Hierarchical Clustering

Most slides are from Hinrich Schütze & Lucia D. Krisnawati

March 11, 2017

Overview

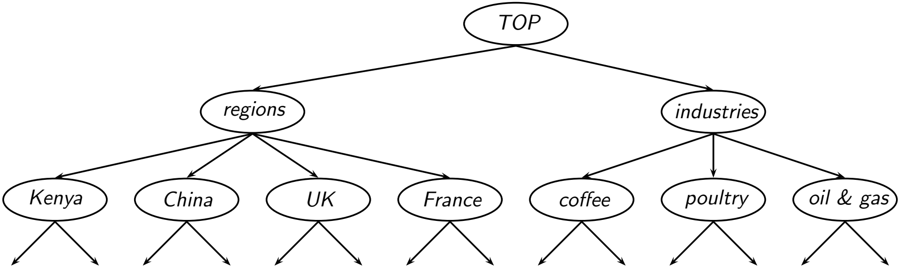
Outline

- Introduction to hierarchical clustering
- Single-link and complete-link clustering
- Centroid and group-average agglomerative clustering (GAAC)
- Bisecting K-means
- How to label clusters automatically

Outline

Hierarchical clustering

- Goal: 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.



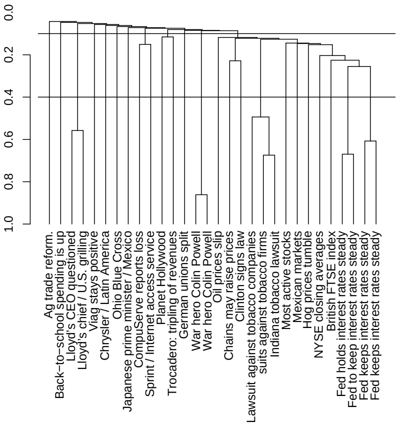
Hierarchical agglomerative clustering (HAC)

- HAC creates a hierarchy in the form of a binary tree.
- Assumes a similarity measure for determining the 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

```
SimpleHAC(d_1, ..., d_N)
for n ← 1 to N
do for i ← 1 to N
     do C[n][i] ← Sim(d_n, d_i)
     I[n] ← 1  (keeps track of active clusters)
A ← []  (collects clustering as a sequence of merges)
for k ← 1 to N − 1
do ⟨i, m⟩ ← arg max_{⟨i,m⟩: i≠m ∧ I[i]=1 ∧ I[m]=1} C[i][m]
     A.Append(⟨i, m⟩)  (store merge)
     for j ← 1 to N
     do   (use i as representative for < i, m >)
          C[i][j] ← Sim(< i, m >, j)
          C[j][i] ← Sim(< i, m >, j)
     I[m] ← 0  (deactivate cluster)
return A
```

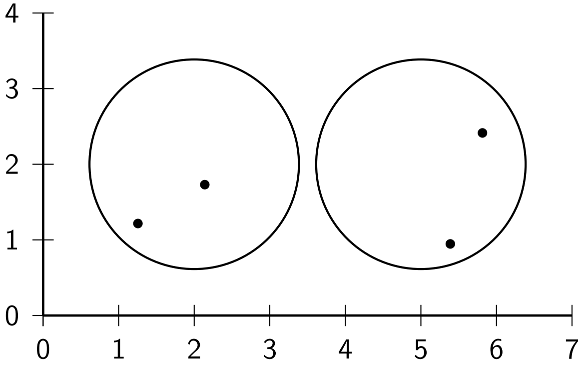
Computational complexity of the naive algorithm

- 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.

