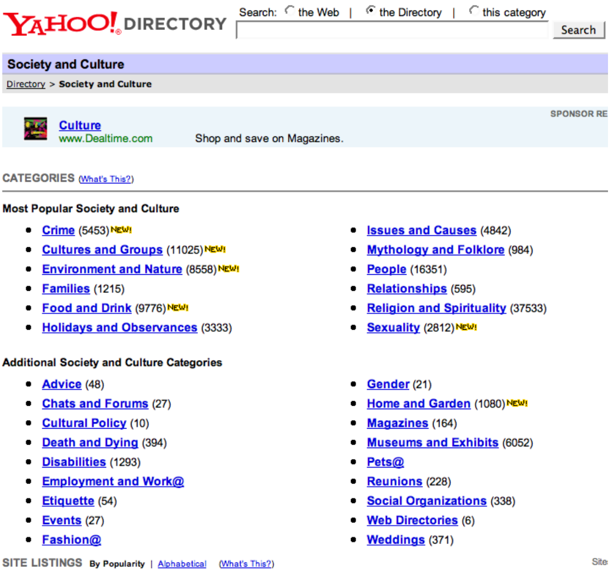
Applications of clustering in IR

application	what is clustered?	benefit
search result clustering	search results	more effective information presentation to user
collection clustering	collection	effective information presentation for exploratory browsing
cluster-based retrieval	collection	higher efficiency: faster search

Search result clustering for better navigation



Global navigation: Yahoo



Global navigation: MESH

MeSH Tree Structures - 2008

Return to Entry Page

1. ⊞ Anatomy [A]
2. ⊞ Organisms [B]
3. ⊟ Diseases [C]
   - Bacterial Infections and Mycoses [C01] +
   - Virus Diseases [C02] +
   - Parasitic Diseases [C03] +
   - Neoplasms [C04] +
   - Musculoskeletal Diseases [C05] +
   - Digestive System Diseases [C06] +
   - Stomatognathic Diseases [C07] +
   - Respiratory Tract Diseases [C08] +
   - Otorhinolaryngologic Diseases [C09] +
   - Nervous System Diseases [C10] +
   - Eye Diseases [C11] +
   - Male Urogenital Diseases [C12] +
   - Female Urogenital Diseases and Pregnancy Complications [C13] +
   - Cardiovascular Diseases [C14] +
   - Hemic and Lymphatic Diseases [C15] +
   - Congenital, Hereditary, and Neonatal Diseases and Abnormalities [C16] +
   - Skin and Connective Tissue Diseases [C17] +
   - Nutritional and Metabolic Diseases [C18] +
   - Endocrine System Diseases [C19] +
   - Immune System Diseases [C20] +
   - Disorders of Environmental Origin [C21] +
   - Animal Diseases [C22] +
   - Pathological Conditions, Signs and Symptoms [C23] +
4. ⊞ Chemicals and Drugs [D]
5. ⊞ Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
6. ⊞ Psychiatry and Psychology [F]
7. ⊞ Biological Sciences [G]
8. ⊞ Natural Sciences [H]
9. ⊞ Anthropology, Education, Sociology and Social Phenomena [I]
10. ⊞ Technology, Industry, Agriculture [J]
11. ⊞ Humanities [K]

Global navigation: MESH (lower level)

- Neoplasms [C04]
  - Cysts [C04.182] +
  - Hamartoma [C04.445] +
  - ➤ Neoplasms by Histologic Type [C04.557]
    - Histiocytic Disorders, Malignant [C04.557.227] +
    - Leukemia [C04.557.337] +
    - Lymphatic Vessel Tumors [C04.557.375] +
    - Lymphoma [C04.557.386] +
    - Neoplasms, Complex and Mixed [C04.557.435] +
    - Neoplasms, Connective and Soft Tissue [C04.557.450] +
    - Neoplasms, Germ Cell and Embryonal [C04.557.465] +
    - Neoplasms, Glandular and Epithelial [C04.557.470] +
    - Neoplasms, Gonadal Tissue [C04.557.475] +
    - Neoplasms, Nerve Tissue [C04.557.580] +
    - Neoplasms, Plasma Cell [C04.557.595] +
    - Neoplasms, Vascular Tissue [C04.557.645] +
    - Nevi and Melanomas [C04.557.665] +
    - Odontogenic Tumors [C04.557.695] +
  - Neoplasms by Site [C04.588] +
  - Neoplasms, Experimental [C04.619] +
  - Neoplasms, Hormone-Dependent [C04.626]
  - Neoplasms, Multiple Primary [C04.651] +
  - Neoplasms, Post-Traumatic [C04.666]
  - Neoplasms, Radiation-Induced [C04.682] +
  - Neoplasms, Second Primary [C04.692]
  - Neoplastic Processes [C04.697] +
  - Neoplastic Syndromes, Hereditary [C04.700] +
  - Paraneoplastic Syndromes [C04.730] +
  - Precancerous Conditions [C04.834] +
  - Pregnancy Complications, Neoplastic [C04.850] +
  - Tumor Virus Infections [C04.925] +

Navigational hierarchies: Manual vs. automatic creation

- Note: Yahoo/MESH are not examples of clustering ...

  but well known examples for using a global hierarchy for navigation.
- Eample for global navigation/exploration based on clustering:
  - Google News

Clustering for improving recall

- To improve search recall:
  - Cluster docs in collection a priori
  - When a query matches a doc *d*, also return other docs in the cluster containing *d*
- Hope: if we do this: the query "car" will also return docs containing "automobile"
  - Because the clustering algorithm groups together docs containing "car" with those containing "automobile".
  - Both types of documents contain words like "parts", "dealer", "mercedes", "road trip".

Goals of clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - We'll see different ways of formalizing this.
- The number of clusters should be appropriate for the data set we are clustering.
  - Initially, we will assume the number of clusters K is given.
  - Later: Semiautomatic methods for determining K
- Secondary goals in clustering
  - Avoid very small and very large clusters
  - Define clusters that are easy to explain to the user
  - ...

Flat vs. hierarchical clustering

- Flat algorithms
  - Usually start with a random (partial) partitioning of docs into groups
  - Refine iteratively
  - Main algorithm: K-means
- Hierarchical algorithms
  - Create a hierarchy
  - 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.
  - Makes more sense for applications like creating browsable hierarchies
  - You may want to put *sneakers* in two clusters: *sports apparel*/*shoes*
  - You can only do that with a soft clustering approach.
- This class: flat and hierarchical hard clustering
- Next class: latent semantic indexing, a form of soft clustering

Flat algorithms

- Flat algorithms compute a partition of N documents into K clusters.
- Given: a set of documents and the number K
- Find: a partition into K clusters optimizing the chosen criterion
- Global optimization: exhaustively enumerate partitions, pick optimal
  - Not tractable
- Effective heuristic method: K-means algorithm

K-means

K-means

- Perhaps the best known clustering algorithm
- Simple, works well in many cases
- Use as default / baseline for clustering documents

Document representations in clustering

- Vector space model
- As in vector space classification, we measure relatedness between vectors by Euclidean distance ...

