Introduction to Information Retrieval
`http://www.informationretrieval.org`

Hierarchical Clustering

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

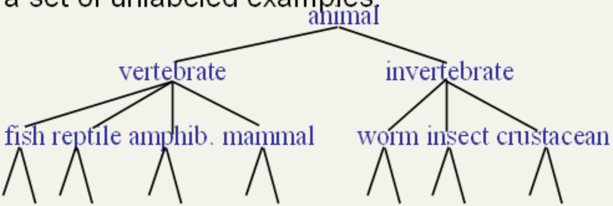
2007.06.19

Overview

Introduction

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples.



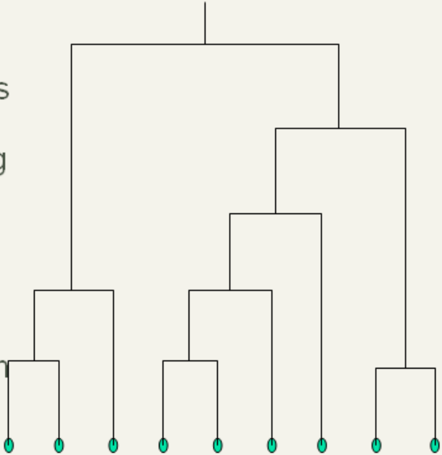
- One option to produce a hierarchical clustering is recursive application of a partitional clustering algorithm to produce a hierarchical clustering.

Hierarchical Agglomerative Clustering (HAC)

- Assumes a similarity function for determining the similarity of two instances.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

A Dendogram: Hierarchical Clustering

- Dendrogram: Decomposes data objects into a several levels of nested partitioning (tree of clusters).
- Clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



Divisive clustering

- Top-down (instead of bottom-up as in HAC)
- Start with all docs in one big cluster
- The recursively split clusters
- Eventually each node forms a cluster on its own

Naive HAC algorithm

- Given
  - N one-document clusters
  - a similarity measure between clusters
- $I := N$
- Repeat until I==1:
  - Compute I^2 similarities between all clusters
  - Find cluster pair with highest similarity
  - Merge the two clusters
  - $I := I - 1$
- $N - 1$ iterations until we have a single cluster with all docs

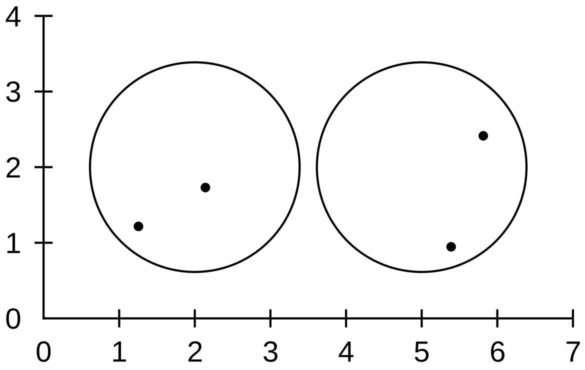
Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- In each of the subsequent n–2 merging iterations, it must compute the distance between the most recently created cluster and all other existing clusters.
  - Since we can just store unchanged similarities
- In order to maintain an overall $O(n^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

Key question: How to define cluster similarity

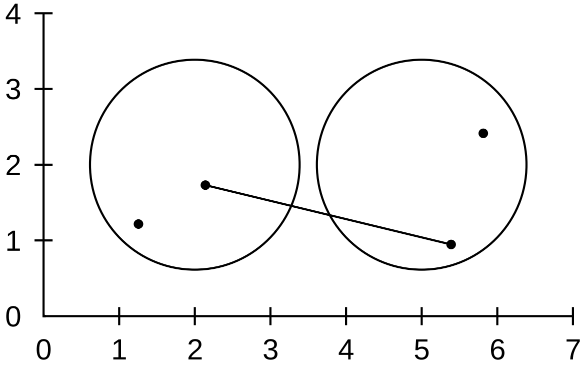
- Single-link: Maximum similarity
  - Maximum over all ω_1-ω_2-pairs
- Complete-link: Minimum similarity
  - Minimum over all ω_1-ω_2-pairs
- Centroid: Average similarity
  - Average over all ω_1-ω_2-pairs
- Group-average: Average similarity
  - Average over all document pairs, including pairs of docs in the same cluster