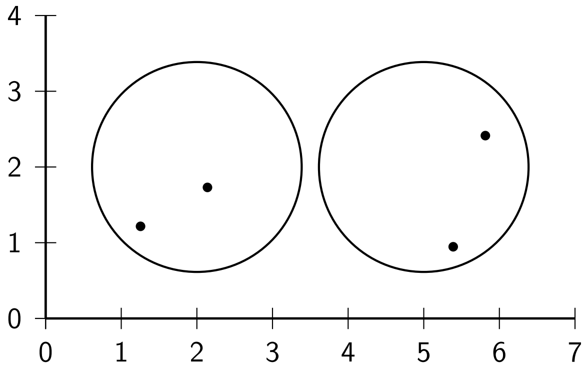
Key question: How to define cluster similarity

- Single-link: Maximum similarity
  - Maximum similarity of any two documents
- Complete-link: Minimum similarity
  - Minimum similarity of any two documents
- Centroid: Average "inter-similarity"
  - 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 similarity of all document pairs, including pairs of docs in the same cluster

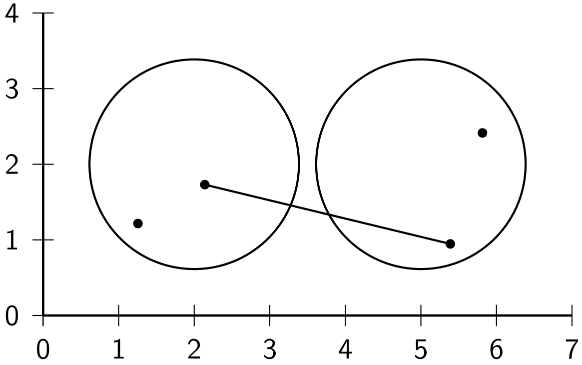
Cluster similarity: Example



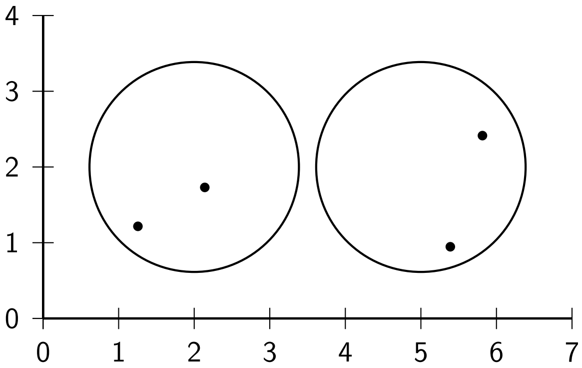
Single-link: Maximum similarity



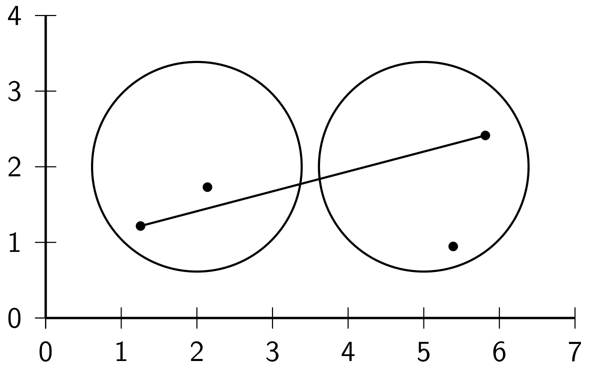
Single-link: Maximum similarity



Complete-link: Minimum similarity

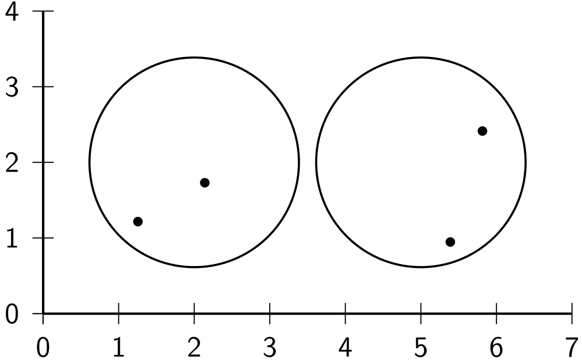