  ...which is almost equivalent to cosine similarity.
- Almost: centroids are not length-normalized.

K-means: Basic idea

- Each cluster in *K*-means is defined by a centroid.
- Objective/partitioning criterion: minimize the average squared difference from the centroid
- Recall definition of centroid (ω denotes a cluster):

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

- We search for minimum avg. squared difference by iterating 2 steps:
  - reassignment: assign each vector to its closest centroid
  - recomputation: recompute each centroid as the average of the vectors that were assigned to it in reassignment

K-means pseudocode (μ_k is centroid of ω_k)

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 $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$

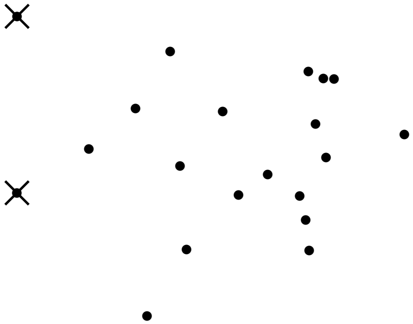
    $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*

  for $k \leftarrow 1$ **to** K

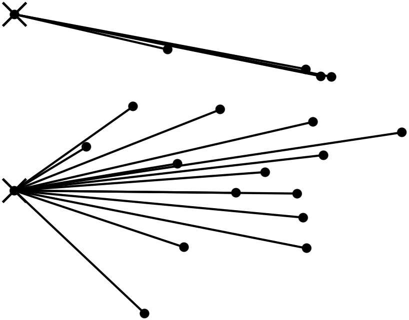
  do $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*

return $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

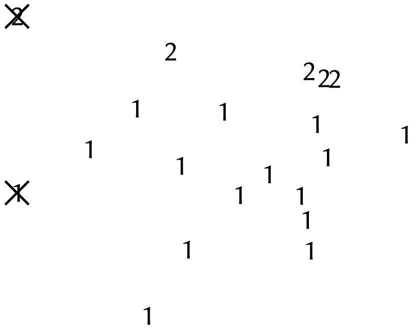
Worked Example: Random selection of initial centroids



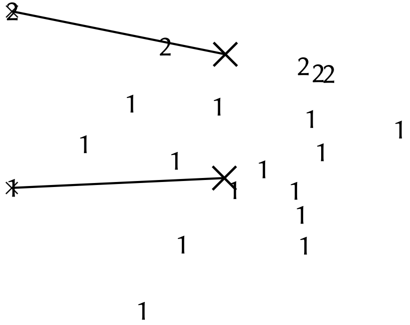
Worked Example: Assign points to closest center



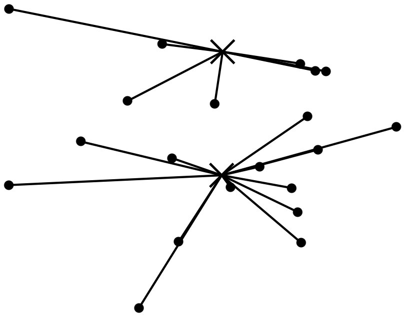
Worked Example: Assignment



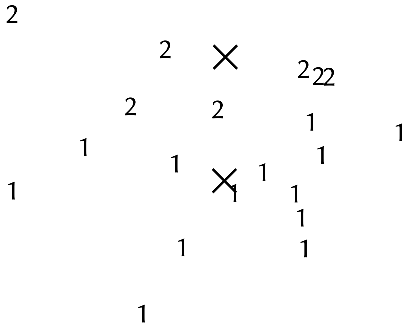
Worked Example: Recompute cluster centroids



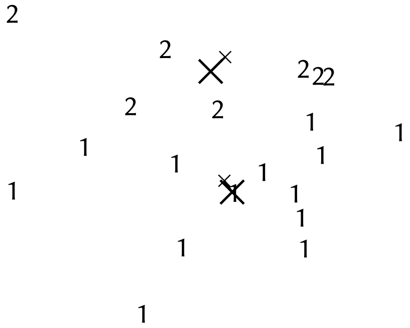
Worked Example: Assign points to closest centroid



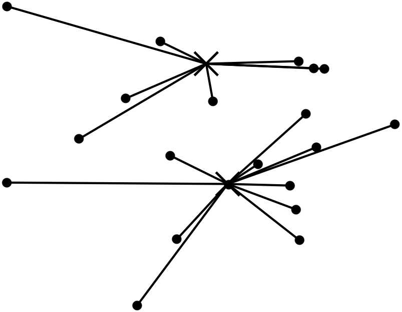
Worked Example: Assignment



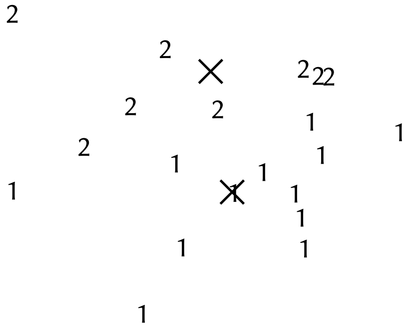
Worked Example: Recompute cluster centroids



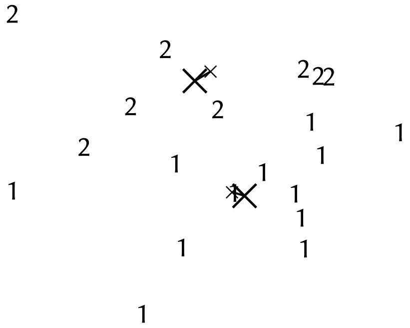
Worked Example: Assign points to closest centroid



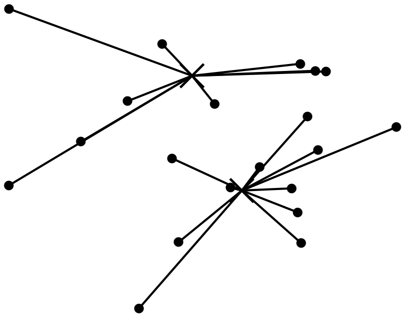
Worked Example: Assignment



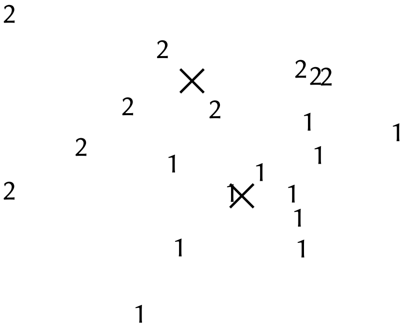
Worked Example: Recompute cluster centroids