Cluster similarity: Example

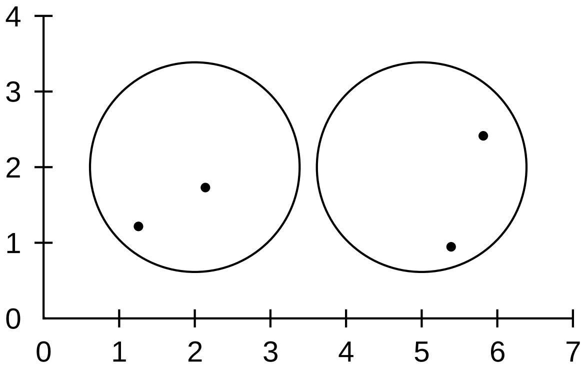


- Single-link?

Single-link: Maximum similarity

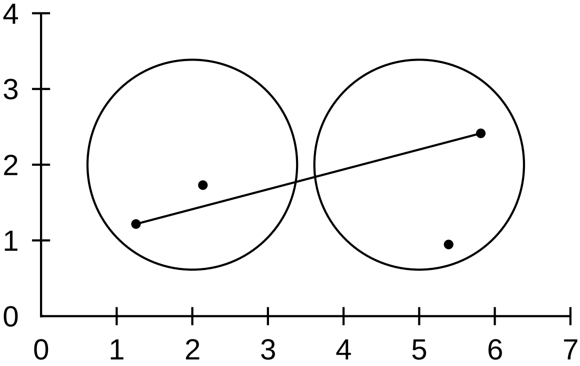


Cluster similarity: Example

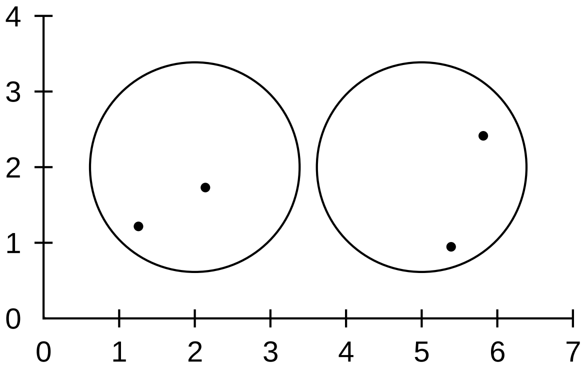


- Complete-link?

Complete-link: Minimum similarity

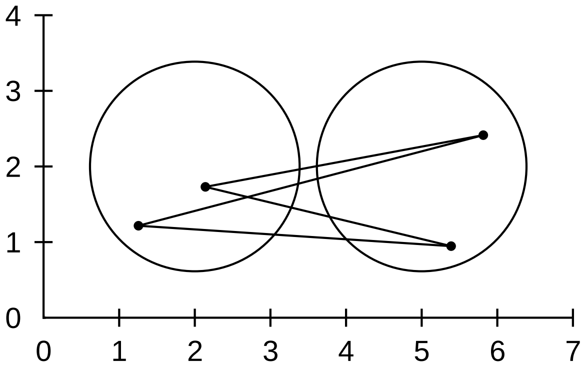


Cluster similarity: Example

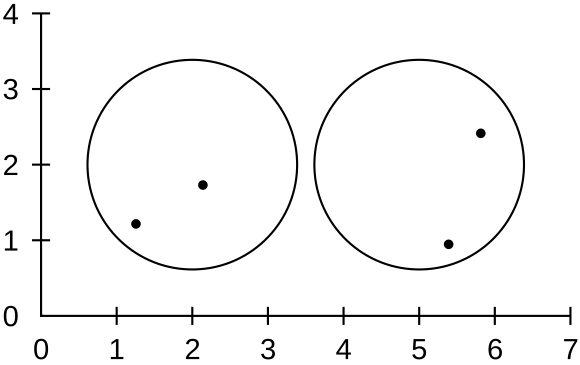


- Centroid?