Complete-link: Minimum similarity



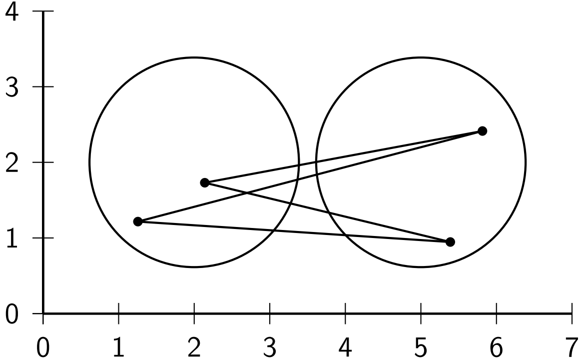
Centroid: Average intersimilarity

intersimilarity = similarity of two documents in different clusters



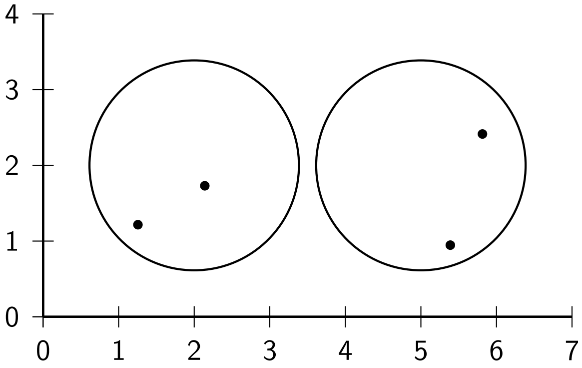
Centroid: Average intersimilarity

intersimilarity = similarity of two documents in different clusters



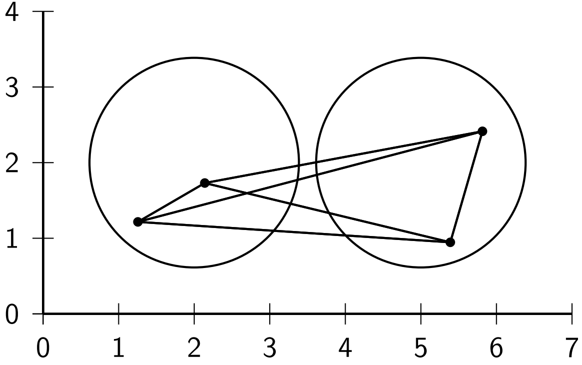
Group average: Average intrasimilarity

intrasimilarity = similarity of any pair, including cases where the two documents are in the same cluster

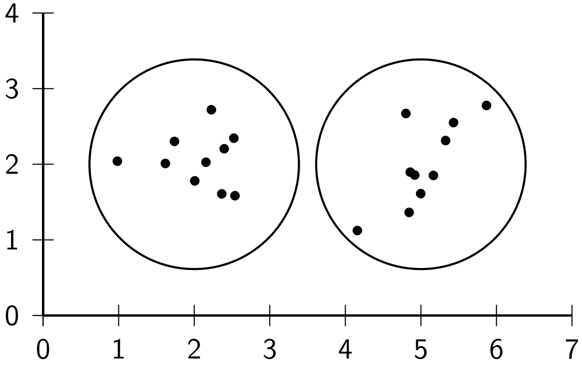


Group average: Average intrasimilarity

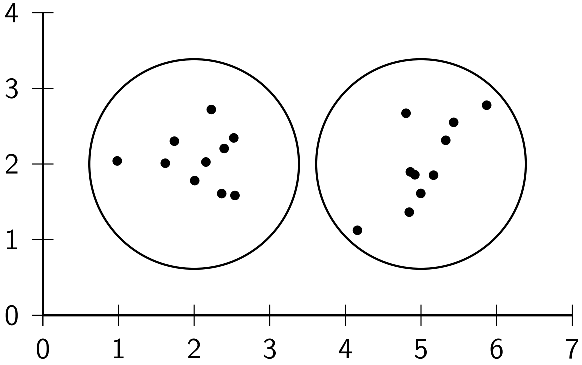
intrasimilarity = similarity of any pair, including cases where the two documents are in the same cluster