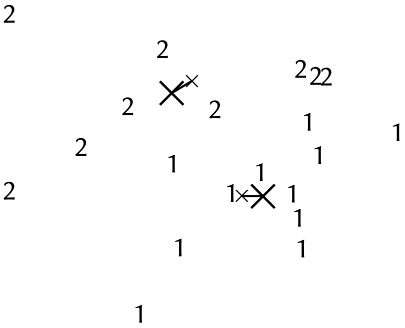
Worked Example: Assign points to closest centroid



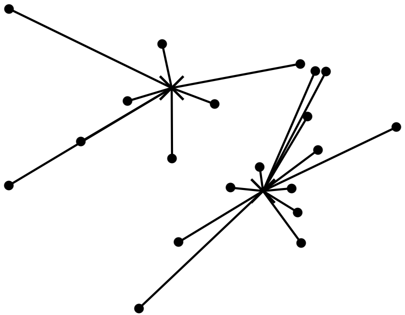
Worked Example: Assignment



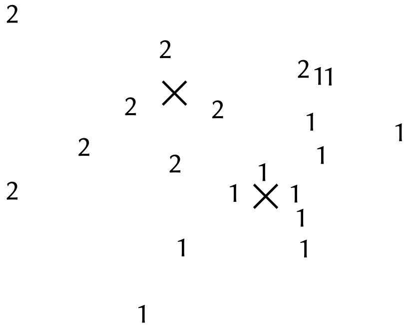
Worked Example: Recompute cluster centroids



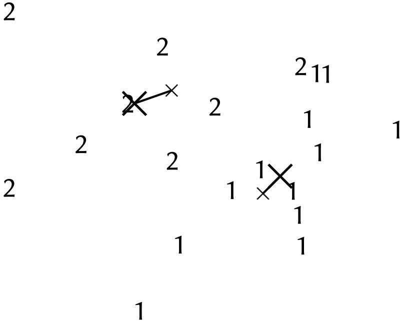
Worked Example: Assign points to closest centroid



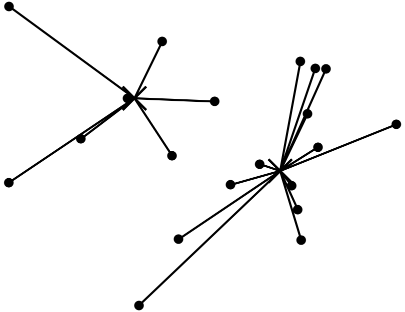
Worked Example: Assignment



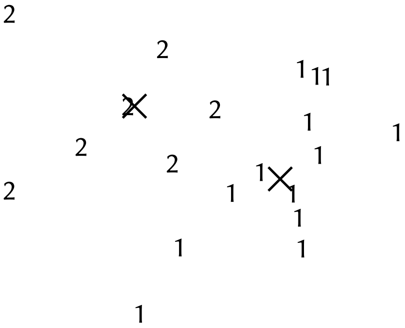
Worked Example: Recompute cluster centroids



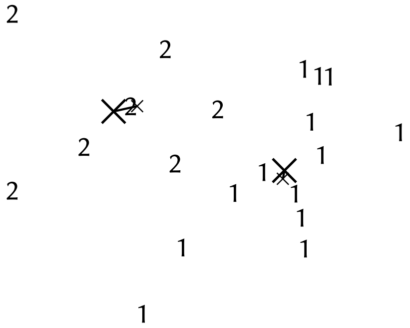
Worked Example: Assign points to closest centroid



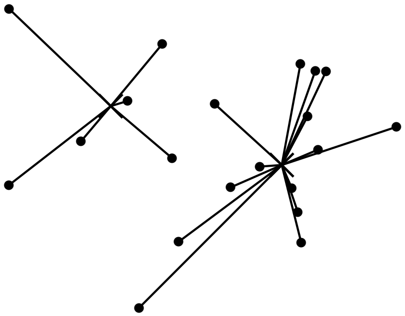
Worked Example: Assignment



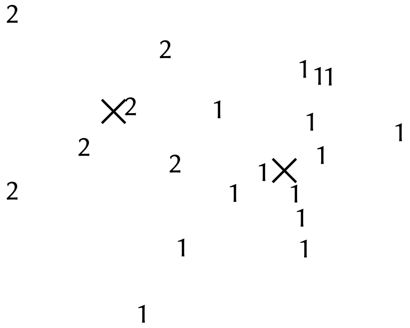
Worked Example: Recompute cluster centroids



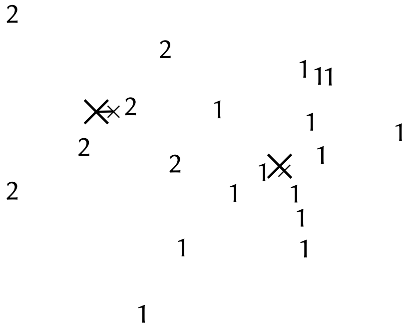
Worked Example: Assign points to closest centroid



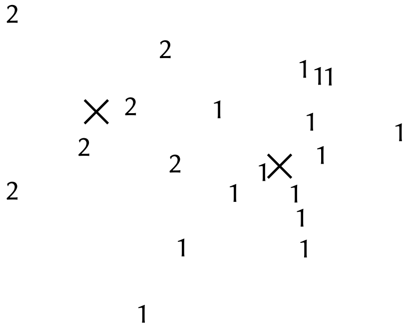
Worked Example: Assignment



Worked Example: Recompute cluster centroids



Worked Example: Centroids and assignments after convergence



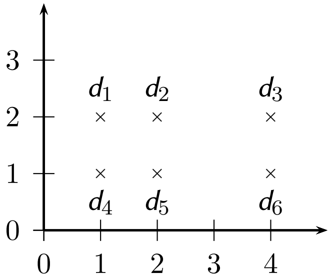
K-means is guaranteed to converge: Proof

- RSS = sum of all squared distances between document vector and closest centroid
- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - See the book for a proof.
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Assumption: Ties are broken consistently.
- Finite set & monotonically decreasing $\rightarrow$ convergence

convergence and pptimality of *K*-means

- *K*-means is guaranteed to converge
- But we don't know how long convergence will take!
- If we don't care about a few docs switching back and forth, then convergence is usually fast ($<$ 10-20 iterations).
- However, complete convergence can take many more iterations.
- Convergence $\neq$ optimality
- Convergence does not mean that we converge to the optimal clustering!
- This is the great weakness of *K*-means.
- If we start with a bad set of seeds, the resulting clustering can be horrible.

Exercise: Suboptimal clustering



- What is the optimal clustering for $K = 2$?
- Do we converge on this clustering for arbitrary seeds d_i, d_j?