Centroid: Average similarity (exc. within cluster)

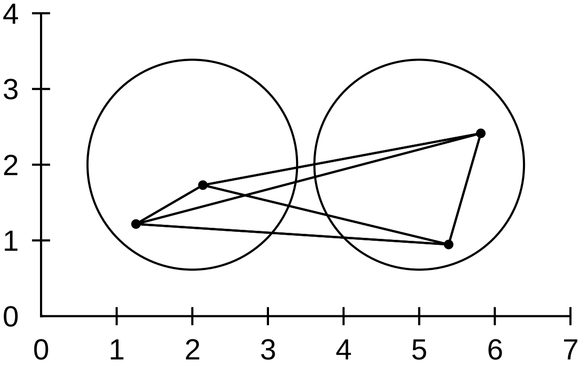


Cluster similarity: Example

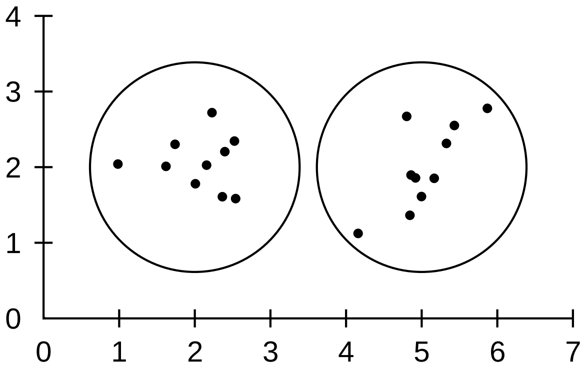


- Group-average?

Group average: Average similarity (inc. within cluster)

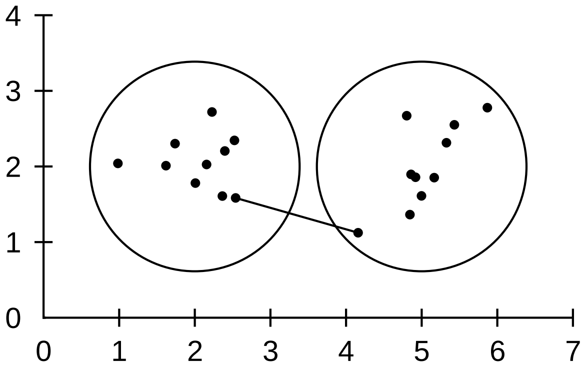


Cluster similarity: Larger example

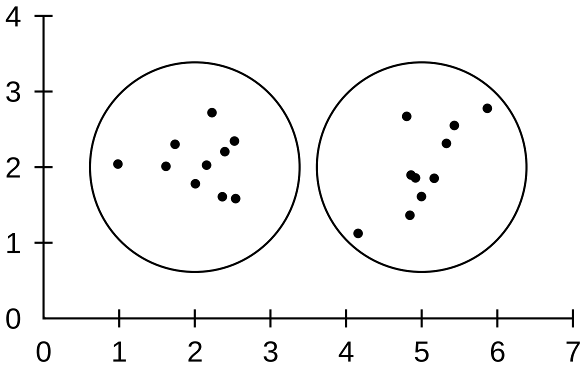


- Single-link?

Single-link: Maximum similarity

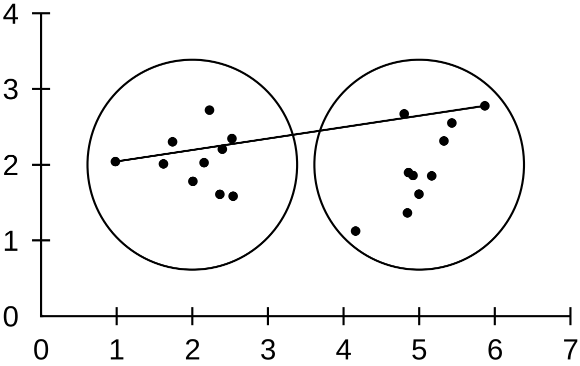


Cluster similarity: Larger example

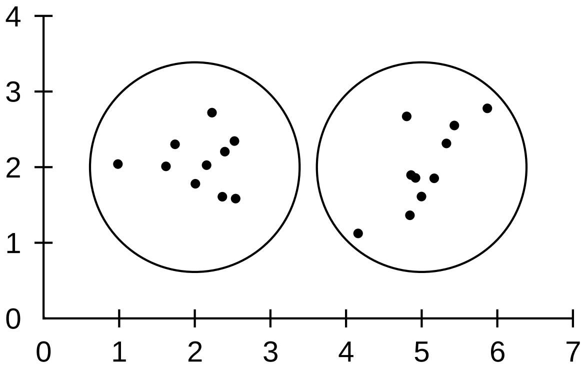


- Complete-link?