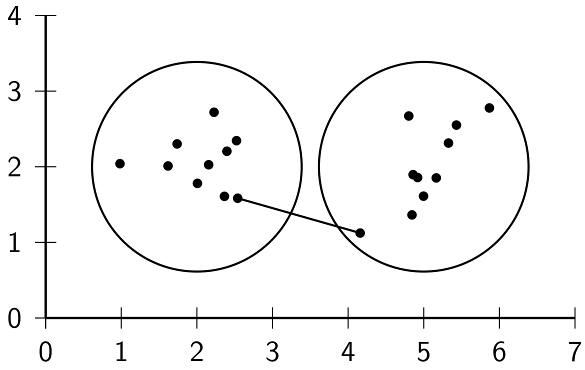
Cluster similarity: Larger Example



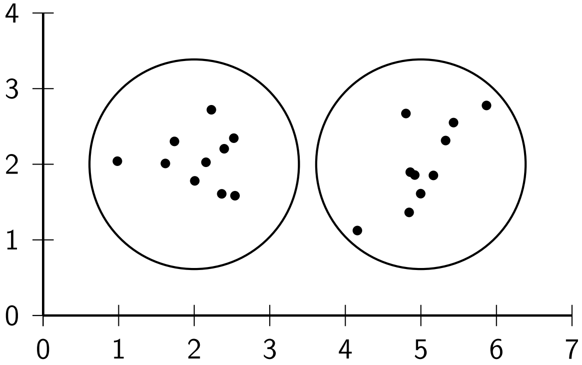
Single-link: Maximum similarity



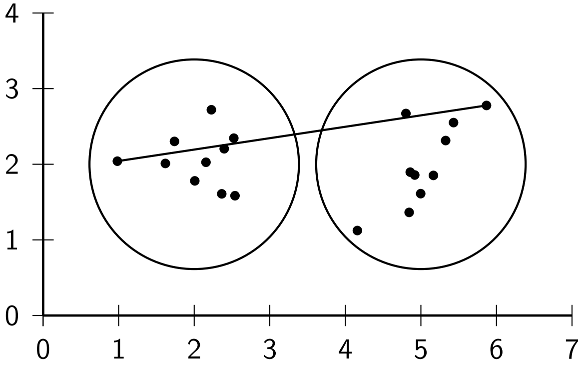
Single-link: Maximum similarity



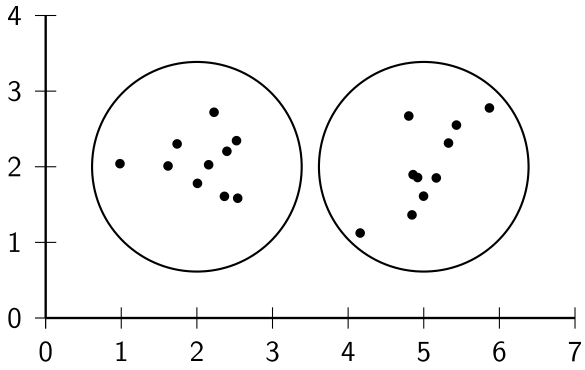
Complete-link: Minimum similarity



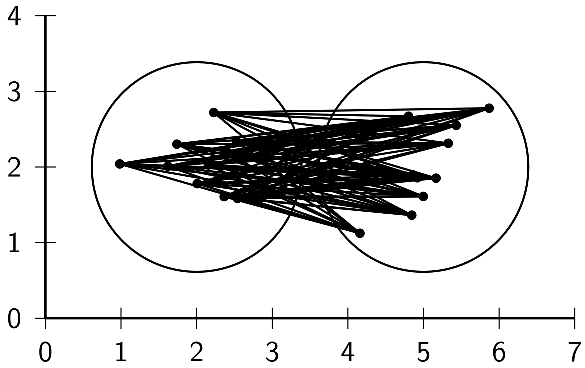
Complete-link: Minimum similarity



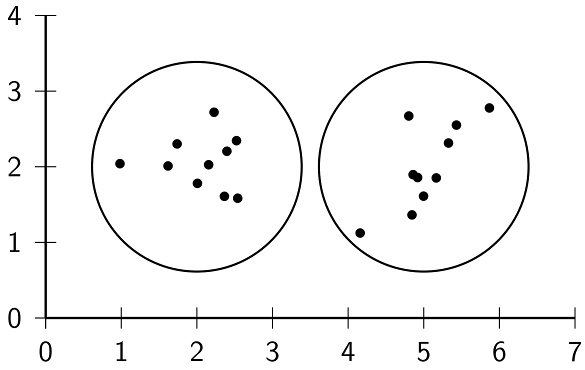
Centroid: Average intersimilarity



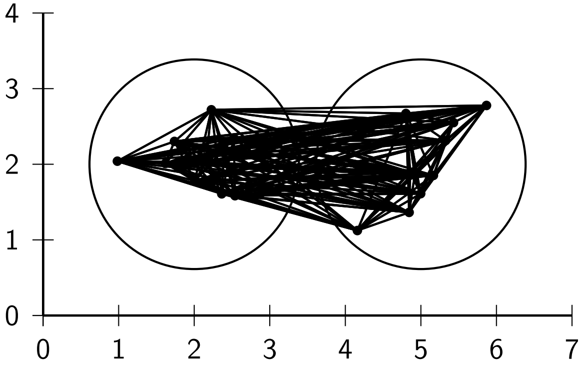
Centroid: Average intersimilarity



Group average: Average intrasimilarity



Group average: Average intrasimilarity



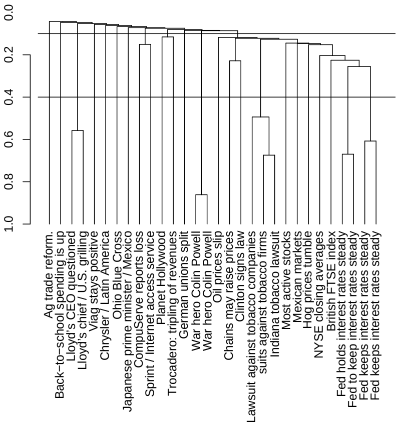