Initialization of *K*-means

- Random seed selection is just one of many ways *K*-means can be initialized.
- Random seed selection is not very robust: It's easy to get a suboptimal clustering.
- Better ways of computing initial centroids:
  - Select seeds not randomly, but using some heuristic (e.g., filter out outliers or find a set of seeds that has "good coverage" of the document space)
  - Use hierarchical clustering to find good seeds
  - Select i (e.g., $i = 10$) different random sets of seeds, do a *K*-means clustering for each, select the clustering with lowest RSS

Time complexity of K-means

- Computing one distance of two vectors is $O(M)$.
- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances)
- Recomputation step: $O(NM)$ (we need to add each of the document's $< M$ values to one of the centroids)
- Assume number of iterations bounded by I
- Overall complexity: $O(IKNM)$ – linear in all important dimensions
- However: This is not a real worst-case analysis.
- In pathological cases, complexity can be worse than linear.

Evaluation

What is a good clustering?

- Internal criteria
  - Example of an internal criterion: RSS in K-means
- But an internal criterion often does not evaluate the actual utility of a clustering in the application.
- Alternative: External criteria
  - Evaluate with respect to a human-defined classification

External criteria for clustering quality

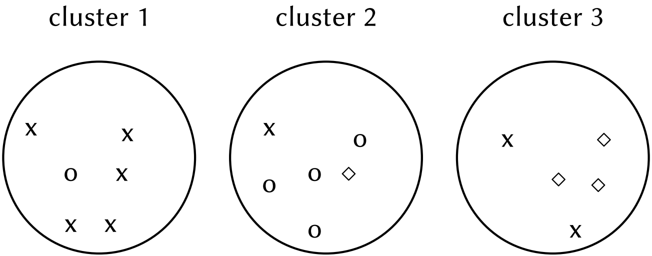
- Based on a gold standard data set, e.g., the Reuters collection we also used for the evaluation of classification
- Goal: Clustering should reproduce the classes in the gold standard
- (But we only want to reproduce how documents are divided into groups, not the class labels.)
- First measure for how well we were able to reproduce the classes: purity

External criterion: Purity

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.
- For each cluster ω_k: find class c_j with most members n_{kj} in ω_k
- Sum all n_{kj} and divide by total number of points

Example for computing purity



To compute purity:

$5 = \max_j |\omega_1 \cap c_j|$ (class x, cluster 1);

$4 = \max_j |\omega_2 \cap c_j|$ (class o, cluster 2); and

$3 = \max_j |\omega_3 \cap c_j|$ (class $\diamond$, cluster 3).

Purity is $(1/17) \times (5 + 4 + 3) \approx 0.71$.

Another external criterion: Rand index

- Purity can be increased easily by increasing K – a measure that does not have this problem: Rand index.

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

- Based on 2x2 contingency table of all pairs of documents:

	same cluster	different clusters
same class	true positives (TP)	false negatives (FN)
different classes	false positives (FP)	true negatives (TN)

- Where:
  - TP+FN+FP+TN is the total number of pairs; $\binom{N}{2}$ for N docs.
  - Each pair is either positive or negative (the clustering puts the two documents in the same or in different clusters) ...
  - ...and either "true" (correct) or "false" (incorrect): the clustering decision is correct or incorrect.

Example: compute Rand Index for the o/◇/x example

- We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

$$\text{TP} + \text{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

- Of these, the x pairs in cluster 1, the o pairs in cluster 2, the ◇ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$\text{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

- Thus, $\text{FP} = 40 - 20 = 20$.
- FN and TN are computed similarly.

Rand index for the o/$\diamond$/x example

	same cluster	different clusters
same class	TP $= 20$	FN $= 24$
different classes	FP $= 20$	TN $= 72$

RI is then $(20 + 72)/(20 + 20 + 24 + 72) \approx 0.68$.

How many clusters?

How many clusters?

- Number of clusters K is given in many applications.
  - E.g., there may be an external constraint on K.
- What if there is no external constraint? Is there a "right" number of clusters?
- One way to go: define an optimization criterion
  - Given docs, find K for which the optimum is reached.
  - What optimization criterion can we use?
  - We can't use RSS or average squared distance from centroid as criterion: always chooses $K = N$ clusters.

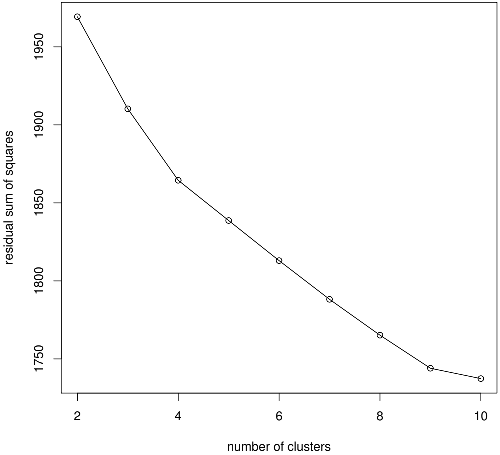
Simple objective function for K: Basic idea

- Start with 1 cluster ($K = 1$)
- Keep adding clusters (= keep increasing K)
- Add a penalty for each new cluster
- Then trade off cluster penalties against average squared distance from centroid
- Choose the value of K with the best tradeoff

Simple objective function for K: Formalization

- Given a clustering, define the cost for a document as (squared) distance to centroid
- Define total distortion RSS(K) as sum of all individual document costs (corresponds to average distance)
- Then: penalize each cluster with a cost λ
- Thus for a clustering with K clusters, total cluster penalty is $K\lambda$
- Define the total cost of a clustering as distortion plus total cluster penalty: RSS(K) + $K\lambda$
- Select K that minimizes (RSS(K) + $K\lambda$)
- Still need to determine good value for λ ...

Finding the “knee” in the curve

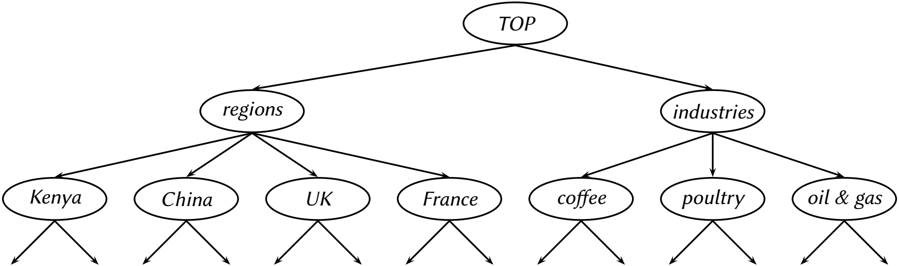


Pick the number of clusters where curve “flattens”. Here: 4 or 9.