Complete-link: Minimum similarity

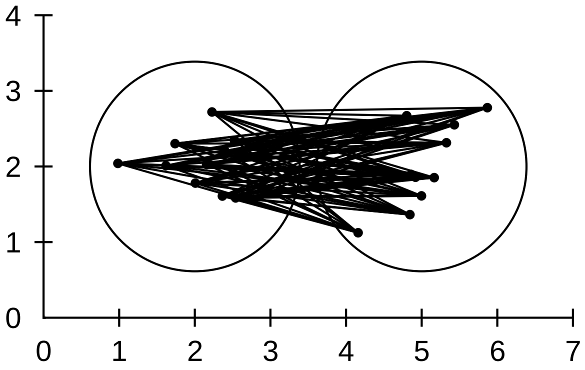


Cluster similarity: Larger example

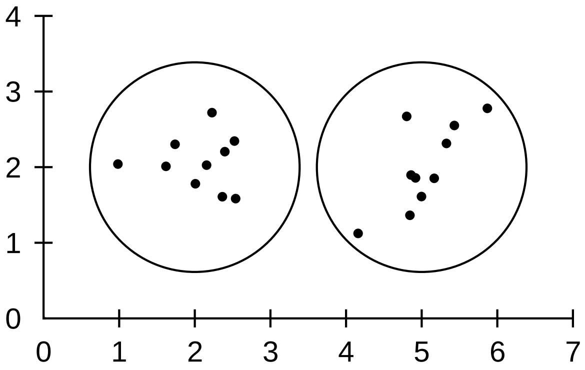


- Centroid?

Centroid: Average similarity (exc. within cluster)

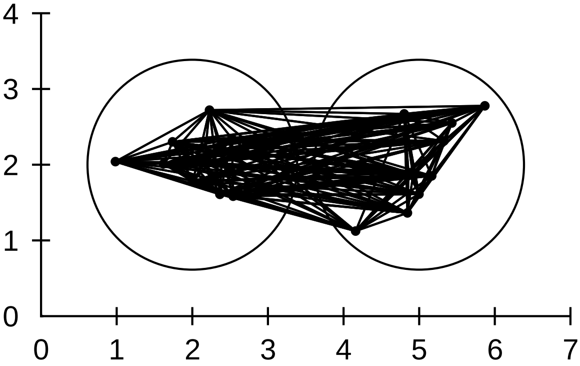


Cluster similarity: Larger example



- Group-average?

Group average: Average similarity (inc. within cluster)



Single-link/Complete-link

Single Link Agglomerative Clustering

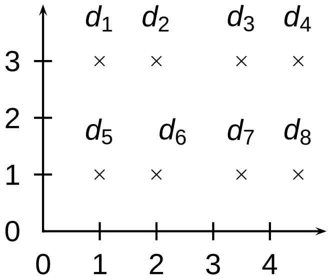
- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

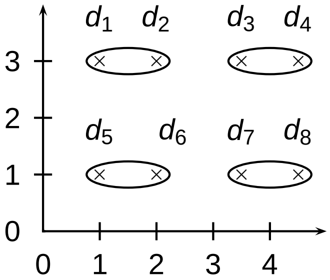
- Can result in "straggly" (long and thin) clusters due to chaining effect.
  - Appropriate in some domains, such as clustering islands: "Hawai'i clusters"
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