Outline

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.

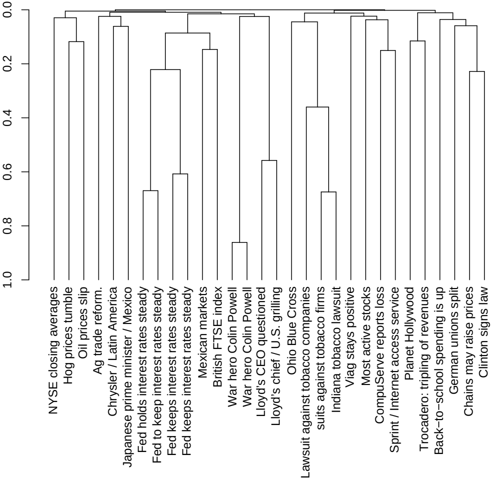
Complete link HAC

- 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:

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

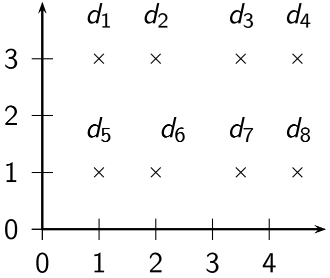
- We measure the similarity of two clusters by computing the diameter of the cluster that we would get if we merged them.

Complete-link dendrogram

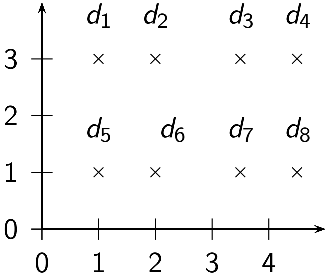


- Notice that this dendrogram is much more balanced than the single-link one.
- We can create a 2-cluster clustering with two clusters of about the same size.