Hierarchical clustering

Hierarchical clustering

Our goal in hierarchical clustering is to create a hierarchy like the one we saw earlier in Reuters:



- We want to create this hierarchy automatically.
- We can do this either top-down or bottom-up.
- The best known bottom-up method is hierarchical agglomerative clustering.

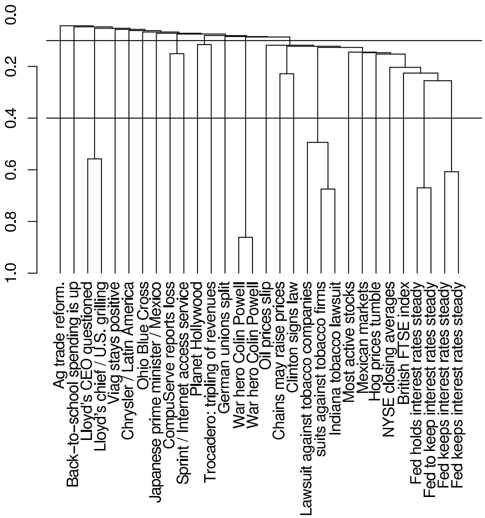
Hierarchical agglomerative clustering (HAC)

- HAC creates a hierachy in the form of a binary tree.
- Assumes a similarity measure for determining similarity of two clusters.
- Up to now, our similarity measures were for documents.
- We will look at four different cluster similarity measures.

HAC: Basic algorithm

- Start with each document in a separate cluster
- Then repeatedly merge the two clusters that are most similar
- Until there is only one cluster.
- The history of merging is a hierarchy in the form of a binary tree.
- The standard way of depicting this history is a dendrogram.

A dendrogram



- The history of mergers can be read off from bottom to top.
- The horizontal line of each merger tells us what the similarity of the merger was.
- We can cut the dendrogram at a particular point (e.g., at 0.1 or 0.4) to get a flat clustering.

Divisive clustering

- Divisive clustering is top-down.
- Alternative to HAC (which is bottom up).
- Divisive clustering:
  - Start with all docs in one big cluster
  - Then recursively split clusters
  - Eventually each node forms a cluster on its own.
- $\rightarrow$ Bisecting *K*-means at the end
- For now: HAC (= bottom-up)

Naive HAC algorithm

$\textsc{SimpleHAC}(d_1, \ldots, d_N)$

$\quad$ **for** $n \leftarrow 1$ **to** N
$\quad$ **do for** $i \leftarrow 1$ **to** N
$\quad\quad$ **do** $C[n][i] \leftarrow \textsc{Sim}(d_n, d_i)$
$\quad\quad$ $I[n] \leftarrow 1$ *(keeps track of active clusters)*
$\quad$ $A \leftarrow []$ *(collects clustering as a sequence of merges)*
$\quad$ **for** $k \leftarrow 1$ **to** $N - 1$
$\quad$ **do** $\langle i, m\rangle \leftarrow \arg\max_{\{\langle i,m\rangle : i \neq m \wedge I[i]=1 \wedge I[m]=1\}} C[i][m]$
$\quad\quad$ $A.\textsc{Append}(\langle i, m\rangle)$ *(store merge)*
$\quad\quad$ **for** $j \leftarrow 1$ **to** N
$\quad\quad$ **do** *(use i as representative for* $< i, m >$*)*
$\quad\quad\quad$ $C[i][j] \leftarrow \textsc{Sim}(< i, m >, j)$
$\quad\quad\quad$ $C[j][i] \leftarrow \textsc{Sim}(< i, m >, j)$
$\quad\quad$ $I[m] \leftarrow 0$ *(deactivate cluster)*
$\quad$ **return** A

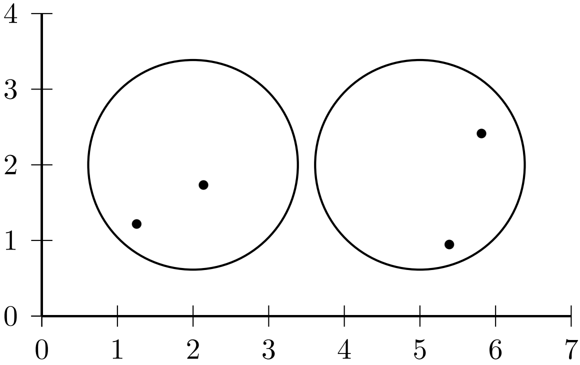
Computational complexity of the naive algorithm

- First, we compute the similarity of all $N \times N$ pairs of documents.
- Then, in each of N iterations:
  - We scan the $O(N \times N)$ similarities to find the maximum similarity.
  - We merge the two clusters with maximum similarity.
  - We compute the similarity of the new cluster with all other (surviving) clusters.
- There are $O(N)$ iterations, each performing a $O(N \times N)$ "scan" operation.
- Overall complexity is $O(N^3)$.
- We'll look at more efficient algorithms later.

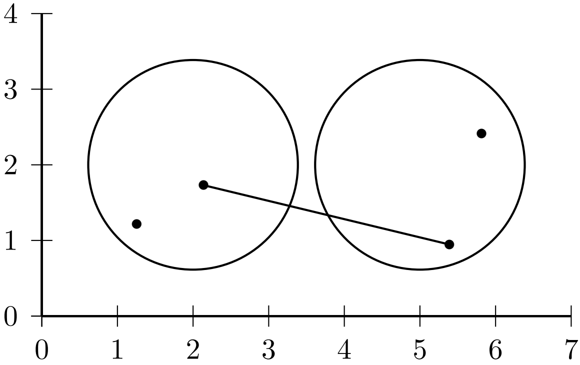
Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum similarity of any two documents
- Complete-link: Minimum similarity
  - Minimum similarity of any two documents
- Centroid: Average “intersimilarity”
  - Average similarity of all document pairs (but excluding pairs of docs in the same cluster)
  - This is equivalent to the similarity of the centroids.
- Group-average: Average “intrasimilarity”
  - Average similary of all document pairs, including pairs of docs in the same cluster