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

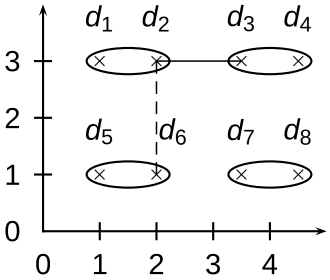
Single-link clustering: Example



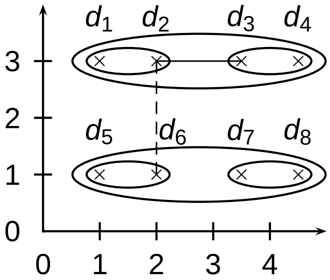
Single-link clustering: Example



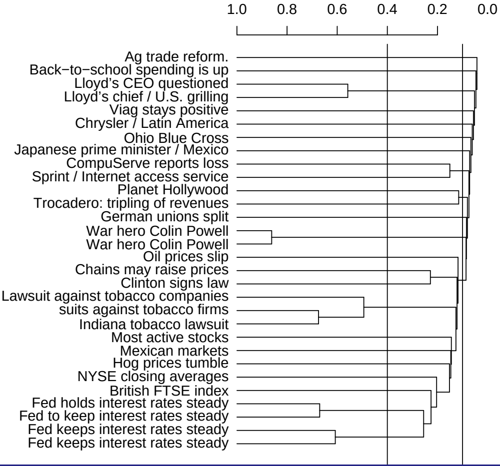
Single-link clustering: Example



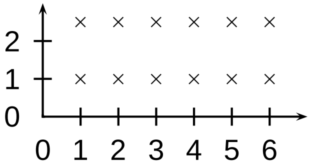
Single-link clustering: Example



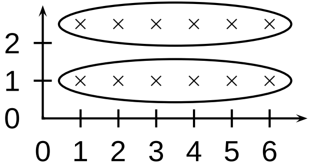
Single-link clustering: Document example



What cluster structure after 10 mergers?



Single-link: Chaining



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

Complete Link Agglomerative Clustering

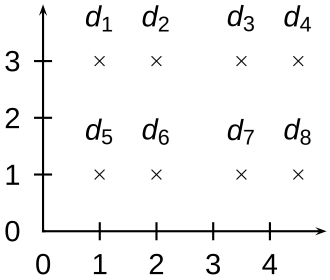
- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

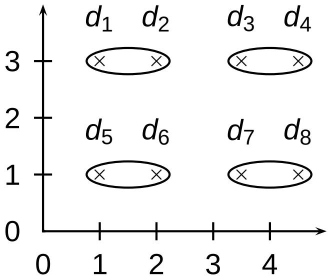
- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

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

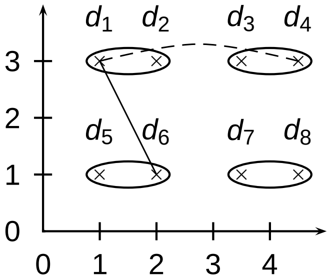
Complete link clustering: Example



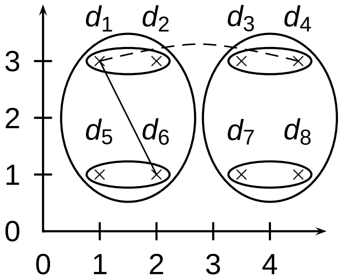
Complete link clustering: Example



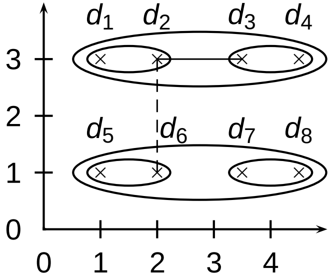
Complete link clustering: Example

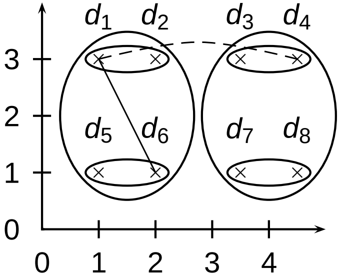


Complete link clustering: Example

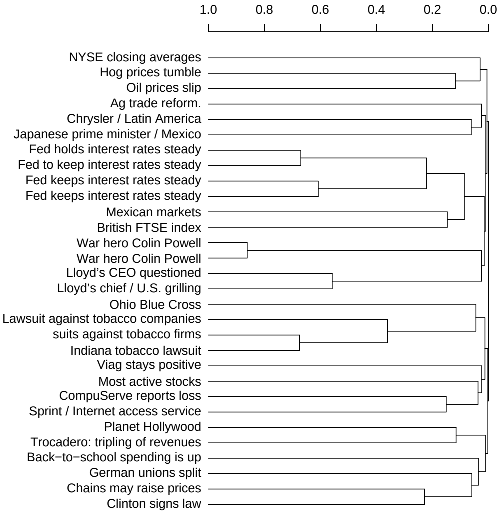


Single-link vs. Complete link clustering

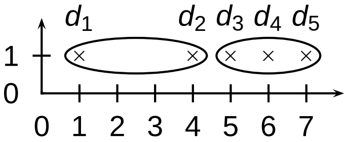




Complete-link clustering: Document example



Complete-link: Sensitivity to outliers



A single outlier can have a large effect on the final outcome of complete-link clustering. Coordinates: $1 + 2 \times \epsilon, 4, 5 + 2 \times \epsilon, 6, 7 - \epsilon$. GAAC (group-average HAC) is better here.