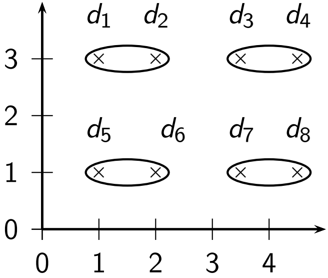
Exercise: Compute single and complete link clusterings



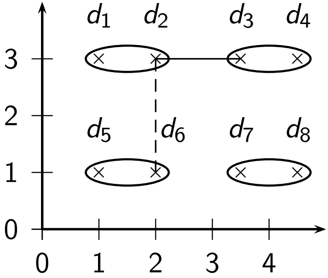
Single-link clustering



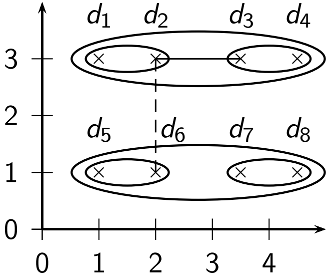
Single-link clustering



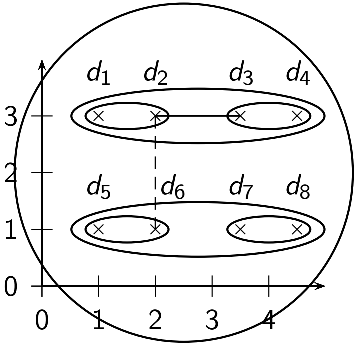
Single-link clustering



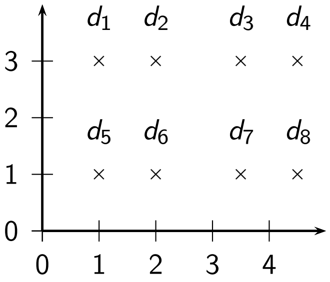
Single-link clustering



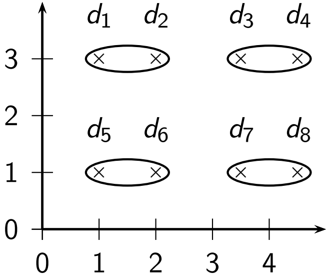
Single-link clustering



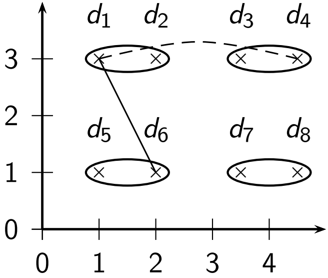
Complete link clustering



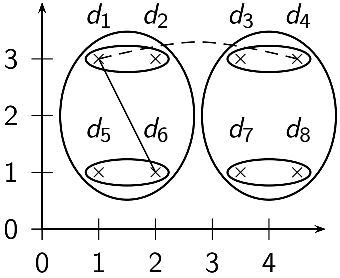
Complete link clustering



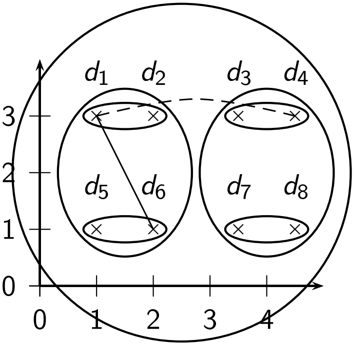
Complete link clustering



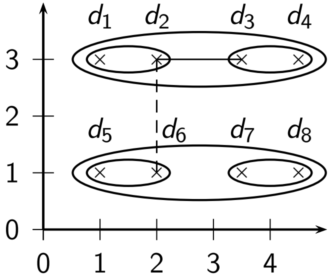
Complete link clustering

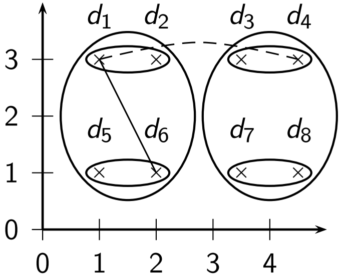


Complete link clustering

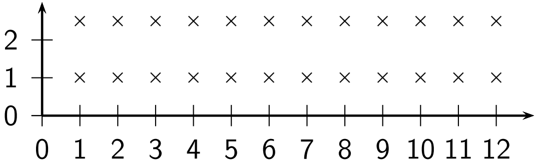