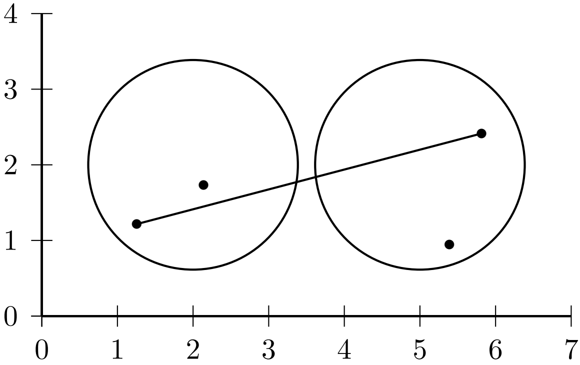
Cluster similarity: Example



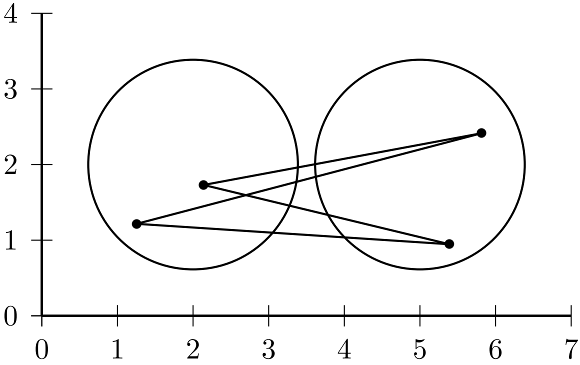
Single-link: Maximum similarity



Complete-link: Minimum similarity

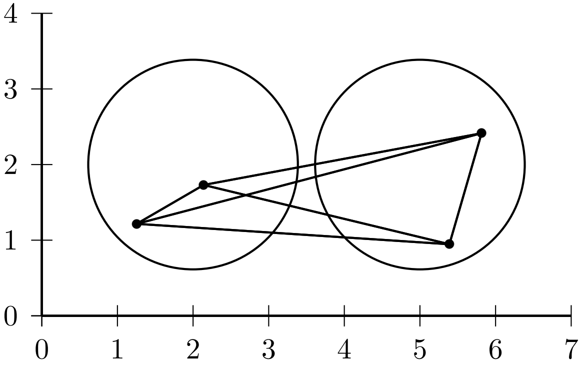


Centroid: Average intersimilarity



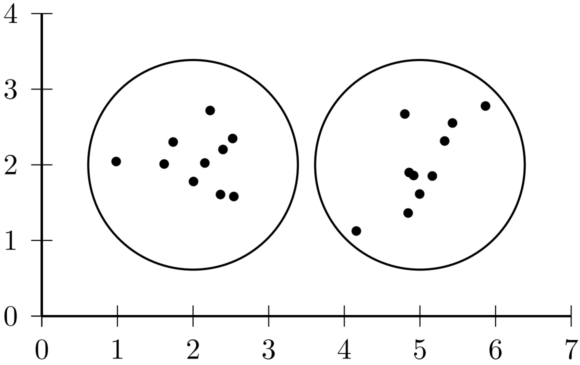
intersimilarity = similarity of two documents in different clusters

Group average: Average intrasimilarity

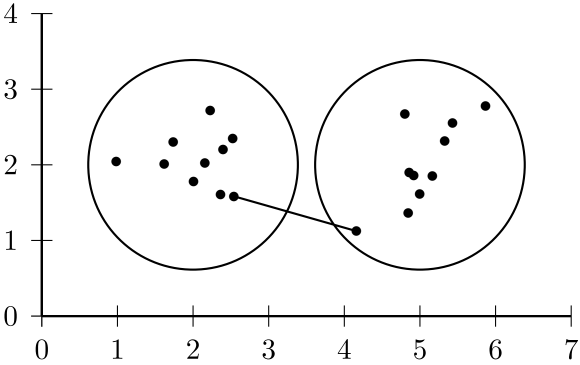


intrasimilarity = similarity of any pair, including cases in the same cluster

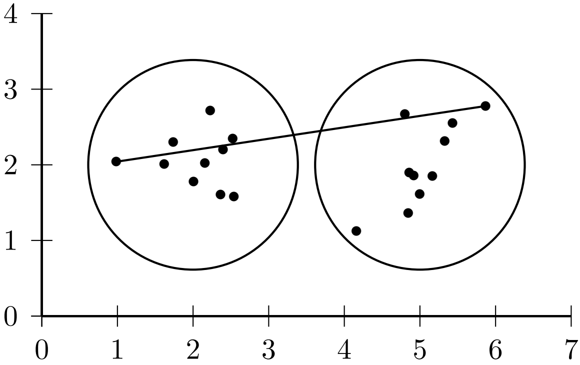
Cluster similarity: Larger Example



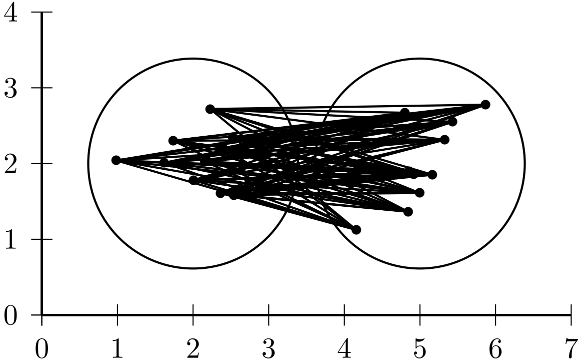
Single-link: Maximum similarity



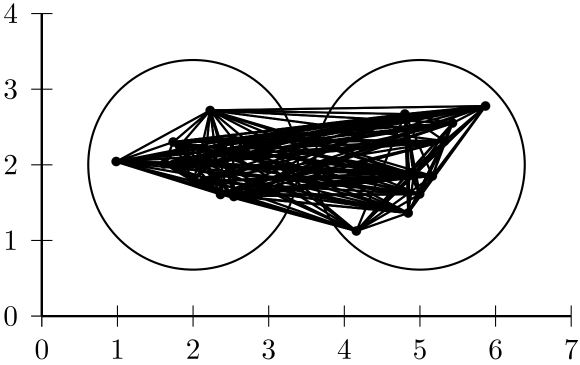
Complete-link: Minimum similarity



Centroid: Average intersimilarity



Group average: Average intrasimilarity

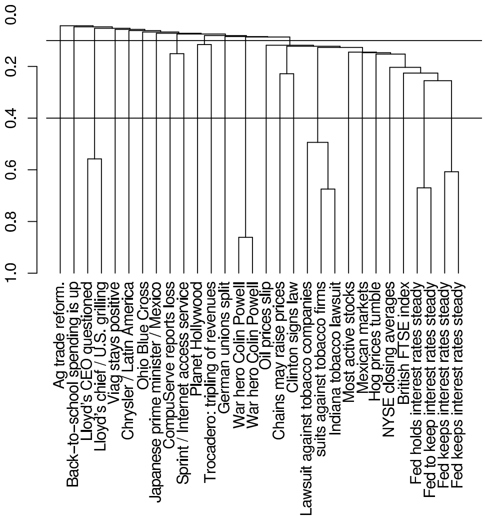


Single link HAC

- The similarity of two clusters is the maximum intersimilarity – the maximum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?
- This is simple for single link:

$$\text{SIM}(\omega_i, (\omega_{k_1} \cup \omega_{k_2})) = \max(\text{SIM}(\omega_i, \omega_{k_1}), \text{SIM}(\omega_i, \omega_{k_2}))$$

This dendrogram was produced by single-link



- Notice: many small clusters (1 or 2 members) being added to the main cluster
- There is no balanced 2-cluster or 3-cluster clustering that can be derived by cutting the dendrogram.