GAAC/Centroid

Group-average agglomerative clustering (GAAC)

$$\text{sim-ga}(\omega_i, \omega_j) = \frac{1}{(N_i+N_j)(N_i+N_j-1)} \sum_{[d_k \in \omega_i \cup \omega_j]} \sum_{[d_\ell \in \omega_i \cup \omega_j, d_\ell \neq d_k]} \vec{d}_k \cdot \vec{d}_\ell$$

- The similarity of two clusters is the average of all pairwise doc similarities – except for self-similarities.
- Avoids the problems of single-link and complete-link.
- Usually the best option for HAC

Efficient computation of similarity in GAAC

$$\begin{aligned}\text{sim-ga}(\omega_i, \omega_j) &= \frac{1}{(N_i+N_j)(N_i+N_j-1)} \sum_{[d_k \in \omega_i \cup \omega_j]} \sum_{[d_\ell \in \omega_i \cup \omega_j, d_\ell \neq d_k]} \vec{d}_k \cdot \vec{d}_\ell \\ &= \frac{1}{(N_i+N_j)(N_i+N_j-1)} [(\sum_{d_k \in \omega_i \cup \omega_j} \vec{d}_k)^2 - (N_i+N_j)]\end{aligned}$$

- Holds because of distributivity of the scalar product with respect to vector addition
- Makes similarity computation efficient

Computing Group Average Similarity

- Assume cosine similarity and normalized vectors with unit length.
- Always maintain sum of vectors in each cluster.

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

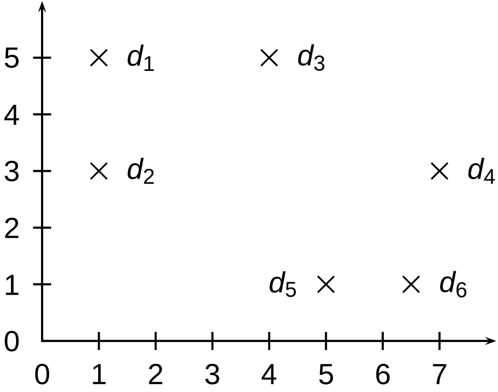
- Compute similarity of clusters in constant time:

$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

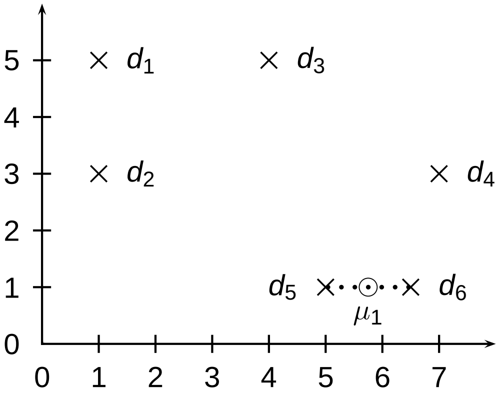
Centroid HAC

- The similarity of two clusters is the scalar product of their centroids.
- Alternative: (negative) distance of the centroids – can result in different clustering.