Single-link vs. Complete link clustering

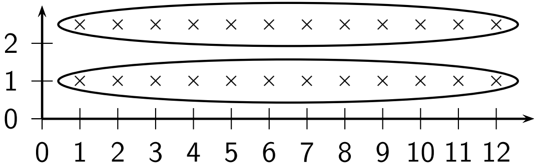




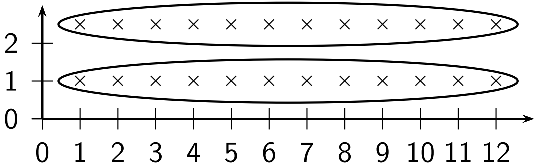
Single-link: Chaining



Single-link: Chaining

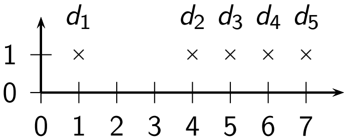


Single-link: Chaining



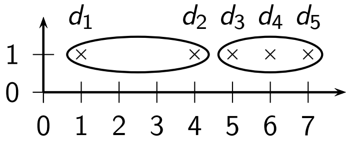
Single-link clustering often produces long, straggly clusters. For most applications, these are undesirable.

What 2-cluster clustering will complete-link produce?



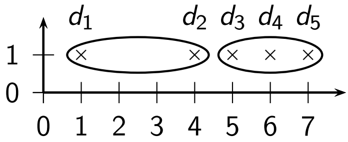
Coordinates: $1 + 2 \times \epsilon, 4, 5 + 2 \times \epsilon, 6, 7 - \epsilon$.

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.

Outline

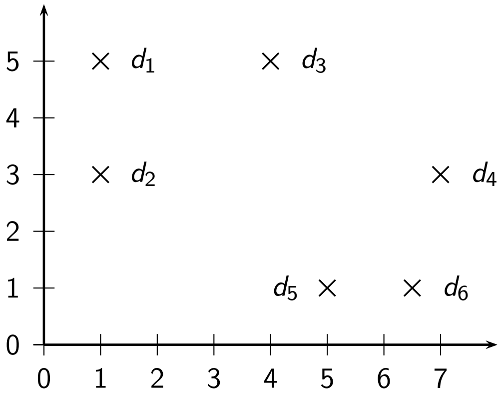
Centroid HAC

- 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:

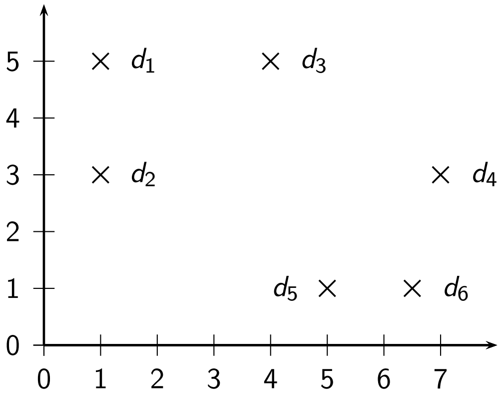
$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