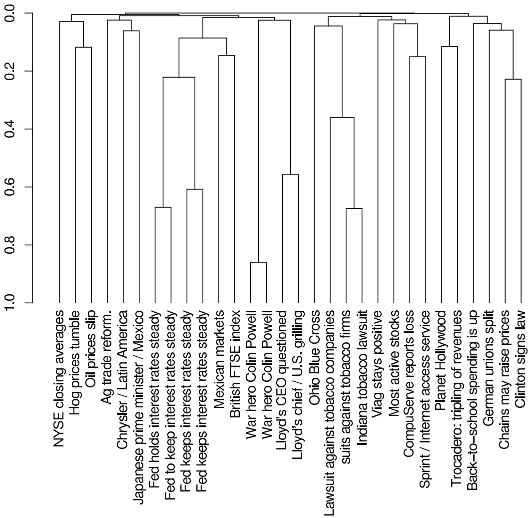
Complete link HAC

- The similarity of two clusters is the minimum intersimilarity – the minimum similarity of a document from the first cluster and a document from the second cluster.
- Once we have merged two clusters, how do we update the similarity matrix?
- Again, this is simple:

$$\text{SIM}(\omega_i, (\omega_{k_1} \cup \omega_{k_2})) = \min(\text{SIM}(\omega_i, \omega_{k_1}), \text{SIM}(\omega_i, \omega_{k_2}))$$

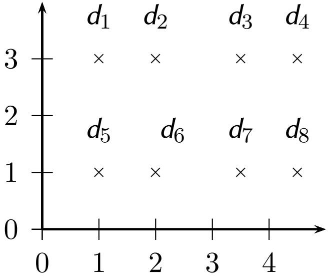
- We measure the similarity of two clusters by computing the diameter of the cluster that we would get if we merged them.

Complete-link dendrogram

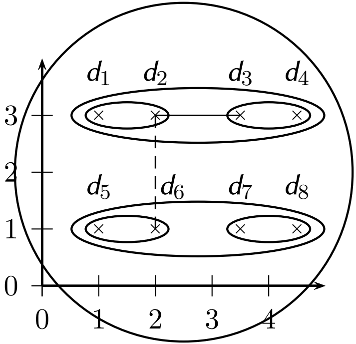


- Notice that this dendrogram is much more balanced than the single-link one.
- We can create a 2-cluster clustering with two clusters of about the same size.

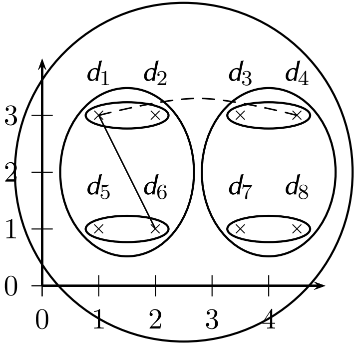
Exercise: Compute single and complete link clusterings



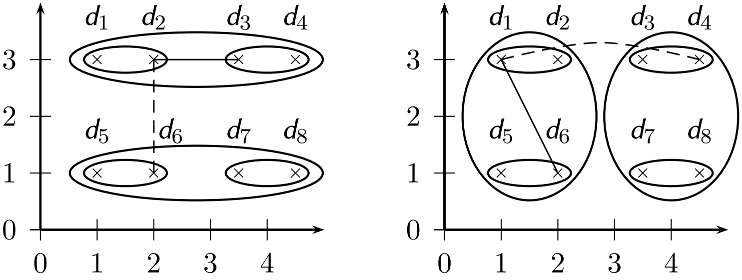
Single-link clustering



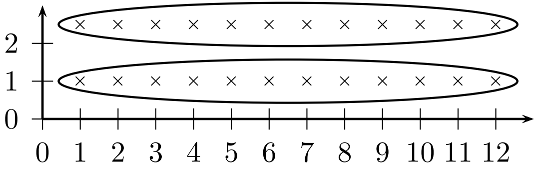
Complete link clustering



Single-link vs. Complete link clustering

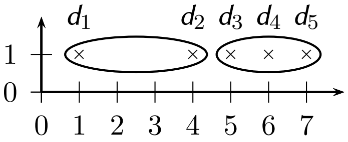


Single-link: Chaining



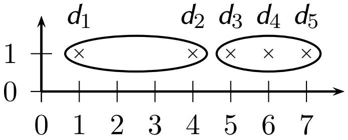
Single-link clustering often produces long, stragglyclusters. For most applications, these are undesirable.

What 2-cluster clustering will complete-link produce?



Coordinates: $1 + 2 \times \epsilon, 4, 5 + 2 \times \epsilon, 6, 7 - \epsilon$.

Complete-link: Sensitivity to outliers



- The complete-link clustering of this set splits d_2 from its right neighbors – clearly undesirable.
- The reason is the outlier d_1.
- This shows that a single outlier can negatively affect the outcome of complete-link clustering.
- Single-link clustering does better in this case.

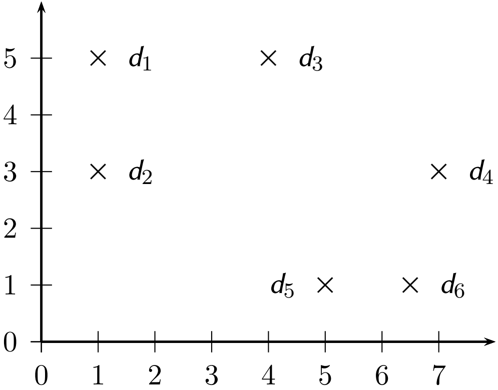
Centroid HAC

- The similarity of two clusters is the average intersimilarity – the average similarity of documents from the first cluster with documents from the second cluster.
- A naive implementation of this definition is inefficient ($O(N^2)$), but the definition is equivalent to computing the similarity of the centroids:

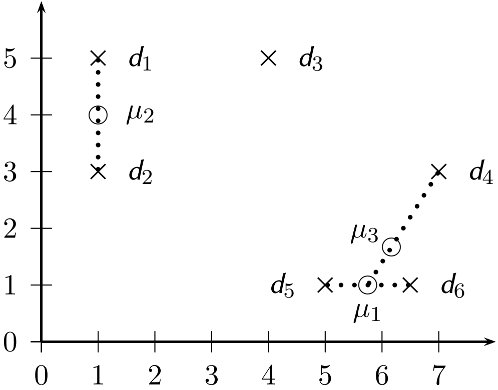
$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

- Hence the name: centroid HAC
- Note: this is the dot product, not cosine similarity!