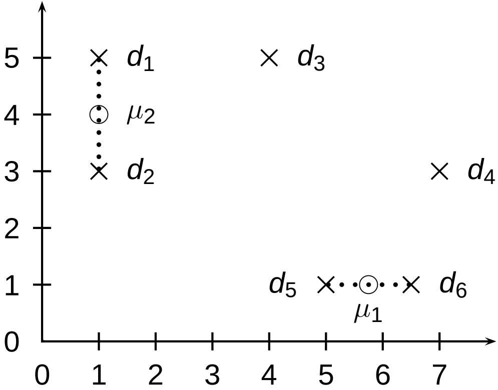
Centroid clustering: Example



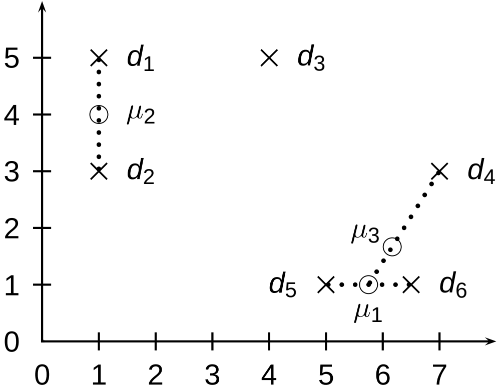
Centroid clustering: Example



Centroid clustering: Example



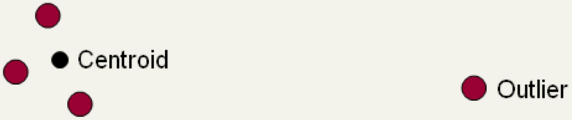
Centroid clustering: Example



Outliers in centroid computation

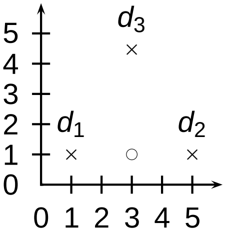
- Can ignore outliers when computing centroid.
- What is an outlier?
  - Lots of statistical definitions, e.g.
  - *moment* of point to centroid > M × some cluster *moment*.

Say 10.



Inversion in centroid clustering

- In an inversion, the similarity increases during a merge sequence. Causes an inversion of the 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.



GAAC vs. Centroid clustering

$$\text{sim-cent}(\omega_i, \omega_j) = (\frac{1}{N_i} \sum_{d_k \in \omega_i} \vec{d}_k) \cdot (\frac{1}{N_j} \sum_{d_\ell \in \omega_j} \vec{d}_\ell) = \frac{1}{N_i N_j} \sum_{d_k \in \omega_i} \sum_{d_\ell \in \omega_j} \vec{d}_k \cdot \vec{d}_\ell$$

- GAAC: average of all similarities, including within original two clusters (except self-similarities)
- Centroid clustering: average similarity over pairs of docs where one doc is from ω_i and one doc is from ω_j

Implementation

Efficient HAC algorithm

Given: N length-normalized vectors $\vec{v}_i$
Compute matrix C
for $k = 1$ to N:
  for $\ell = 1$ to N:
    $C[k][\ell]$.sim $= \vec{v}_k \cdot \vec{v}_\ell$
    $C[k][\ell]$.index $= \ell$
for $k = 1$ to N:
  $P[k]$:= priority queue for $C[k]$ sorted on sim
  Delete $C[k][k]$ from $P[k]$ *(don't want self-similarities)*
Initialization
$A = [\,]$
for $k = 1$ to N:
  $I[k] = 1$
Compute clustering
for $k = 1$ to $N - 1$:
  $k_1 = \arg\max_{k, I[k]=1} P[k]$.max()
  $k_2 = P[k_1]$.max().index
  A.append($\langle k_1, k_2 \rangle$)
  $I[k_2] = 0$
  $P[k_1] = \emptyset$
  for all ℓ with $I[\ell] = 1, \ell \neq k_1$:
    $C[k_1][\ell]$.sim $= C[\ell][k_1]$.sim $=$ sim(ℓ, k_1, k_2)
    Delete $C[\ell][k_1]$ and $C[\ell][k_2]$ from $P[\ell]$
    Insert $C[\ell][k_1]$ in $P[\ell]$, $C[k_1][\ell]$ in $P[k_1]$

Combination similarities for HAC 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	$\frac{1}{(N_m+N_\ell)(N_m+N_\ell-1)}[(\vec{v}_m + \vec{v}_\ell)^2 - (N_m + N_\ell)]$

One iteration in efficient HAC algorithm

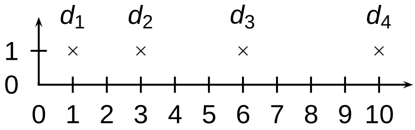
compute $C[5]$	1 0.2	2 0.8	3 0.6	4 0.4
create $P[5]$ (by sorting)	2 0.8	3 0.6	4 0.4	1 0.2
merge 2 and 3, update similarity of 2, delete 3	2 0.3	4 0.4	1 0.2	
reinsert 2	4 0.4	2 0.3	1 0.2	