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

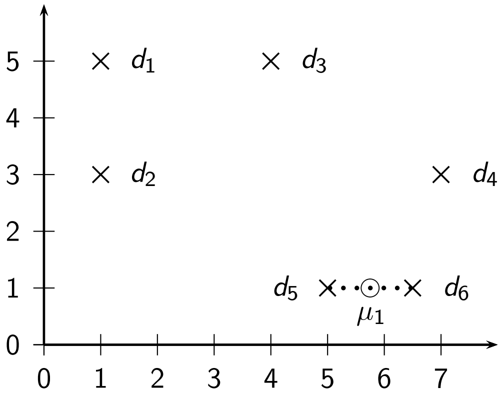
Exercise: Compute centroid clustering



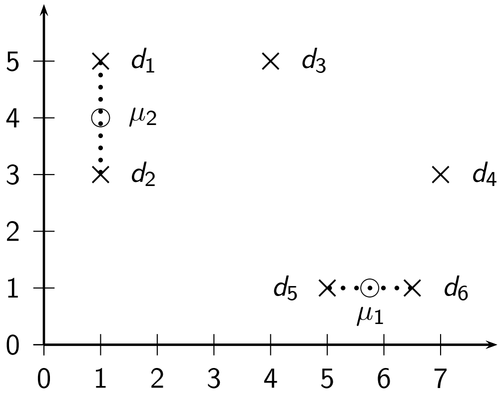
Centroid clustering



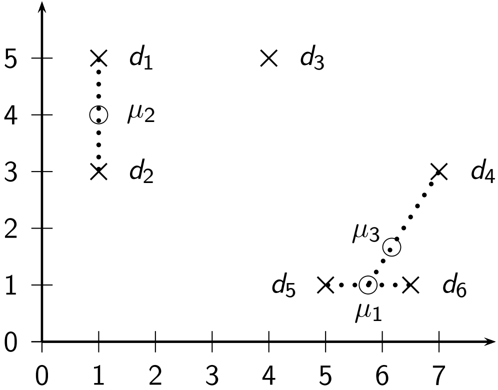
Centroid clustering



Centroid clustering

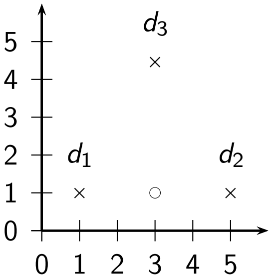


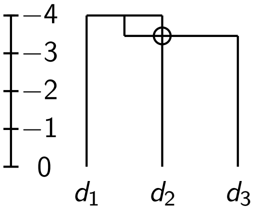
Centroid clustering



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:

$$\text{SIM-GA}(\omega_i, \omega_j) =$$

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

- 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)

Outline

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?
- We need a pithy label for each cluster.
- For example, in search result clustering for "jaguar", The labels of the three clusters could be "animal", "car", and "operating system".
- Topic of this section: How can we automatically find good labels for clusters?

Exercise

- Come up with an algorithm for labeling clusters
- Input: a set of documents, partitioned into K clusters (flat clustering)
- Output: A label for each cluster
- Part of the exercise: What types of labels should we consider? Words?

Discriminative labeling

- To label cluster ω, compare ω with all other clusters
- Find terms or phrases that distinguish ω from the other clusters
- We can use any of the feature selection criteria we introduced in text classification to identify discriminating terms: mutual information, χ^2 and frequency.
- (but the latter is actually not discriminative)

Non-discriminative labeling

- Select terms or phrases based solely on information from the cluster itself
  - E.g., select terms with high weights in the centroid (if we are using a vector space model)
- Non-discriminative methods sometimes select frequent terms that do not distinguish clusters.
- For example, MONDAY, TUESDAY, . . . in newspaper text

Using titles for labeling clusters

Using titles for labeling clusters

- Terms and phrases are hard to scan and condense into a holistic idea of what the cluster is about.
- Alternative: titles
- For example, the titles of two or three documents that are closest to the centroid.
- Titles are easier to scan than a list of phrases.

Cluster labeling: Example

		labeling method		
	# docs	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
- All three methods do a pretty good job.

Outline

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

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

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

$\mathit{leaves} \leftarrow \{\omega_0\}$

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

do $\omega_k \leftarrow \textsc{PickClusterFrom}(\mathit{leaves})$

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

$\quad \mathit{leaves} \leftarrow \mathit{leaves} \setminus \{\omega_k\} \cup \{\omega_i, \omega_j\}$

return *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 are also $O(N^2)$ algorithms for complete-link, centroid and GAAC.

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.

Outline

- Introduction to hierarchical clustering
- Single-link and complete-link clustering
- Centroid and group-average agglomerative clustering (GAAC)
- Bisecting K-means
- How to label clusters automatically

Resources

- Chapter 17 of IIR
- Resources at `http://cislmu.org`
    - Columbia Newsblaster (a precursor of Google News): McKeown et al. (2002)
    - Bisecting K-means clustering: Steinbach et al. (2000)
    - PDDP (similar to bisecting K-means; deterministic, but also less efficient): Saravesi and Boley (2004)