Exercise: Compute centroid clustering

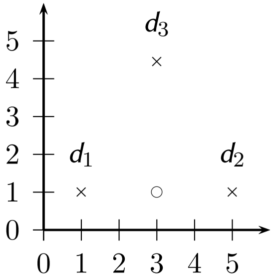


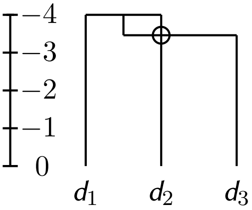
Centroid clustering



Inversion in centroid clustering

- In an inversion, the similarity increases during a merge sequence. Results in an "inverted" dendrogram.
- Below: Similarity of the first merger ($d_1 \cup d_2$) is -4.0, similarity of second merger ($(d_1 \cup d_2) \cup d_3$) is ≈ -3.5.





Inversions

- Hierarchical clustering algorithms that allow inversions are inferior.
- The rationale for hierarchical clustering is that at any given point, we've found the most coherent clustering for a given K.
- Intuitively: smaller clusterings should be more coherent than larger clusterings.
- An inversion contradicts this intuition: we have a large cluster that is more coherent than one of its subclusters.
- The fact that inversions can occur in centroid clustering is a reason not to use it.

Group-average agglomerative clustering (GAAC)

- GAAC also has an “average-similarity” criterion, but does not have inversions.
- The similarity of two clusters is the average intrasimilarity – the average similarity of all document pairs (including those from the same cluster).
- But we exclude self-similarities.

Group-average agglomerative clustering (GAAC)

- Again, a naive implementation is inefficient ($O(N^2)$) and there is an equivalent, more efficient, centroid-based definition:

$$\textsc{sim-ga}(\omega_i, \omega_j) = \frac{1}{(N_i + N_j)(N_i + N_j - 1)}[(\sum_{d_m \in \omega_i \cup \omega_j} \vec{d}_m)^2 - (N_i + N_j)]$$

- Again, this is the dot product, not cosine similarity.

Which HAC clustering should I use?

- Don't use centroid HAC because of inversions.
- In most cases: GAAC is best since it isn't subject to chaining and sensitivity to outliers.
- However, we can only use GAAC for vector representations.
- For other types of document representations (or if only pairwise similarities for documents are available): use complete-link.
- There are also some applications for single-link (e.g., duplicate detection in web search).

Flat or hierarchical clustering?

- For high efficiency, use flat clustering (or perhaps bisecting *k*-means)
- For deterministic results: HAC
- When a hierarchical structure is desired: hierarchical algorithm
- HAC also can be applied if K cannot be predetermined (can start without knowing K)

Variants

Bisecting K-means: A top-down algorithm

- Start with all documents in one cluster
- Split the cluster into 2 using K-means
- Of the clusters produced so far, select one to split (e.g. select the largest one)
- Repeat until we have produced the desired number of clusters

Bisecting K-means

$\textsc{BisectingKMeans}(d_1, \ldots, d_N)$

$\quad \omega_0 \leftarrow \{\vec{d}_1, \ldots, \vec{d}_N\}$

$\quad \textit{leaves} \leftarrow \{\omega_0\}$

$\quad \textbf{for } k \leftarrow 1 \textbf{ to } K-1$

$\quad \textbf{do } \omega_k \leftarrow \textsc{PickClusterFrom}(\textit{leaves})$

$\qquad \{\omega_i, \omega_j\} \leftarrow \textsc{KMeans}(\omega_k, 2)$

$\qquad \textit{leaves} \leftarrow \textit{leaves} \setminus \{\omega_k\} \cup \{\omega_i, \omega_j\}$

$\quad \textbf{return } \textit{leaves}$

Bisecting K-means

- If we don't generate a complete hierarchy, then a top-down algorithm like bisecting K-means is much more efficient than HAC algorithms.
- But bisecting K-means is not deterministic.
- There are deterministic versions of bisecting K-means (see resources at the end), but they are much less efficient.

Efficient single link clustering

```
SingleLinkClustering(d_1, ..., d_N, K)
for n ← 1 to N
do for i ← 1 to N
    do C[n][i].sim ← SIM(d_n, d_i)
       C[n][i].index ← i
    I[n] ← n
    NBM[n] ← arg max_{X∈{C[n][i]:n≠i}} X.sim
A ← []
for n ← 1 to N − 1
do i_1 ← arg max_{{i:I[i]=i}} NBM[i].sim
    i_2 ← I[NBM[i_1].index]
    A.Append(⟨i_1, i_2⟩)
    for i ← 1 to N
    do if I[i] = i ∧ i ≠ i_1 ∧ i ≠ i_2
         then C[i_1][i].sim ← C[i][i_1].sim ← max(C[i_1][i].sim, C[i_2][i].sim)
       if I[i] = i_2
         then I[i] ← i_1
    NBM[i_1] ← arg max_{X∈{C[i_1][i]:I[i]=i∧i≠i_1}} X.sim
return A
```

Time complexity of HAC

- The single-link algorithm we just saw is $O(N^2)$.
- Much more efficient than the $O(N^3)$ algorithm we looked at earlier!
- There is no known $O(N^2)$ algorithm for complete-link, centroid and GAAC.
- Best time complexity for these three is $O(N^2 \log N)$: See book.
- In practice: little difference between $O(N^2 \log N)$ and $O(N^2)$.

Combination similarities of the four algorithms

clustering algorithm	$\text{sim}(\ell, k_1, k_2)$
single-link	$\max(\text{sim}(\ell, k_1), \text{sim}(\ell, k_2))$
complete-link	$\min(\text{sim}(\ell, k_1), \text{sim}(\ell, k_2))$
centroid	$(\frac{1}{N_m}\vec{v}_m) \cdot (\frac{1}{N_\ell}\vec{v}_\ell)$
group-average	$\frac{1}{(N_m+N_\ell)(N_m+N_\ell-1)}[(\vec{v}_m + \vec{v}_\ell)^2 - (N_m + N_\ell)]$

Comparison of HAC algorithms

method	combination similarity	time compl.	optimal?	comment
single-link	max intersimilarity of any 2 docs	$\Theta(N^2)$	yes	chaining effect
complete-link	min intersimilarity of any 2 docs	$\Theta(N^2 \log N)$	no	sensitive to outliers
group-average	average of all sims	$\Theta(N^2 \log N)$	no	best choice for most applications
centroid	average intersimilarity	$\Theta(N^2 \log N)$	no	inversions can occur

What to do with the hierarchy?

- Use as is (e.g., for browsing as in Yahoo hierarchy)
- Cut at a predetermined threshold
- Cut to get a predetermined number of clusters K
  - Ignores hierarchy below and above cutting line.