Efficient single-link clustering

Given: N one-document clusters
Compute similarity matrix
for $k = 1$ to N:
  for $\ell = 1$ to N:
    $C[k][\ell] = \mathrm{sim}(d_k, d_\ell)$
Initialization
$A = [\,]$
for $k = 1$ to N:
  $I[k] = 1$
  $\mathrm{NBM}[k].\mathrm{index} = \arg\max_{i, i \neq k} C[k][i]$
  $\mathrm{NBM}[k].\mathrm{sim} = C[k][\mathrm{NBM}[k].\mathrm{index}]$
Compute clustering
for $n = 1$ to $N - 1$:
  $k_1 = \arg\max_{i, I[i]=1} \mathrm{NBM}[i].\mathrm{sim}$
  $k_2 = \mathrm{NBM}[k_1].\mathrm{index}$
  $A.\mathrm{append}(\langle k_1, k_2 \rangle)$
  $k_{\mathrm{min}} = \arg\min_{i = k_1, k_2} \mathrm{NBM}[i].\mathrm{sim}$
  $I[k_{\mathrm{min}}] = 0$

Complete link clustering ist not merge-persistent

- Is there an $O(n^2)$ algorithm for complete-link and GAAC? No!
- Single-link is merge-persistent, complete-link and GAAC are not.
- Merge-persistent: If d_2 is best merge candidate for d_3, then after merging d_2 with another cluster d_1, the merger $\{d_1, d_2\}$ will be the best merge candidate for d_3.
- Example below shows that this is not the case for complete-link clustering: The best candidate is d_4.



Efficiency: Medoid As Cluster Representative

- The centroid does not have to be a document.
- Medoid: A cluster representative that is one of the documents
- For example: the document closest to the centroid
- One reason this is useful
  - Consider the representative of a large cluster (>1000 documents)
  - The centroid of this cluster will be a dense vector
  - The medoid of this cluster will be a sparse vector
- Compare: mean/centroid vs. median/medoid

Comparison of HAC algorithms

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

Bisecting K-means

- Divisive hierarchical clustering method using K-means
- For I=1 to k-1 do {
  - Pick a leaf cluster C to split
  - For J=1 to ITER do {
    - Use K-means to split C into two sub-clusters, C_1 and C_2
    - Choose the best of the above splits and make it permanent}

    }

  }

- Steinbach et al. suggest HAC is better than k-means but Bisecting K-means is better than HAC for their text experiments

Exercise

- Consider running 2-means clustering on a corpus, each doc of which is from one of two different languages. What are the two clusters we would expect to see?
- Is HAC likely to produce results different to the above?

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
- Hierarchical clustering is often used to get K flat clusters. The hiearchy is then ignored.

Labeling

Major issue - labeling

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say "Animal" or "Car" in the ***jaguar*** example.
  - In topic trees (Yahoo), need navigational cues.
    - Often done by hand, a posteriori.

Ideas?

How to Label Clusters

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling
  - But harder to scan

Labeling

- Common heuristics - list 5-10 most frequent terms in the centroid vector.
  - Drop stop-words; stem.
- Differential labeling by frequent terms
  - Within a collection "Computers", clusters all have the word ***computer*** as frequent term.
  - Discriminant analysis of centroids.
- Perhaps better: distinctive noun phrase

Cluster labeling: Example

	# docs	labeling method		
		centroid	mutual information	title
4	622	oil plant mexico production crude **power 000 refinery gas** bpd	plant oil production **barrels** crude bpd mexico **dolly capacity petroleum**	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security **russian** people military peace killed told **grozny court**	police killed military security peace told **troops forces rebels** people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices **cents september** tonne	**delivery** traders futures tonne tonnes **desk** wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

- Three methods: most prominent terms in centroid, differential labeling using MI, title of doc closest to centroid
- Any feature selection method can also be used for labeling

Flat or hierarchical clustering?

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

Resources

- IIR 17
- Data clustering: A review. A. K. Jain, M. N. Murty and P. J. Flynn. ACM Computering Surveys, 1999
- A comparison of document clustering techniques. Michael Steinbach, George Karypis and Vipin Kumar. KDD Workshop on Text Mining, 2000